# **OPTIMASI LEACH PROTOCOL DENGAN METODE K-MEANS++ CLUSTERING PADA WIRELESS SENSOR NETWORK (WSN)**

**SKRIPSI**

****

**PUTU MAS ANGGITA PUTRA**

**NIM. 1708561007**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS UDAYANA**

**JIMBARAN**

**2021**

# **SURAT PERNYATAAN KEASLIAN KARYA ILMIAH**

Yang bertanda tangan di bawah ini menyatakan bahwa naskah Skripsi dengan judul:

**OPTIMASI LEACH *PROTOCOL* DENGAN METODE *K-MEANS++* *CLUSTERING* PADA *WIRELESS SENSOR NETWORK* (WSN)**

Nama : Putu Mas Anggita Putra

NIM : 1708561007

Program Studi : Teknik Informatika

E-mail : [anggitaputra13@gmail.com](mailto:anggitaputra13@gmail.com)

Nomor telp/HP : 0895370008969

Alamat : Jl. RY Sesetan Gang VI No 9.B, Br.Ambengan, Pedungan,

Denpasar Selatan, Bali

Belum pernah dipublikasikan dalam dokumen skripsi, jurnal nasional maupun internasional atau dalam prosiding manapun, dan tidak sedang atau akan diajukan untuk publikasi di jurnal atau prosiding manapun. Apabila di kemudian hari terbukti terdapat pelanggaran kaidah – kaidah akademik pada karya ilmiah saya, maka saya bersedia menanggung sanksi-sanksi yang dijatuhkan karena kesalahan tersebut, sebagaimana diatur oleh Peraturan Menteri Pendidikan Nasional Nomor 17 Tahun 2010 tentang Pencegahan dan Penanggulangan Plagiat di Perguruan Tinggi.

Demikian Surat Pernyataan ini saya buat dengan sesungguhnya untuk dapat dipergunakan bilamana diperlukan.

Denpasar, 23 Februari 2021

Yang membuat pernyataan,

Putu Mas Anggita Putra

NIM. 1708561007

# **LEMBAR PENGESAHAN TUGAS AKHIR**

Judul : Optimasi LEACH *Protocol* Dengan Metode *K-Means++*

*Clustering* Pada *Wireless Sensor Network* (WSN)

Nama : Putu Mas Anggita Putra

NIM : 1708561007

Tanggal Disetujui :

Disetujui Oleh:

|  |  |
| --- | --- |
| Pembimbing I,  I Gusti Agung Gede Arya Kadnyanan, S.Kom., M.Kom  NIP.198503152010121007 | Pembimbing II,  Dr. Ngurah Agus Sanjaya ER, S.Kom., M.Kom  NIP.197803212005011001 |

Mengetahui,

Komisi Seminar dan Tugas Akhir

Program Studi Informatika FMIPA Universitas Udayana

I Gusti Ngurah Anom Cahyadi Putra, S.T., M.Cs.

NIP.198403172019031005

Judul : Optimasi Leach *Protocol* Dengan Metode *K-Means++* *Clustering*

Pada *Wireless Sensor Network* (WSN)

Nama : Putu Mas Anggita Putra

Pembimbing : 1. I Gusti Agung Gede Arya Kadnyanan, S.Kom., M.Kom

2. Dr. Ngurah Agus Sanjaya ER, S.Kom., M.Kom

# ABSTRAK

Title : Optimasi Leach *Protocol* Dengan Metode *K-Means++* *Clustering*

Pada *Wireless Sensor Network* (WSN)

Name : Putu Mas Anggita Putra

Supervisor : 1. I Gusti Agung Gede Arya Kadnyanan, S.Kom., M.Kom

2. Dr. Ngurah Agus Sanjaya ER, S.Kom., M.Kom

ABSTRACT

# **KATA PENGANTAR**

Proposal penelitian ini dengan judul Optimasi Leach Protokol dengan Metode *K-Means++ Clustering* Pada *Wireless Sensor Network* (WSN) ini disusun dalam rangkaian kegiatan pelaksanaan Tugas Akhir di Program Studi Informatika FMIPA UNUD. Proposal ini disusun dengan harapan dapat menjadi pedoman dan arahan dalam melaksanakan penelitian di atas.

Sehubungan dengan telah terselesaikannya proposal ini, maka diucapkan terima kasih dan penghargaan kepada berbagai pihak yang telah membantu pengusul, antara lain:

1. Bapak I Gusti Agung Gede Arya Kadyanan, S.Kom., M.Kom sebagai Pembimbing I yang telah banyak membantu menyempurnakan proposal ini.
2. Bapak Dr. Ngurah Agus Sanjaya ER, S.Kom., M.Kom sebagai Pembimbing II yang telah banyak membantu menyempurnakan proposal ini.
3. Bapak I Komang Ari Mogi, S.Kom., M.Kom. sebagai pembimbing di penjurusan jaringan sensor nirkabel yang telah banyak membantu menyempurnakan proposal ini,
4. Bapak-bapak dan ibu-ibu dosen di Program Studi Informatika, yang telah meluangkan waktu turut memberikan saran dan masukan dalam penyempurnaan proposal ini;
5. Kawan-kawan di Program Studi Informatika yang telah memberikan dukungan moral dalam penyelesaian proposal ini.

Disadari pula bahwa sudah tentu proposal ini masih mengandung kelemahan dan kekurangan. Memperhatikan hal ini, maka masukan dan saran-saran penyempurnaan sangat diharapkan.

Jimbaran, 23 Februari 2021

Penyusun

# **DAFTAR ISI**

[LEMBAR PENGESAHAN i](#_Toc64900842)

[SURAT PERNYATAAN KEASLIAN KARYA ILMIAH ii](#_Toc64900843)

[LEMBAR PENGESAHAN TUGAS AKHIR iii](#_Toc64900844)

[ABSTRAK iv](#_Toc64900845)

[KATA PENGANTAR vi](#_Toc64900846)

[DAFTAR ISI vii](#_Toc64900847)

[DAFTAR TABEL ix](#_Toc64900848)

[DAFTAR GAMBAR x](#_Toc64900849)

[DAFTAR LAMPIRAN xi](#_Toc64900850)

[BAB I PENDAHULUAN 1](#_Toc64900851)

[1.1 Latar Belakang 1](#_Toc64900852)

[1.2 Rumusan Masalah 2](#_Toc64900853)

[1.3 Batasan Masalah 2](#_Toc64900854)

[1.4 Tujuan Penelitian 3](#_Toc64900855)

[1.5 Manfaat Penelitian 3](#_Toc64900856)

[1.6 Metodelogi Penelitian 4](#_Toc64900857)

[1.6.1 Analisis Sistem 4](#_Toc64900858)

[1.6.2 Pengumpulan Data 4](#_Toc64900859)

[1.6.3 Desain Penelitian 5](#_Toc64900860)

[BAB II TINJAUAN PUSTAKA 7](#_Toc64900861)

[2.1 Tinjauan Empiris 7](#_Toc64900862)

[2.2 Tinjauan Teoritis 10](#_Toc64900863)

[2.2.1 Wireless Sensor Network 10](#_Toc64900864)

[2.2.2 Node Sensor 11](#_Toc64900865)

[2.2.3 Routing Protokol 12](#_Toc64900866)

[2.2.4 LEACH Protokol 13](#_Toc64900867)

[2.2.5 Fitur-Fitur LEACH Protokol 14](#_Toc64900868)

[2.2.6 Algoritma LEACH 15](#_Toc64900869)

[2.2.7 K-Means Clustering 16](#_Toc64900870)

[2.2.8 K-Means++ Clustering 20](#_Toc64900871)

[2.2.9 Energy Dissipation Model 21](#_Toc64900872)

[2.2.10 Matlab 2015 A 22](#_Toc64900873)

[BAB III ANALISIS DAN PERANCANGAN SISTEM 23](#_Toc64900874)

[3.1 Pengumpulan Data 23](#_Toc64900875)

[3.2 Desain Penelitian 23](#_Toc64900876)

[3.3 Metode Yang Digunakan 25](#_Toc64900877)

[3.4 LEACH Dengan K-Means 26](#_Toc64900878)

[3.5 LEACH Dengan K-Means++ 29](#_Toc64900879)

[3.6 Implementasi 31](#_Toc64900880)

[3.7 Skenario Pengujian 32](#_Toc64900881)

[3.8 Pengukuran dan Hasil Simulasi 33](#_Toc64900882)

[BAB IV HASIL DAN PEMBAHASAN 35](#_Toc64900883)

[4.1. Implementasi Sistem 35](#_Toc64900884)

[4.1.1 Deklarasi Variabel & Membentuk Topologi WSN 35](#_Toc64900885)

[4.1.2 Setup Phase K-Means 37](#_Toc64900886)

[4.1.3 Setup Phase K-Means++ 43](#_Toc64900887)

[4.1.4 Steady State Phase 50](#_Toc64900888)

[4.2. Hasil dan Pengujian Sistem 53](#_Toc64900889)

[4.2.1 Hasil Clustering Metode K-Means 54](#_Toc64900890)

[4.2.2 Hasil Clustering Metode K-Means++ 56](#_Toc64900891)

[4.2.3 Pengujian Hasil Total Konsumsi Energi 59](#_Toc64900892)

[4.2.4 Pengujian Hasil Jumlah *Death Node* 60](#_Toc64900893)

[4.2.5 Pengujian Hasil Jumlah Node Hidup 61](#_Toc64900894)

[BAB V PENUTUP 63](#_Toc64900895)

[5.1 Kesimpulan 63](#_Toc64900896)

[5.2 Saran 63](#_Toc64900897)

[DAFTAR PUSTAKA 65](#_Toc64900898)

[LAMPIRAN 67](#_Toc64900899)

# **DAFTAR TABEL**

[Table 2.1 Contoh Data Posisi Node 18](#_Toc64901528)

[Table 2.2 Perhitungan Centroid Terdekat Antar Node 19](#_Toc64901529)

[Table 2.3 Perhitungan Centroid Terdekat Antar Node Final 20](#_Toc64901530)

[Table 2.4 Parameter Simulasi 33](#_Toc64901531)

[Table 2.5 Parameter Uji Protokol 35](#_Toc64901532)

[Table 4.1 Deklarasi Variabel ……………………………………………………36](#_Toc64901511)

[Table 4.2 Pembentukan Topologi WSN 37](#_Toc64901512)

[Table 4.3 Proses Penentuan Titik Pusat Klaster Algoritma K-Means 39](#_Toc64901513)

[Table 4.4 Proses Pengelompokan Klaster Algoritma K-means 40](#_Toc64901514)

[Table 4.5 Pengecekan Titik Pusat Klaster Algoritma K-Means 42](#_Toc64901515)

[Table 4.6 Pemilihan CH Algoritma K-Means 43](#_Toc64901516)

[Table 4.7 Penentuan Titik Pusat Klaster Algoritma K-Means++ 45](#_Toc64901517)

[Table 4.8 Proses Pengelompokan Klaster Algoritma K-Means++ 47](#_Toc64901518)

[Table 4.9 Pengecekan Titik Pusat Klaster Algoritma K-Means++ 48](#_Toc64901519)

[Table 4.10 Pemilihan CH algoritma K-Means++ 49](#_Toc64901520)

[Table 4.11 Perhitungan Energi Pada Node Biasa 52](#_Toc64901521)

[Table 4.12 Perhitungan Energi Node Sebagai CH 53](#_Toc64901522)

[Table 4.13 Perhitungan Total Energi, Node Mati & Node Hidup 54](#_Toc64901523)

[Table 4. 14 Data Titik Pusat Klaster Algoritma K-Means 55](#_Toc64901524)

[Table 4.15 Ploting Hasil Clustering Algoritma K-Means 56](#_Toc64901525)

[Table 4.16 Data Titik Pusat Klaster Algoritma K-Means++ 58](#_Toc64901526)

[Table 4.17 Ploting Hasil Clustering Algoritma K-Means++ 59](#_Toc64901527)

# **DAFTAR GAMBAR**

[Gambar 2.1 Skema WSN 11](#_Toc64901718)

[Gambar 2.2 Struktur Pada Node 11](#_Toc64901719)

[Gambar 2.3 Routing Protokol 13](#_Toc64901720)

[Gambar 2.4 Skema LEACH Protokol 14](#_Toc64901721)

[Gambar 2.5 Fase Pada LEACH 16](#_Toc64901722)

[Gambar 2.6 Algoritma K-Means 17](#_Toc64901723)

[Gambar 2.7 Energy Dissipation Model 22](#_Toc64901724)

[Gambar 2.8 Software MATLAB 22](#_Toc64901725)

[Gambar 3.1 Desain Alur Sistem 25](#_Toc64901806)

Gambar 3.2 Flowchart Rancangan Program 27

[Gambar 3.3 *Flowchart Setup Phase* LEACH Dengan *K-Means* 28](#_Toc64901808)

[Gambar 3.4 Flowchart Steady State LEACH 29](#_Toc64901809)

[Gambar 3.5 Flowchart Setup Phase KM-LEACH 31](#_Toc64901810)

[Gambar 4.1 Topologi WSN 38](#_Toc64901914)

[Gambar 4.2 Plotting Titik Pusat Klaster Algoritma K-Means 56](#_Toc64901915)

[Gambar 4.3 Plotting Hasil Clustering Dengan Algoritma K-Means 57](#_Toc64901916)

[Gambar 4.4 Titik Pusat Klaster Algoritma K-Means++ 59](#_Toc64901917)

[Gambar 4.5 Plotting Hasil Clustering Dengan Algoritma K-Means++ 60](#_Toc64901918)

[Gambar 4.6 Hasil Perbandingan Total Konsumsi Energi 61](#_Toc64901919)

[Gambar 4.7 Hasil Perbandingan Jumlah Dead Node 62](#_Toc64901920)

[Gambar 4.8 Perbandingan Jumlah Node Alive 63](#_Toc64901921)

# **DAFTAR LAMPIRAN**

[Lampiran 1. Titik Kordinat Node 68](#_Toc64902432)

[Lampiran 2. Data Bilangan Acak 72](#_Toc64902433)

[Lampiran 3. Data Total Konsumsi Energi 86](#_Toc64902434)

[Lampiran 4. Data Jumlah Death Node 100](#_Toc64902435)

[Lampiran 5. Data Jumlah Node Alive 114](#_Toc64902436)

# **BAB I PENDAHULUAN**

## **Latar Belakang**

Perkembangan teknologi di bidang komunikasi dewasa ini sangatlah pesat. Salah satu teknologi yang banyak diterapkan saat ini adalah *Wireless Sensor Network* (WSN). *Wireless Sensor Network* dapat terdiri atas ratusan hingga ribuan *node*, dimana setiap *node* memiliki kemampuan untuk *sensing, processing data* dan *wireless communication.* Masing-masing *node* menggunakan baterai sebagai *supply node sensor* yang menyebabkan penggunaan energi menjadi salah satu point penting dalam membangun WSN. Penggantian baterai pada *node sensor* menjadi sulit untuk dilakukan mengingat *node sensor* diletakkan secara acak dalam jaringan dan dalam kondisi tertentu sulit untuk diketahui posisi dari semua *node sensor* yang ada pada WSN. Tentunya hal tersebutnya menyebabkan terganggunya stabilitas jaringan pada WSN dan mempengerahui kinerja jaringan dalam melakukan *sensing* atau *processing data* (Tonapo & Ratna W, 2016). Oleh sebab itu, diperlukan solusi lain untuk mengatasi keterbatasan energi pada WSN, yaitu dengan memilih *routing protocol* yang hemat energi.

*Routing protocol* adalah metode untuk menentukan jalur terdekat dalam proses *transfer dat*a dari *node* menuju *base station*. Banyak penelitian dilakukan untuk menganalisis penggunaan *routing protocol* yang tepat dalam mengoptimalkan penggunaan energi pada WSN. Algoritma pertama yang dikembangkan adalah *routing protocol* *Low-Energy Adaptive Clustering Hierarchy* (LEACH). Namun Algoritma LEACH memiliki beberapa kekurangan diantaranya seperti sisa energi *node* tidak dipertimbangkan dalam proses pemilihan *Cluster head* (CH). Oleh karena itu, kegagalan CH karena energi sisa yang rendah tidak dapat dihindari. Kemudian dalam pemilihan CH juga tidak mempertimbangkan jarak antara CH dengan *Base Station*. Sehingga konsumsi daya pada CH yang terletak jauh dari *Base Station* lebih tinggi dibandingkan yang di dekat *Base Station* (Ridwan dkk, 2020).

Pada penelitian sebelumnya oleh (Saheb & Sharma, 2017) menggunakan algoritma *K-means Clustering* untuk mengoptimalisasi pemilihan *cluster head* dalam sebuah klaster untuk membuat klaster yang seragam dan dapat mengurangi jarak antara *cluster head* dengan *node* lainnya. Hasil Penelitian ini menunjukan klaster yang dihasilkan simetris dan secara signifikan mengurangi jarak antara *node* dan *cluster head* yang mana memiliki konsumsi energi *node* yang seimbang dengan peningkatan masa hidup jaringan.

Oleh sebab itu, pada penelitian ini bertujuan untuk mengoptimasi protokol LEACH dengan menggunakan algoritma pengembangan dari *k-means*, yaitu dengan menggunakan algoritma *K-means++* untuk menjadi algoritma LEACH yang baru yaitu KM-LEACH. KM-LEACH merupakan optimasi dari protokol LEACH dimana mengoptimalkan algoritma *K-means++ Clustering* untuk lebih mengoptimalkan pemilihan *cluster head* yang akan dipilih berdasarkan energi yang tersisa. Sehingga diharapkan dengan pengoptimalan pemilihan *cluster head*, dapat mengurangi masalah tidak meratanya distribusi *cluster head,* konsumsi energi yang tidak merata dan meningkatkan stabilitas energi dalam *Wireless Sensor Network* (WSN).

## **Rumusan Masalah**

Rumusan masalah yang akan dijadikan acuan dalam penelitian ini yaitu:

1. Bagaimana pengaruh penerapan KM-LEACH terhadap efisiensi penggunaan energi (total konsumsi energi/Joule) pada WSN
2. Bagaimana kinerja perbandingan dari protokol LEACH optimasi dengan *k-means* dan KM-LEACH dioptimasi dengan *k-means++* yang sudah dioptimasi dalam segi total konsumsi energi, jumlah *node* yang mati dan jumlah *node* hidup per *round* pada WSN.

## **Batasan Masalah**

Agar penelitian tidak keluar dari topik penelitian, maka penelitian akan dibatasi sebagai berikut:

1. Simulasi dilakukan terbatas hanya dengan protokol *routing* LEACH untuk optimalisasi serta hanya menggunakan algoritma *K-Means++ Clustering*
2. Penelitian akan dilakukan secara simulasi menggunakan program Matlab 2015a.
3. Parameter kinerja yang diukur dan diamati adalah penggunaan total konsumsi energi (Joule), jumlah *node* yang mati dan jumlah *node* hidup per *round*.
4. Perancangan WSN hanya menggunakan 100 *node* sensor + 1 *base station* yang disimulasikan.
5. Pengujian dilakukan selama 400 *round.*

## **Tujuan Penelitian**

Adapun tujuan dari dilakukannya penelitian ini yaitu:

1. Menerapkan algoritma *K-means++ Clustering* sebagai optimalisasi dari Protokol LEACH serta mengetahui pengaruhnya terhadap efisiensi penggunaan energi (total konsumsi energi/Joule) pada WSN
2. Membandingkan kinerja protokol LEACH dioptimasi dengan *K-Means* dengan KM-LEACH dioptimasi dengan *K-Means++* dalam segi total konsumsi energi, jumlah *node* yang mati dan jumlah mode hidup per *round* pada WSN.

## **Manfaat Penelitian**

Manfaat yang diperoleh dari penelitian ini yaitu:

1. Bagi penulis

Hasil penelitian ini nantinya diharapkan dapat memberikan gambaran tentang kinerja dari protokol LEACH dioptimasi dengan *k-means* dan protokol LEACH dioptimasi dengan *k-means++* (KM-LEACH) pada *wireless sensor network.*

1. Bagi pihak lain

Hasil penelitian ini nantinya diharapkan dapat bermanfaat, menjadi referensi untuk menambah wawasan tentang *Wireless Sensor Network* khususnya mengenai penggunaan protokol LEACH terhadap serta bagaimana protokol ini bekerja.

## **Metodelogi Penelitian**

Pada metodologi penelitian akan dijelaskan langkah-langkah dalam penyusunan penelitian yang akan dibuat mengenai Optimasi LEACH *Protocol* Dengan Metode *K-Means++ Clustering* Pada *Wireless Sensor Network* (WSN). Sub bab bahasan yang akan dijelaskan meliputi pengumpulan data, desain penelitian, dan metode yang digunakan.

### **Analisis Sistem**

Pada tahap ini dilakukan identifikasi perkiraan terkait kebutuhan yang diperlukan terkait penelitian yang akan dilakukan. Kebutuhan yang dimaksudkan terkait materi penunjang penelitian terkait penerapan routing protokol LEACH, penggunaan algoritma *K-Means* dan *K-Means++* serta perancangan proses simulasi yang akan dilakukan dalam penelitian ini.

### **Pengumpulan Data**

Data yang akan digunakan dalam penelitian ini diperoleh dari hasil perbandingan simulasi pada aplikasi MATLAB 2015a dengan beberapa parameter uji untuk mendapatkan data:

* 1. Total konsumsi energi (Joule)
  2. Jumlah *node* hidup selama 400 round
  3. Jumlah death *node* selama 400 round

Dimana data yang didapat menggunakan skenario uji berdasarkan parameter dari penelitian sebelumnya oleh (Saheb & Sharma, 2017). Dimana nantinya parameter simulasi tersebut akan digunakan untuk menjadi skenario simulasi untuk mendapatkan data primer pada penelitian ini. Dimana skenario yang dirancang akan menggunakan 100 *node* yang disebar pada titik kordinat x dan y dengan luas daerah 100 x 100 untuk membuat skema topologi jaringan WSN.

### **Desain Penelitian**

Pada penelitian ini menggunakan proses simulasi yang merupakan suatu Teknik untuk meniru suatu proses-proses yang terjadi pada suatu sistem dengan bantuan perangkat komputer serta dilandasi dengan beberapa asumsi-asumsi tertentu sehingga sistem tersebut dapat dipelajari secara ilmiah (Kelton, 1991). Proses simulasi yang akan dilakukan dalam penelitian ini akan terdiri dari 2 proses utama yaitu proses *Setup Phase* atau pembentukan klaster dan proses *Steady State* atau transmisi data. Pada fase *Setup Phase* akan dimodifikasi dengan menggunakan algoritma yang ditentukan dalam penelitian itu yaitu algoritma *K-Means* dan *K-Means*++.

Kemudian setelah dilakukan proses simulasi akan didapatkan hasil dari penggunaan kedua metode yang nantinya akan dibandingkan hasilnya dengan menggunakan beberapa parameter uji untuk mengetahui metode mana yang lebih efisien dalam penggunaan energi diantara kedua metode LEACH yang dimodifikasi dengan algoritma *K-Means* dan *K-Means++* tersebut.

1. **Metode Yang Digunakan**

Metode yang digunakan dalam penelitian ini adalah algoritma *K-Means* dan algoritma *K-Means++* dalam memodifikasi protokol LEACH. Kedua algoritma tersebut diterapkan pada proses *setup phase* dalam protokol LEACH. Dalam proses setup phase atau pembentukan klaster, algoritma *K-Means* dan *K-Means++* memiliki perbedaan yaitu dalam algoritma *K-Means* dalam penentuan klaster diawal dilakukan secara acak sedangkan pada algoritma *K-Means++* dimana penentuan klaster pertama dipilih berdasarkan jarak terjauh dengan *Base Station*, kemudian klaster lainnya akan dipilih berdasarkan jarak terjauh dengan klaster sebelumnya dengan menggunakan rumus *ecludien distance.* Hasil klaster yang akan didapatkan antara 2 algoritma tentunya berbeda sehingga akan berdampak juga dalam proses transmisi energi pada proses steady state.

1. **Skenario Pengujian**

Skenario pengujian merupakan salah satu hal yang penting dalam pelaksanaan sebuah penelitian. Dengan adanya skenario pengujian, penelitian dapat berjalan sesuai dengan apa yang direncanakan dan diharapkan akan mendapatkan hasil yang sesuai. Dalam penelitian ini seteleh dilakukan proses simulasi akan didapatkan hasil penelitian yaitu:

* Total Konsumsi Energi (Joule)
* Jumlah *Node* Hidup selama 400 *round*
* Jumlah *Node* Mati selama 400 *round*

Kemudian hasil tersebut akan dibandingkan antara keduan metode yaitu LEACH dengan algoritma *K-Means* dan LEACH dengan algoritma *K-Means++* untuk diketahui protokol LEACH mana yang lebih efisien dan hemat energi pada *Wireless Sensor Network* (WSN).

# **BAB II TINJAUAN PUSTAKA**

Pada bagian ini, akan dipaparkan sejumlah tinjauan empiris dan tinjauan teoritis yang dijadikan sebagai acuan dalam penelitian mengenai optimasi LEACH *Protocol* dengan metode *K-Means Clustering* pada *wireless sensor network* (WSN) yang dilakukan ini.

## **Tinjauan Empiris**

Pada penelitian ini digunakan beberapa acuan yang bersumber dari penelitian terdahulu baik dari segi metode ataupun jenis penelitian yang dilakukan. Beberapa penelitian terkait yang penulis jadikan acuan antara lain:

1. ***Improved LEACH Protocol Based on K-Means Clustering Algorithm for Wireless Sensor Network*** (Saheb & Sharma, 2017)

Pada penelitian ini menggunakan algoritma *K-means Clustering* untuk mengoptimalisasi pemilihan *cluster head* dalam sebuah klaster untuk membuat klaster yang seragam dan dapat mengurangi jarak antara *cluster head* dengan *nod*e lainnya. Penelitian ini juga menambahkan parameter residual energi dalam penentuan *cluster head*. Penelitian ini menggunakan 100 *node* yang disimulasikan pada software MATLAB dimana *node* disebar pada titik kordinat dengan luas area100 x 100 . Dimana simulasi berdasarkan 2 parameter uji yaitu jumlah *node* yang hidup dan total konsumsi energi untuk membandingkan kinerja protokol LEACH dengan LEACH yang dioptimasi dengan algoritma *K-Means*.

Hasil Penelitian ini menunjukan klaster yang dihasilkan simetris dan secara signifikan mengurangi jarak antara *node* dan *cluster head* yang mana meliki konsumsi energi *node* yang seimbang dengan peningkatan masa hidup jaringan. Persamaan dengan penelitian yang akan dilakukan adalah sama-sama menggunakan algoritma *K-Means* untuk mengatasi masalah penentuan *cluster head* yang optimal. Perbedaannya terletak pada pada penelitian yang akan dilakukan menggunakan algoritma pengembangan dari *K-Means* yaitu algoritma *K-Means++.*

1. **Optimasi Protokol LEACH Untuk Meningkatkan Stabilitas Pada *Wireless Sensor Network*** (Ridwan dkk, 2020)

Pada penelitian ini mencoba mengatasi kekurangan dari protokol LEACH yaitu pemilihan *cluster head* yang tidak mempertimbangkan energi sisa di setiap *node*, sehingga dapat menyebabkan kegagalan dalam proses pemilihan kepala klaster dan akan sangat mempengaruhi stabilitas pada jaringan. Solusi yang ditawarkan penulis yaitu metode *Optimization*-LEACH (O-LEACH) dimana proses pemilihan *cluster head* dikembangkan berdasarkan energi awal setiap *node* dan tingkat energi awal yang baru dihitung dari setiap *node* untuk setiap putaran.

Hasil penelitian menunjukan bahwa protokol LEACH biasa hanya mampu mempertahankan 2 % dan metode O-LEACH 40 % dari 200 *node* yang disimulasikan sehingga metode O-LEACH lebih optimal. Persamaan dengan penelitian yang akan dilakukan adalah sama-sama ingin mengoptimalisasi protokol LEACH dalam penentuan *cluster head* untuk meningkatkan masa hidup jaringan. Perbedaannya terletak pada metode yang digunakan, dimana dalam penelitian ini dalam pemilihan *cluster head* berdasarkan energi awal setiap *node* setiap putaran.

1. **Enhanced LEACH Protocol For Increasing a Lifetime Of WSNs** (Salem & Shudifat, 2019)

Pada penelitian ini ingin meningkatkan masa hidup jaringan dengan mengoptimalisasi pemilihan *cluster head* dalam protokol LEACH, dimana dalam pemilihan *cluster head* dengan protokol LEACH biasa dilakukan secara random namun dalam penelitian ini pemilihan *cluster head* dimodifikasi dengan mempertimbangkan beberapa parameter seperti jarak *cluster head* dengan *base station* dan energi yang ada pada *node*. Penelitian ini menggunakan software MATLAB untuk simulasi dengan menggunakan 45-85 *node* pada titik kordinat dengan luas area 100 x 100 .

Hasil penelitian ini menunjukkan bahwa konsumsi daya diminimalkan, masa pakai jaringan dapat ditingkatkan, jumlah putaran dimaksimalkan, dan konsumsi daya yang menurun. Akhirnya, dibandingkan dengan protokol LEACH, pengembangan protokol LEACH ini dapat bekerja lebih baik. Persamaan dengan penelitian yang akan dilakukan terletak pada parameter yang digunakan sebagai acuan penentuan *cluster head* yaitu energi yang ada pada *node*. Perbedaannya pada penelitian ini tidak memperhitungkan jarak antara *cluster head* dengan *node* yang ada dalam klaster yang sama pada penentuan *cluster head* sedangkan pada penelitian yang akan dilakukan melakukannya dengan metode *K-Means++.*

1. **Perbandingan Pengaruh Nilai *Centroid* Awal Pada Algoritma *K-Means* dan *K-Means++* Terhadap Hasil *Cluster* Menggunakan Metode *Confusion Matrix*** (Nasir & Budiman, 2017)

Pada penelitian ini membandingkan tingkat akurasi antara 2 jenis algoritma, yaitu algoritma *K-Means* dan *K-Means++* pada klaster. Penelitian ini dilakukan untuk mengetahui bagaimana pengaruh dari pemilihan *centroid* awal yang berbeda dari kedua algoritma tersebut terhadap hasil klaster. Evaluasi hasil klaster yang digunakan menggunakan metode confusion matrix untuk menghitung tingkat akurasi.

Dari hasil penelitian didapatkan bahwa kedua algoritma mempunyai hasil akhir klaster dapat berubah-ubah. Hal ini dibuktikan dari percobaan yang dilakukan 100 kali pada kedua algoritma dengan data set iris, algoritma *K-Means* mempunyai 6 hasil klaster yang berbeda dan algoritma *K-Means++* mempunyai 5 hasil klaster berbeda. Algoritma *K-Means++* Mempunyai rata-rata akurasi yang lebih tinggi dari pada algoritma *K-Means*. Rata-rata tingkat akurasi algoritma *K-Means* yaitu 80,46 % dan pada algoritma *K-Means++* yaitu 83,66. Algoritma *K-Means++* mempunyai peluang lebih besar untuk mendapatkan hasil klaster terbaik, dari percobaan 100 kali, algoritma *K-Means++* terdapat 48 percobaan sedangkan algoritma *K-Means* terdapat 19 percobaan.

Persamaan dengan penelitian yang akan dilakukan adalah metode yang digunakan yaitu *K-Means++*, dimana hasil penelitian metode *K-Means++* memiliki peluang lebih baik dalam pembentukan klaster dibandingkan metode *K-Means* biasa. Hal itu yang menjadi acuan dalam penelitian yang akan dilakukan yaitu dengan metode *K-Means++* untuk mengurangi kekurangan pada protokol LEACH. Perbedaanya terletak pada studi kasus permasalahan yang akan diteliti pada penelitian ini dengan penelitian yang akan dilakukan.

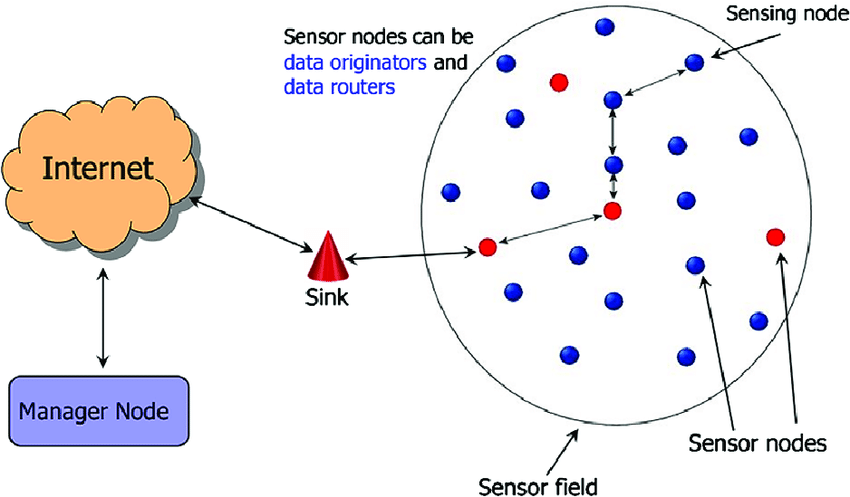
## **Tinjauan Teoritis**

Pada penelitian ini digunakan beberapa teori yang digunakan serta dijadikan acuan dalam proses penelitian, antara lain:

### **Wireless Sensor Network**

Wireless sensor network merupakan sekumpulan *node* yang dapat berupa sensor yang akan melakukan pengambilan data pada parameter ukur dan kemudian dikirimkan pada sebuah *node* sentral atau sebuah server untuk dilakukan pengolahan data. Setiap *node* *sensor* memiliki kemampuan untuk mengumpulkan data di sekitarnya dan merutekannya kembali menuju *sink node* melalui transmisi radio secara intensif. Data yang dapat dikumpulkan oleh sensor node bisa berupa tekanan suhu, pergerakan suatu benda atau objek, kelembaban tanah dan sebagainya (Ridwan dkk, 2020).

Dimana pada umumnya wireless sensor network (WSN) terdiri dari dua bagian utama yaitu *node sensor* dan *sink* (Tonapa dkk, 2016). *Node sensor* merupakan suatu perangkat yang dapat menghasilkan informasi, biasanya merupakan sebuah sensor atau juga dapat berupa sebuah aktuator yang menghasilkan feedback pada keseluruhan operasi. Secara umum *node sensor* disebar dengan volume dan kerapatan yang tinggi. *Sink* merupakan kesatuan alat yang berfungsi pengumpulan informasi dari *node sensor* sehingga dapat dilakukan pengolahan informasi lebih lanjut sebelim data disimpan ke server/clound. Dengan adanya karakteristik tersebut, perlu adanya metodologi yang mampu melampaui dan mengembangkan karakteristik-karakteristik pada WSN serta tidak membatasi pengiriman informasi, jaringan, manajemen operasional, kerahasiaan, integritas, dan proses di dalam jaringan.

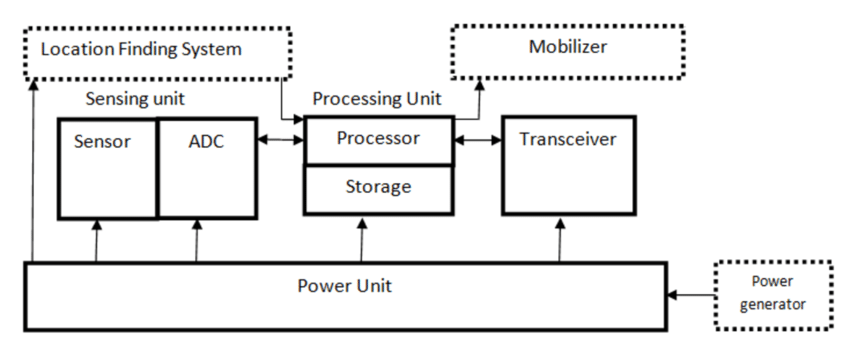


Gambar 2.1 Skema WSN

(Sumber: Mihobi dkk, 2017)

### **Node Sensor**

*Node* dalam wireless sensor network dikenal dengan istilah “mote”. Dimana pada dasarnya *node* adalah sebuah komputer, walaupun memiliki bentuk dan kemampuannya tidak seperti umumnya komputer yang digunakan saat ini karena kemampuan yang masih terbatas dan ukurannya yang cukup kecil (smart dust), tetapi fungsi *node* sama seperti komputer pada umumnya dimana sampai saat ini terus mengalami perkembangan untuk meningkat kemampuan dari *node* (Sohraby dkk, 2011). *Node* dilengkapi alat pemroses (CPU), memori, sejumlah antarmuka Input/Output yang dapat diprogram ulang (terintegrasi pada mikrokontroler), transceiver untuk komunikasi radio, sumber daya energi yang menggunakan baterai, dan beberapa peralatan tambahan yang dapat disertakan sesuai kebutuhan penggunaan *node* itu sendiri.



Gambar 2.2 Struktur Pada Node

(Sumber: Sohraby dkk, 2011. Wireless Sensor Network Technology, Protocols, Applications Book. New Jersey:John Wiley & Sons, Inc)

Pada *wireless sensor network*, *node* memiliki fungsi dan kemampuan yang berbeda-beda, berikut beberapa jenis *node* dalam WSN (Sohraby dkk, 2011):

1. Node Sensor

*Node Sensor* yaitu *node* yang berfungsi untuk membaca data lingkungan atau objek yang dipantau. Untuk keperluan pembacaan atau penginderaan, *node* ini dilengkapi dengan satu atau beberapa perangkat sensor. Dari kemampuannya, *node* ini dapat dibagi menjadi dua jenis. Pertama, *Node* dengan kemampuan standar dan kedua yaitu *Node* yang telah dilengkapi fasilitas yang lebih kaya seperti CCD camera, wireless LAN, logger, Web server dan lain-lain. *Node* jenis kedua ini juga mampu melakukan komputasi yang lebih kompleks dibanding jenis pertama.

1. Router

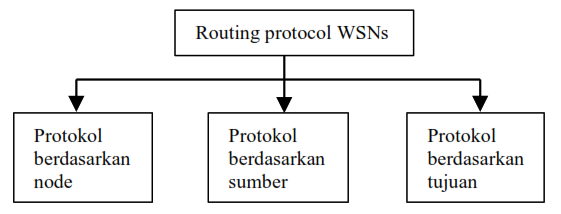
Router adalah mini komputer/*node* yang berfungsi untuk meneruskan paket data dari sebuah *node* ke *node* lain dan menghubungkan antara 2 jaringan yang berbeda. *Node* ini berguna untuk keperluan komunikasi multi-hop. Dalam aplikasi nyata, sebuah sensor *node* dapat deprogram untuk bertindak sebagai router.

1. Sink Node

*Sink node* adal­­­ah *node* yang berfungsi untuk mengumpulkan data penginderaan dari *node xensor*, kemudian meneruskannya ke perangkat lain, seperti ke database server untuk penyimpanan. Selain untuk mengumpulkan data dari *node sensor*, *sink* juga berfungsi sebagai penyebar paket dari perangkat atau sistem lain ke WSN, misalnya untuk keperluan pemrograman atau konfigurasi ulang *node sensor* secara remote.

### **Routing Protokol**

*Routing* *protocol* merupakan suatu aturan-aturan mengenai *routing* atau aktifitas penting dalam suatu jaringan untuk memilih antara 2 *node* atau lebih agar bisa saling berkomunikasi satu sama lain. Aktifitas ini dikerjakan oleh *routing* *protocol* pada *router* dengan cara memilih jalur terbaik untuk menghubungkan *node-node* yang ada untuk berkomunikasi dan transfer data.



Gambar 2.3 Routing Protokol

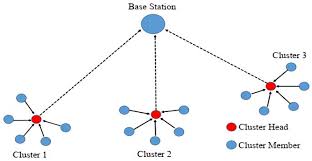
(Sumber: Novid dkk, 2019)

Pada WSN *routing protocol* dapat dibedakan menjadi 3 jenis yaitu routing protocol berdasarkan *node*, *routing protocol* berdasarkan sumber dan routing protocol berdasarkan tujuan (Novid dkk, 2019). Protokol berdasarkan *node* yang dimaksud adalah dalam penerapan routing protokol *node*-*node* yang ada dalam suatu WSN diklasifikasikan menjadi beberapa struktur hirarki *node*. Dimana setiap *node* akan terbagi menjadi beberapa *cluster*-*cluster* yang ada untuk membantu proses komunikasi antar *node* yang ada. Salah satu protokol yang menggunakan metode ini adalah LEACH (*Low Energy Adaptive Clustering Hierarchy*)

Protokol berdasarkan sumber hanya bekerja ketika protokol diminta untuk mengirimkan data. Routing protocol akan dibuat oleh *node* sumber hingga data yang dikirim mencapai *node* tujuan. Protokol yang menggunakan *routing protocol* jenis ini adalah SPIN (*Sensor Protocol for Information via Negotiation*). Sedangkan protokol yang menggunakan *routing protocol* berdasarkan tujuan ini adalah suatu protokol yang sudah memilih jalur yang tetap, sehingga tidak perlu mengirimkan sinyal pemberitahuan terlebih dahulu sebelum melakukan pengiriman data. Dengan itu proses pengiriman data akan menjadi lebih efisien dan sumber daya yang digunakan juga akan lebih sedikit. Contoh protokol dengan *routing protocol* berdasarkan tujuan yakni protokol *Direct Diffusion* (DD).

### **LEACH Protokol**

LEACH merupakan salah satu routing protocol yang digunakan pada *wireless sensor network* (WSN). Protokol ini dimulai dengan pemilihan suatu *node* sebagai *cluster head* (CH) lalu dengan algoritma *clustering* memilih *node* non-CH sebagai anggota sehingga membentuk klaster. Mekanisme ini menghemat energi karena hanya CH yang transmisi data ke *base station*, sedangkan tiap *node sensor* cukup mengirim data ke CH masing-masing. Akibatnya, konsumsi energi berkurang. sehingga lifetime jaringan sensor menjadi maksimal (Permana dkk, 2015). Seperti yang direpresentasikan pada gambar 2.4.



Gambar 2.4 Skema LEACH Protokol

(Sumber: Ridwan dkk, 2020)

Pada awalnya *node*-*node* tersebar dalam jumlah besar pada suatu area dan proses pengiriman data masih terpusat pada *base station*. Namun dengan adanya algoritma LEACH, *node*-*node* tersebut dikelompokkan dalam beberapa klaster pada satu jaringan. Masing-masing klaster memiliki sebuah *cluster head* yang bertugas untuk mengkoordinasi pengiriman data dari *node* *sensor* ke *base station*.

### **Fitur-Fitur LEACH Protokol**

Leach Protokol memiliki beberapa fitur diantaranya (Salem & Shudifat, 2019):

1. *Data fusion*, yaitu penggabungan data sehingga mengurangi disipasi energi dan menambah lifetime jaringan.
2. *Adaptive*, yaitu mudah menyesuaikan diri saat pembentukan formasi klaster.
3. *Local Compression*, yaitu mengkompres data agar ukuran data yang dikirim ke *base station* lebih kecil.
4. *Randomization Rotation*, yaitu perputaran kedudukan *cluster head* secara acak.
5. *Self-Organizing*, yaitu tiap *node sensor* memiliki sikap pengambilan keputusan sendiri untuk menjadi *cluster head*.

### **Algoritma LEACH**

LEACH terbagi ke dalam beberapa sesi, tergantung dari jumlah *cluster head* yang diinginkan dan masa observasi. LEACH memastikan setiap *node* akan menjadi *cluster head* untuk satu sesi. Akibatnya, kedudukan *cluster head* menjadi tidak tetap atau bergantian sehingga suatu klaster memiliki formasi yang dinamis atau berubah-ubah setiap sesi. Algoritma LEACH dibagi menjadi 2 fase yaitu fase setup dan fase steady state (Heinzelman dkk, 2002). Proses algoritma LEACH dapat dijelaskan sebagai berikut:

1. Fase Setup

Pada *fase setup* terjadi penentuan *cluster head* dan proses pembentukan klaster atau sering disebut juga dengan algoritma *clustering*. Pada fase ini beberapa *node* secara *independent* akan memilih dirinya sendiri sebagai *cluster head* berdasarkan pada level energi terkininya dan nilai *threshold* T(n) yang dirumuskan dengan:

|  |  |
| --- | --- |
|  | (1) |

Dimana P adalah presentasi *cluster head* yang diinginkan dan G adalah sekumpulan *node* yang tidak menjadi *cluster head* dan r adalah putaran saat ini dan pada 1/P *round* terakhir (Ibrian dkk, 2020). Dimana dalam pemilihan *cluster head* berdasarkan residual energi yang terbesar pada setiap *node* yang berada pada klaster yang sama. Dalam menghitung residual energi tiap node dengan menggunakan persamaan :

|  |  |
| --- | --- |
|  | (2) |

Dimana:

**=** Residual energi setiap *node* selama *round*

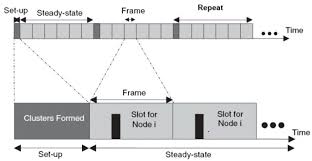
***=*** Energi setiap *node* selama *round*

***=*** Total Konsumsi energi node selama *round*

Pemilihan *cluster* tersebut meningkatkan jarak transmisi agregat antara *node* dan *cluster head* dan ini membuat lebih banyak energi yang dihabiskan dalam komunikasi *intra-cluster*. Untuk mengatasi masalah ini, mereka harus dipilih sebagai *cluster head* yang dekat dengan pusat *cluster*.

1. Fase Steady State

Pada fase ini terjadi proses transfer data antar *node* yang melibatkan aktivitas transmisi dan observasi. Proses *steady state* memakan waktu yang lebih lama dibandingkan dengan proses *setup phase,* karena proses transfer data terjadi melalui gelombang transmisi radio secara intensif. Sedangkan pada proses setup hanya terjadi proses penentuan *cluster head* dan pembentukan klaster. Pembagian fase terhadap waktu pada LEACH dapat direpresentasikan oleh gambar 6.5:

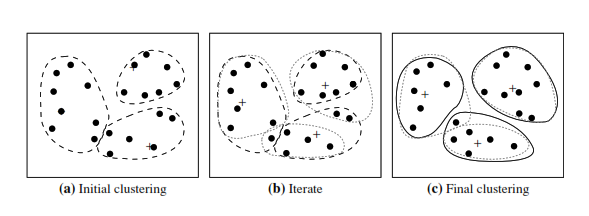


Gambar 2.5 Fase Pada LEACH

(Sumber: Ridwan dkk, 2020)

### **K-Means Clustering**

Algoritma *K-means Clustering* adalah suatu metode penganalisaan data atau metode data mining yang melakukan proses pemodelan tanpa *supervisi (unsupervised)* dan merupakan salah satu metode yang melakukan pengelompokan data dengan sistem partisi (Jiawei dkk, 2011). Terdapat dua jenis data *clustering* yang sering dipergunakan dalam proses pengelompokan data yaitu *hierarchical* dan *non-hierarchical*, dan *K-Means* merupakan salah satu metode data *clustering* *non-hierarchical* atau *partitional clustering*.



Gambar 2.6 Algoritma K-Means

(Sumber: Jiawei dkk, 2011. Data Mining Concept and Techniques. USA:Elsevier Inc)

Metode *K-Means* berusaha mengelompokkan data yang ada ke dalam beberapa kelompok, dimana data dalam satu kelompok mempunyai karakteristik yang sama satu sama lainnya dan mempunyai karakteristik yang berbeda dengan data yang ada di dalam kelompok yang lain. Dengan kata lain, metode *K-Means* bertujuan untuk meminimalisasikan fungsi objektif yang di set dalam proses *clustering* dengan cara meminimalkan variasi antar data yang ada di dalam suatu klaster dan memaksimalkan variasi dengan data yang ada di klaster lainnya.

Pada kasus ini, data berupa *node* yang dimana masing-masing *node* direpresentasikan sebagai titik kordinat yang berada pada posisi (x,y) sehingga untuk menghitung jarak pada masing-masing *node* harus mengurangi selisih antara nilai jarak x dan jarak y sehingga memodifikasi rumus Euclidean Distance menjadi seperti berikut (Fakhri M. Falahi, 2019):

|  |  |
| --- | --- |
|  | (3) |

Dimana:

J **=** jarak antara titik data *node* dengan *centroid*

= *node* ke-j pada klaster ke-i

= centroid ke-i pada klaster ke-i

Berikut adalah Langkah-langkah algoritma *k-means* (M. Ni’man Nasir & Irwan Budiman, 2017):

1. Tentukan jumlah klaster
2. Tentukan pusat awal K untuk setiap klaster secara *random*
3. Hitung jarak antara setiap titik data dari pusat mencakup mereka di titik pusat terdekat.
4. Ketika semua poin telah ditetapkan, hitung ulang posisi dari pusat K.
5. Ulangi Langkah C dan D hingga konvergensi. Ini menghasilkan pemisahan titik menjadi kelompok-kelompok yang matriksnya akan dibatasi dapat ditentukan.
6. Memperbarui pusat dengan rumus berikut

|  |  |
| --- | --- |
|  | (4) |

1. Konvergensi diperoleh ketika tidak ada variasi lagi di pusat *cluster*

Dimana dalam penelitian ini akan menggunakan data sebaran *node* sesuai dengan Table 1 Parameter Simulasi dimana *node* terdiri dari 100 *node* yang disebar pada titik koordinat 100 x 100 dengan jumlah klaster yang akan dibentuk yaitu 10 klaster. Dan berikut merupakan contoh proses perhitungan menggunakan persamaan (4) dan (5) untuk menghitung jarak antara *node* dan *centroid* serta bagaimana rumus perhitungan untuk memperbarui pusan klaster/*centroid*:

Table 2.1 Contoh Data Posisi Node

|  |  |
| --- | --- |
| Node ke (i) | Posisi *node* (x,y) meter |
| 1 | (4,3) |
| 2 | (5,4) |
| 3 | (6,7) |
| 4 | (7,8) |
| 5 | (8,6) |

Selanjutnya dari data diatas dipilih *node* ke 2 sebagai c1 dan 5 sebagai c2 sebagai titik centroid secara acak sesuai algoritma k-means, kemudian hitung jarak antara masing” *node* dengan *centroid* dengan rumus persamaan *(4)* seperti berikut:

m

m

m

605 m

m

m

m

m

m

Table 2.2 Perhitungan Centroid Terdekat Antar Node

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Node* ke (i) | Jarak ke c1 | Jarak ke c2 | Jarak terdekat | Klaster |
| 1 |  | 5 |  | c1 |
| 2 | 0 | 605 | 0 | c1 |
| 3 |  |  |  | c2 |
| 4 |  |  |  | c2 |
| 5 |  | 0 | 0 | c2 |

Setelah *node* dikelompokkan dengan *centroid* terdekat, selanjutnya kita perbaharui pusat masing-masing klaster dengan persamaan (4) kemudian hitung ulang jarak setiap *node* dengan titik *centroid* baru dengan persamaan (3) untuk mengelumpokan *node* dengan klaster terdekat.

Table 2.3 Perhitungan Centroid Terdekat Antar Node Final

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Node* ke (i) | Jarak ke c1 | Jarak ke c2 | Jarak terdekat | Klaster |
| 1 | 0.707 | 5 | 0.707 | c1 |
| 2 | 0.707 | 3.605 | 0.707 | c1 |
| 3 | 3.807 | 1 | 1 | c2 |
| 4 | 5.147 | 1 | 1 | c2 |
| 5 | 4.301 | 1.414 | 1.414 | c2 |

Karena posisi *node* pada klaster tidak mengalami perubahan maka iterasi dihentikan dan data *node* diatas merupakan hasil pengelompokan *node* dengan titik *centroid* atau klaster terdekat pada WSN.

### **K-Means++ Clustering**

Algoritma *K-Means++ Clustering* adalah salah satu algoritma yang digunakan untuk klasifikasi atau pengelompokan data. Algoritma *K-Means++* merupakan pengembangan dari algoritma *K-Means* yang dikembangkan oleh Arthur dan Vassilvitskii pada tahun 2007 (Arthur & Vassilvitskii, 2007). Pada *K-Means++* nilai *Centroid* awal tidak dilakukan secara acak seperti pada algoritma *K-Means* tetapi melalui tahap perhitungan. Setelah nilai *centroid* awal ditentukan maka tahapan selanjutnya sama persis seperti algoritma *K-Means*.

Algoritma *K-Means* dimulai dengan pemilihan K yang dipilih secara acak, K merupakan jumlah klaster yang ingin dibuat yang nilainya ditentukan secara acak dan nilai tersebut dijadikan sebagai pusat dari klaster atau *centroid*. Selanjutnya rumus *Euclidean Distance* untuk menghitung jarak dari setiap data terhadap masing-masing *centroid* sampai setiap data tersebut ditemukan jarak yang paling dekat dengan *centroid*. Setiap data diklasifikasikan berdasarkan jarak kedekatan dengan *centroid*. Langkah-langkah tersebut dilakukan berulang sampai diperoleh kestabilan dari nilai *centroid* (Falahi, 2019).

Sedangkan *K-Means++* yaitu pengembangan dari algoritma *K-Means*. Perbedaan dari algoritma *K-Means* yaitu pada pemilihan nilai awal. *K-Means++* digunakan untuk meminimalisir dampak buruk dari algoritma *K-Means* yang bergantung dari nilai awal. Rumus menentukan nilai awal pada *K-Means++:*

|  |  |
| --- | --- |
|  | (5) |

= Jarak *euclidean distance*

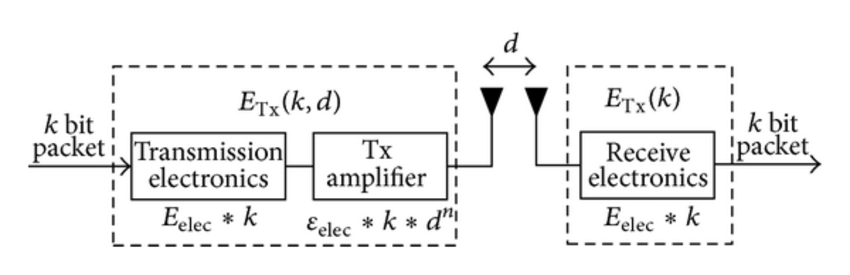
= Jumlah jarak *euclidean distance*

Adapun langkah-langkah pada algoritma *K-Means++*adalah sebagai berikut (Fox & Emily, 2016):

1. Tentukan titik centroid pertama secara acak dari pada titik data *node*.
2. Untuk setiap *node* yang diamati, hitung jarak D(x) dengan centroid terdekat dengan rumus persamaan (3).
3. Pilih centroid baru dari antara titik *node*. dengan probabilitas x dipilih proporsional dengan rumus persamaan (5), dimana posisi *node* dengan nilai K terbesar dipilih sebagai centroid.
4. Ulangi langkah B dan C hingga jumlah *centorid* sama dengan jumlah klaster yang ditentukan.
5. Lanjut ke proses algoritma *K-Means* biasa.

### **Energy Dissipation Model**

*Energy dissipation model* adalah pemodelan bagaiamana pemancar menghilangkan energi untuk menjalankan elektronik radio dan power amplifier, dan penerima membuang energi untuk menjalankan elektronik radio. Dalam model disipasi energi radio yang efektif, baik ruang bebas (𝑑 kehilangan daya) dan model saluran multipath fading (𝑑 24 daya hilang) digunakan, tergantung pada jarak antara pemancar dan penerima. Jika jaraknya kurang dari ambang 𝑑 0, ruang model akan digunakan; jika tidak, model multipath digunakan. Oleh karena itu, konsumsi energi untuk mentransmisikan pesan bit lebih jauh dapat diformulasikan seperti gambar 2.7 (Qiang dkk, 2015).



Gambar 2.7 Energy Dissipation Model

(Saheb & Sharma, 2017)

### **MATLAB 2015 A**

MATLAB merupakan singkatan dari (Matrix LABoratory) adalah sebuah lingkungan komputasi numerik dan bahasa pemrograman komputer generasi keempat. Dikembangkan oleh The MathWorks.Inc, MATLAB memungkinkan manipulasi matriks, pemplotan fungsi dan data, implementasi algoritma, pembuatan antarmuka pengguna, dan pengantarmukaan dengan program dalam bahasa lainnya. Meskipun hanya bernuansa numerik, sebuah kotak kakas (toolbox) yang menggunakan mesin simbolik MuPAD, memungkinkan akses terhadap kemampuan aljabar komputer. Sebuah paket tambahan, Simulink, menambahkan simulasi grafis multi ranah dan desain berdasar-model untuk sistem terlekat dan dinamik.



Gambar 2.8 Software MATLAB

(Sumber: ch.mathworks.com)

Matlab memiliki karakteristik yang berbeda dengan bahasa pemrograman lain yang sudah ada lebih dahulu seperti Delphi, Basic maupun C++. Matlab merupakan bahasa pemrograman level tinggi yang dikhususkan untuk kebutuhan komputasi teknis, visualisasi dan pemrograman seperti komputasi matematik, analisis data, pengembangan algoritma, simulasi dan pemodelan dan grafik-grafik perhitungan.

Matlab hadir dengan membawa warna yang berbeda. Hal ini karena matlab membawa keistimewaan dalam fungsi-fungsi matematika, fisika, statistik, dan visualisasi. Matlab dikembangkan oleh MathWorks, yang pada awalnya dibuat untuk memberikan kemudahan mengakses data matrik pada proyek LINPACK dan EISPACK. Saat ini matlab memiliki ratusan fungsi yang dapat digunakan sebagai problem solver mulai dari simpel sampai masalah-masalah yang kompleks dari berbagai disiplin ilmu.

# **BAB III ANALISIS DAN PERANCANGAN SISTEM**

## **Pengumpulan Data**

Data yang akan digunakan dalam penelitian ini diperoleh dari hasil perbandingan simulasi pada aplikasi MATLAB 2015a dengan beberapa parameter uji untuk mendapatkan data:

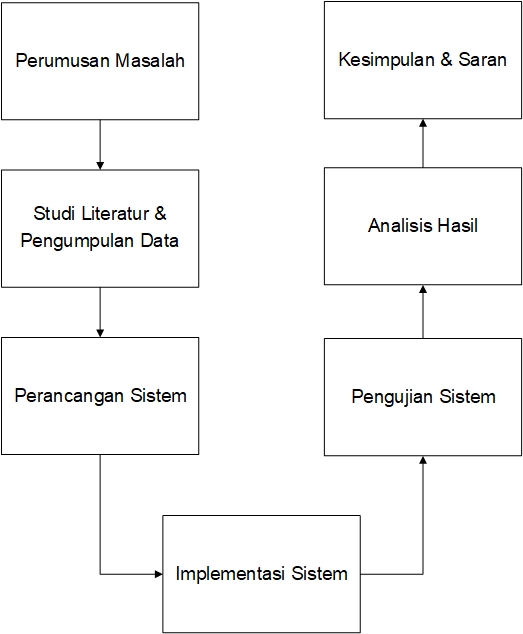
1. Total konsumsi energi (Joule)
2. Jumlah *node* hidup selama 400 round
3. Jumlah death *node* selama 400 round

Dimana data yang didapat menggunakan skenario uji berdasarkan parameter dari penelitian sebelumnya oleh (Saheb & Sharma, 2017). Namun terdapat beberapa penyesuaian dimana dalam penelitian ini jumlah round dibatasi dan satuan nilai proses transmisi energi ditingkatkan untuk mengurangi jumlah round yang dilakukan. Dari proses simulasi yang dilakukan akan didapatkan data primer dari penelitian ini. Dimana skenario yang dirancang akan menggunakan 100 *node* yang disebar pada titik kordinat x dan y dengan luas daerah 100 x 100 untuk membuat skema topologi jaringan WSN.

Kemudian setelah dilakukan proses simulasi, data yang didapatkan direpresentasikan dalam bentuk grafik pada software MATLAB yang menampilkan data total konsumsi energi (joule), jumlah *node* mati dan *node* hidup selama 400 round simulasi, setelah itu hasil dibandingkan dengan hasil penelitian sebelumnya untuk membuktikan apakah kinerja dari protokol KM-LEACH yang dioptimasi dengan algoritma *K-Means++* lebih baik daripada protokol LEACH yang dioptimasi dengan algoritma *K-Means* saja. Selain itu dilakukan juga studi literatur untuk memberikan argumen yang logis, serta memperkuat hasil dari pengujian tersebut.

## **Desain Penelitian**

Tahapan penelitian pada perancangan simulasi ini melalui beberapa tahapan yang dilakukan seperti flowchart pada gambar 3.1:



Gambar 3.1 Desain Alur Sistem

(Sumber: Kadnyanan, 2018)

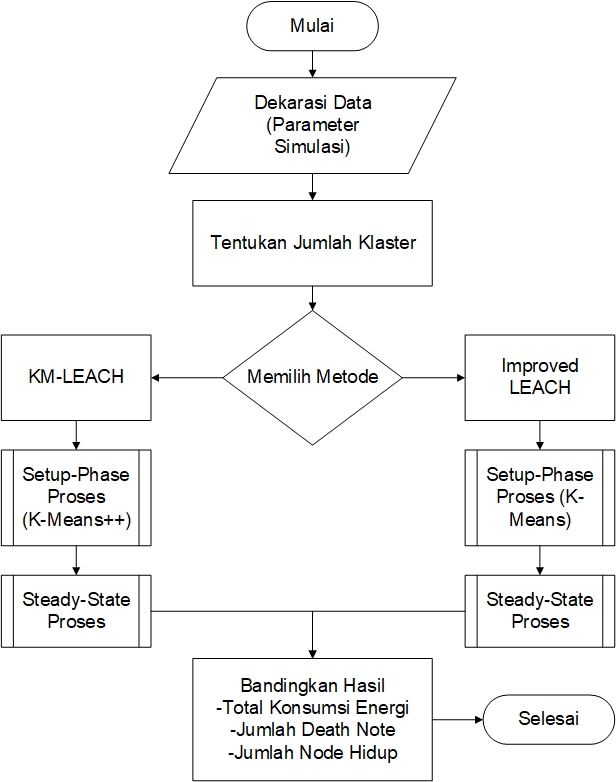
Sesuai gambar 3.1, Desain ini merupakan adaptasi dari sekema waterfall model yang diambil berdasarkan penelitian sebelumnya oleh (Kadnyanan, 2018). Dimana sesuai dengan desain metode pada gambar 3.1, pada tahap pertama yang akan dilakukan pada penelitian ini adalah perumusan masalah terkait permasalahan yang akan diangkat dalam penelitian ini. Setelah ditetapkan permasalahan yang akan diteliti, tahap kedua yang dilakukan adalah studi literatur dan pengumpulan data terkait permasalahan yang diangkat dan informasi-informasi atau kajian teori yang menunjang penelitian. Tahap ketiga dilakukan perancangan sistem untuk mengetahui alur dan segala keperluan sistem yang akan dibuat.

Barulah pada tahap keempat dilakukan implementasi sistem yang mana implementasi sistem akan dilakukan dalam bentuk simulasi dan kemudian dilakukan pengujian serta analisis sistem untuk mengetahui hasil akhir penelitian yang dilakukan, untuk mengetahui apakah penelitian yang dilakukan berhasil atau tidak. Dan di tahap akhir adalah kesimpulan & saran yang berisi hasil dari penelitian yang dilakukan dan saran atau arahan untuk pengembangan selanjutnya yang dapat dilakukan terkait penelitian yang diteliti.

Sistem ini dapat diimplementasikan secara nyata pada masing-masing *node* dan *base station* pada WSN. Dimana masing-masing *node* dapat terdiri dari perangkat Arduion uno, Xbee sensor ataupun jenis *node sensor* lainnya, dimana nantinya sistem optimasi protokol LEACH dengan *K-Means++* tersebut ditanamkan atau di program pada perangkat tersebut dalam jaringan WSN dengan menggunakan bahasa pemrograman C/C++.

## **Metode Yang Digunakan**

Pada penelitian ini akan menggunakan 2 metode yaitu algoritma LEACH dengan *K-Means* dan *K-Means++.* Setelah itu akan dibandingkan kinerja antara protokol LEACH dengan *K-Means* dan protokol LEACH dengan *K-Means++* untuk mendapatkan hasil penelitian. Berikut adalah gambaran program yang akan dibuat secara umum yang direpresentasikan pada gambar 3.2:

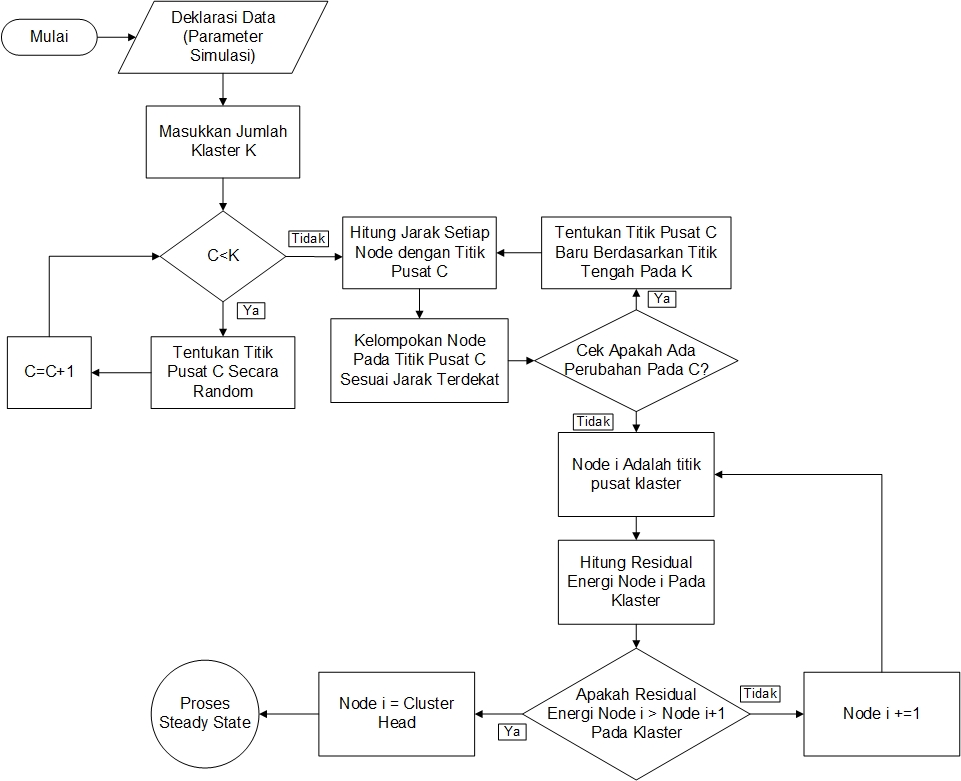


Gambar 3.2 Flowchart Rancangan Program

## **LEACH Dengan K-Means**

Secara umum algoritma LEACH memiliki 2 proses utama yaitu *Setup Phase* dan *Steady State*. Pada proses *Setup Phase* inilah akan dimodifikasi dengan menggunakan algortima *K-Means*. Algoritma *K-Means* berperan dalam proses pembentukan klaster dimana dengan bantuan algoritma *K-Means* dapat membantuk membentuk klaster yang lebih baik dibandingkan algoritma *clustering* biasa pada algoritma LEACH. Berikut adalah flowchart protokol LEACH yang sudah dioptimasi dengan *K-Means* pada gambar 3.3:

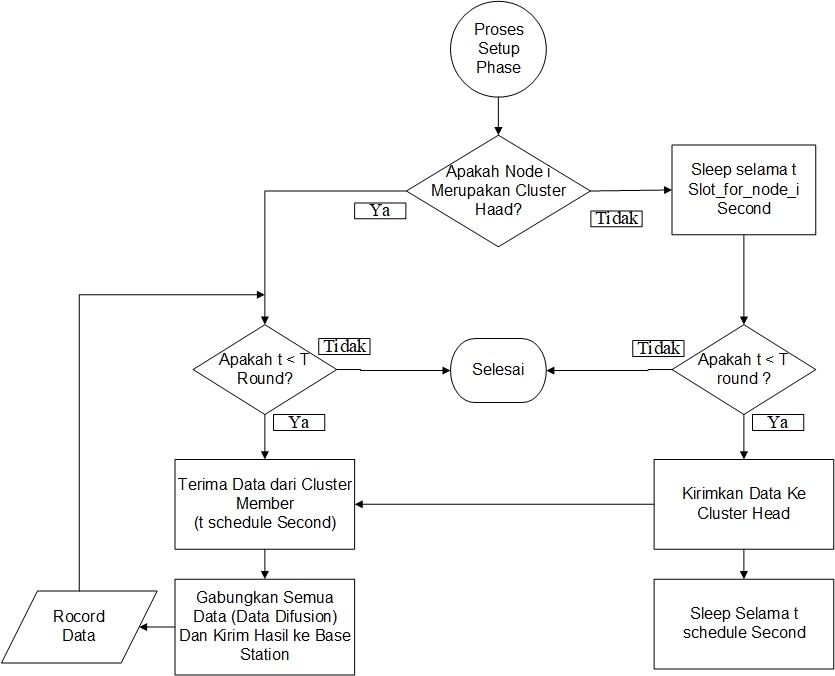
1. *Setup Phase*



Gambar 3.3 *Flowchart Setup Phase* LEACH Dengan *K-Means*

Penjelasan *flowchart*:

1. Tahap pertama adalah membentuk topologi jaringan berdasarkan beberapa parameter simulasi yang sudah ditentukan
2. Kedua masukkan jumlah klaster
3. Kemudian lakukan iterasi untuk penentuan *centroid*/titik pusat klaster sebanyak C=Jumlah klaster yang ditentukan
4. Letakkan titik pusat klaster/*centroid* C secara *random*
5. Hitung jarak setiap *node* dengan *centroid* C dengan persamaan (3*)*.
6. Kelompokan *node* pada titik pusat klaster/*centroid* C sesuai jarak terdekat
7. Kemudian cek apakah ada perubahan pusat klaster/*centroid* C
8. Jika iya maka tentukan titik pusat klaster/*centroid* C baru berdasarkan titik tengah pada k dengan persamaan (4), kemudian ulangi langkah 5-7 hingga tidak ada perubahan pusat klaster/*centroid* C
9. Jika tidak maka *node* i = titik pusat klaster/*centroid* C
10. Kemudian masuk pada penentuan *cluster head* dimana dalam pemilihan memerhatikan parameter residual energi pada *node* tiap klaster yang dihitung dengan persamaan (2)
11. Bandingkan apakah residual energi *nod*e i > *node* i+1 , jika iya maka pilih node i sebagai CH berdasarkan residual energi terbesar
12. Jika tidak maka *node* i = *node* i+1 dan ulangi langkah 9 hingga 11
13. Jika iya maka *node* i adalah *cluster head* dan lanjut ke proses *steady state*
14. Steady State

****

Gambar 3.4 Flowchart Steady State LEACH

(Sumber: Sohraby dkk, 2011. Wireless Sensor Network Technology, Protocols, Applications Book. New Jersey:John Wiley & Sons, Inc)

Penjelasan *Flowchart*:

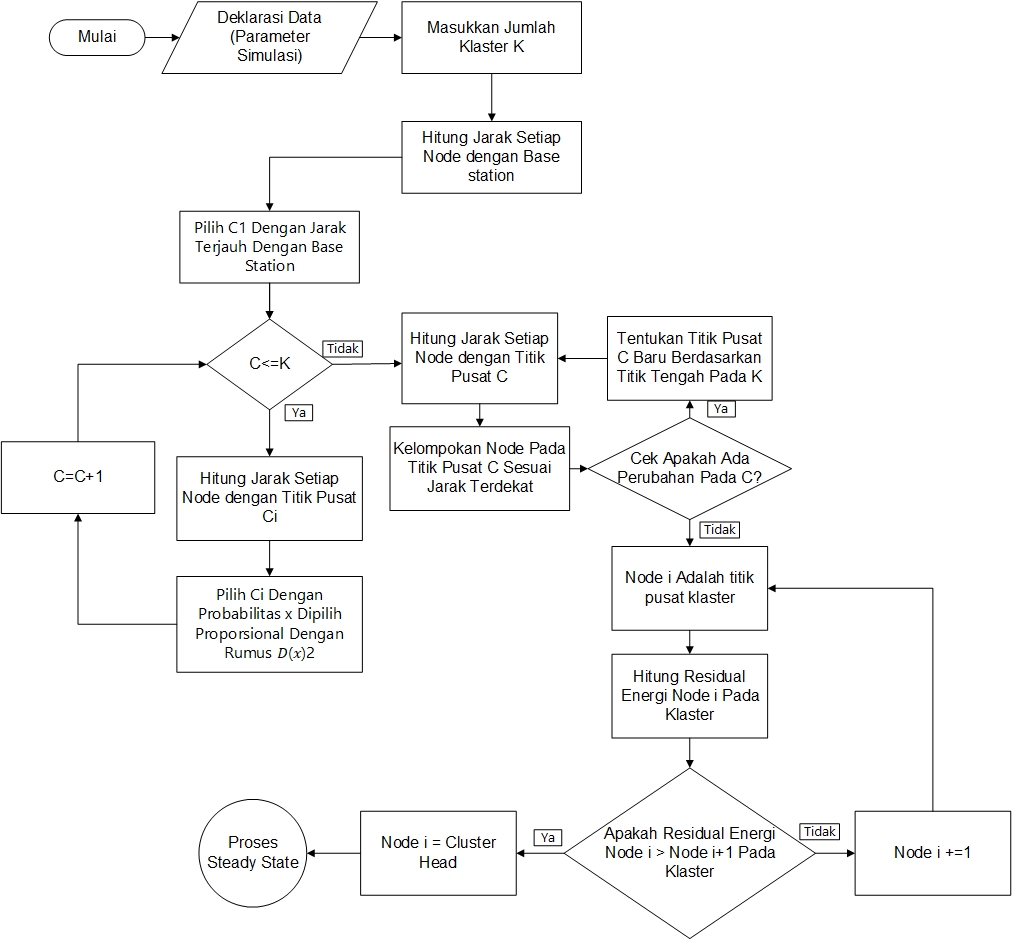
1. Cek apakah *node* merupakan sebuah *cluster head*
2. Apabila merupakan *cluster head* maka, *cluster head* akan menerima data dari semua anggota klaster.
3. *Node* yang bukan *cluster head* hanya akan mengirim data sesuai dengan jadwal TDMA yang sudah diterima.
4. *Node* *cluster head* akan mengumpulkan dan melakukan kompresi terhadap data
5. Kirim semua data yang sudah dikompresi ke *Base Station*
6. Proses *steady-state* selesai.

## **LEACH Dengan K-Means++**

Algoritma *K-Means++ Clustering* adalah salah satu algoritma yang digunakan untuk klasifikasi atau pengelompokan data, algoritma ini merupakan pengembangan dari algoritma *K-Means*. Pada protokol LEACH dengan bantuan algoritma *K-Means++* akan memodifikasi *setup phase* dari protokol LEACH dimana dalam *setup phase* pemilihan *cluster head* menggunakan algoritma *KMeans*++.

Namun dalam penelitian ini dalam pemilihan Centroid awal tidak dilakukan secara acak seperti algoritma *K-Means++,* namun pemilihan dilakukan berdasarkan 1 parameter tambahan yaitu jarak terjauh antara posisi *node* dengan *base station*. Untuk mencegah penentuan centroid awal yang dilakukan secara acak, hal ini dilakukan sesuai prinsip kerja algoritma *K-Means++* dalam penentuan centroid yaitu berdasarkan jarak *node* terjauh yang dimana jarak antara *node* dengan basestation dihitung dengan rumus *ecludien distance* pada persamaan (3). Untuk penjelasan lebih lanjut mengenai langkah-langkah yang dilakukan dapat dilihat pada *flowchart setup phase* dengan algoritma *K-Means++* yang direpresentasikan pada gambar 3.5:

1. Setup Phase



Gambar 3.5 Flowchart Setup Phase KM-LEACH

Penjelasan Flowchart:

1. Tahap pertama buat bentuk *node* sesuai dengan parameter yang ditentukan
2. Input jumlah klaster
3. Hitung jarak semua *node* dengan *base station* dengan rumus dari persamaan (3)
4. Tentukan pusat klaster C1 berdasarkan jarak terjauh antara *node* dengan *base station*.
5. Pemilihan Ci selanjutnya hitung jarak semua *node* dengan pusat klaster C1 dengan rumus dari persamaan (3)
6. Tentukan Ci dengan probabilitas x dipilih proporsional dengan persamaan (2)**.**
7. Ulangi langkah 4 dan 5 hingga semua klaster terbentuk
8. Hitung jarak setiap *node* dengan pusat klaster/*centroid* C dengan persamaan (3)
9. Kelompokan *node* pada titik pusat klaster/*centroid* C sesuai jarak terdekat
10. Kemudian cek apakah ada perubahan pusat klaster/*centroid* C
11. Jika iya maka tentukan titik pusat klaster/*centroid* C baru berdasarkan titik tengah pada k dengan persamaan (4), kemudian ulangi langkah 8-10 hingga tidak ada perubahan pusat klaster/*centroid* C
12. Jika tidak maka *node* i = titik pusat klaster/*centroid* C
13. Kemudian masuk pada penentuan *cluster head* dimana dalam pemilihan memerhatikan parameter residual energi pada *node* tiap klaster yang dihitung dengan persamaan (2).
14. Bandingkan apakah residual energi *nod*e i > *node* i+1
15. Jika tidak maka *node* i = *node* i+1 dan ulangi langkah 13 hingga 14
16. Jika iya maka *node* i adalah *cluster head* dan lanjut ke proses *steady state*

## **Implementasi**

Untuk mengimplementasikan penelitian ini, terdapat komponen-komponen pendukung yaitu:

1. Kebutuhan Perangkat Keras (*Hardware*)

Dari segi kebutuhan perangkat keras dibutuhkan perangkat elektronik dengan spesifikasi yaitu:

* Processor intel(R) Core(TM) i7-7700HQ CPU @2.80GHz
* Memory 8192MB RAM
* VGA Card Nvidia Geforce 950M
* Harddisk 1000 GB HDD

1. Kebutuhan Perangkat Lunak (Software)

Dari segi kebutuhan perangkat lunak (software), sistem ini akan diimplementasikan pada sistem operasi Windows 10. Aplikasi yang digunakan dalam penelitian ini adalah:

* MATLAB 2015 A

Serta aplikasi tambahan yang dibutuhkan seperti berikut:

* Google Chrome
* Nitro Pro

## **Skenario Pengujian**

Pada penelitian ini Protokol LEACH dengan optimasi *K-Means* dan KM-LEACH dengan optimasi *K-Means++* disimulasikan dengan software MATLAB 2015a dengan beberapa parameter yang disesuaikan berdasarkan hasil penelitian (Saheb & Sharma, 2017) untuk membuat skenario simulasi yang telah ditetapkan, diantaranya:

Table 2.4 Parameter Simulasi

(Sumber: Saheb & Sharma, 2017)

|  |  |
| --- | --- |
| Parameter | Nilai |
| Jumlah *node* | 100 |
| Luas area simulasi | 100 x 100 |
| Jumlah *round* (putaran) | 400 |
| Posisi *base station* | (50,50) |
| Probabilitas *cluster head %* | 10 |
| Energi awal *node* | 0.5 Joule |
| Data Length | 4000 bits |
| Eelec | 50nJ/bit |
| mp | 13nJ/bit/ |
|  | 10nJ/bit/ |
| Data Aggregation (EDA) | 50nJ/bit/signal |

Dengan parameter diatas, skenario akan terus dijalankan hingga round 400. Simulasi akan menghasilkan data berupa grafik untuk mengukur kinerja dari protokol LEACH dengan metode *K-Means* dan KM-LEACH dengan metode *K-Means++* dan pada masing-masing skenario, yang dimana nilai hasil simulasi yang dibahas lebih lanjut pada sub bab pengukuran dan hasil simulasi.

## **Pengukuran dan Hasil Simulasi**

Kualitas kinerja dari protokol KM-LEACH dengan protokol LEACH akan diukur dengan beberapa parameter seperti rata-rata konsumsi energi, banyaknya *death node* dan jumlah *node* yang hidup selama 100 round berlangsung. Dimana untuk menghitung total konsumsi energi yang dikeluarkan per-*bit* didapat dengan menggunakan konsumsi energi komunikasi radio yang sama yang diperlihatkan sebagai digunakan sebagai protokol LEACH. Model ini terdiri dari dua bagian: model konsumsi energi transmisi dan model konsumsi energi penerima. Sesuai dengan model disipasi energi komunikasi radio, didapatkan rumus untuk menghitung konsumsi energi dengan cara (Saheb & Sharma, 2017:

|  |  |
| --- | --- |
|  | (6) |

Dimana penjelas dari variabel rumus diatas yaitu:

= *Threshold Distance,*

**=** *Electronic Energy*

**=** *Parameter for Free Space Model*

**=** *Parameter for Multi Path Model*

Konsumsi energi untuk menerima pesan *k-bit* dapat dihitung dengan persamaan:

|  |  |
| --- | --- |
|  | (7) |

Sedangkan untuk mengetahui jumlah *node* mati dan *node* hidup dihitung dengan:

|  |  |
| --- | --- |
| ***Death node* = total *node - node* hidup** | (8) |
| ***Node* hidup = total *node – node* mati** | (9) |

Dimana setelah data didapatkan dari 2 skenario pada LEACH protokol dengan metode *K-Means* dan KM-LEACH protokol dengan metode *K-Means++,* kemudia data dibantingkan dengan tabel berikut untuk mendapatkan hasil akhir untuk mengetahui protokol yang lebih optimal diantara kedua protokol tersebut. Berikut adalah tabel parameter pengukuran kedua protokol:

Table 2.5 Parameter Uji Protokol

|  |  |  |  |
| --- | --- | --- | --- |
| Protokol | Jumlah *Death Node*/100 *Round* | Jumlah *Node* Hidup /100 *Round* | Total Konsumsi Energi/100 *Round* |
| LEACH dengan metode K*-Means* |  |  |  |
| KM-LEACH dengan metode *K-Means++* |  |  |  |

# **BAB IV HASIL DAN PEMBAHASAN**

## **Implementasi Sistem**

Terdapat beberapa proses dalam implementasi sistem pada penelitian ini. Proses-proses dalam implementasi sistem ini adalah sebagai berikut.

### **Deklarasi Variabel & Membentuk Topologi WSN**

Pada proses ini dilakukan pendeklarasian sejumlah variabel yang diperlukan dalam membangun simulasi ini. Selain itu pada tahap ini dibangun topologi dasar untuk proses simulasi yang akan dilakukan, dimana topologi ini berdasarkan parameter yang sudah ditentukan sebelumnya. Berikut merupakan *source code* setiap langkahnya terlihat pada Table 4.1 berikut.

Table 4.1 Deklarasi Variabel

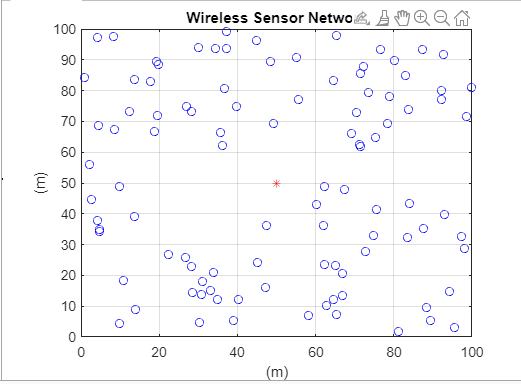
|  |
| --- |
| clc;  clear;  close all;  %%Deklarasi Variabel%%  %Pemetaan Sensor dalam satuan meter%  n=100;  xm=100;  ym=100;  sinkx=50;  sinky=50;  dead\_nodes=0;  operating\_nodes=n;  total\_energi=[];  total\_energi2=0;  total\_dn=[];  total\_na=[];  %Deklarasi Energi%  %Inisialisasi energi pada setiap node(Joules)%  Eo=0.5; %Satuan joules  Eelec=50\*10^(-8); %Satuan joules/bit  ETx=50\*10^(-8); %Satuan joules/bit  ERx=50\*10^(-8); %Satuan joules/bit  Eamp=13\*10^(-8); %Satuan joules/bit/m^2  EDA=5\*10^(-8); %Satuan joules/bit  kk=4000; %unit bits  rnd=1; %penanda iterasi  round=400; %jumlah putaran/iterasi yang ditentukan  k=10; %jumlah klaster yang ditentukan  %proses membaca titik node pada file  x=xlsread('node','A1:A100');  y=xlsread('node','B1:B100'); |

Pada tabel 4.1 diatas dilakukan beberapa pendeklarasian variabel seperti pemetaan sensor dalam satuan meter, variabel energi, penentuan klaster dan round dan proses pembacaan data titik koordinat node, dimana data node ini diambil dari file dengan nama node.xlsx yang sudah kita siapkan sebelumnya. Kemudian dilakukan proses pembentukan topologi jaringan awal yang source codenya dapat dilihat pada tabel berikut.

Table 4.2 Pembentukan Topologi WSN

|  |
| --- |
| %Membentuk Topologi Awal%  for i=1:n  SN(i).id=i; %Sensor Id  SN(i).x=x(i); %Posisi node x  SN(i).y=y(i); %Posisi node y  SN(i).E=Eo; %set energi node sama dengan eo"  SN(i).role=0; %set role node jika node biasa =0 dan jika  sebagai CH set =1  SN(i).cluster=0; %set node berada pada cluster mana,  secara default set 0  SN(i).cond=1; %Set status node jika node masih memiliki  energi maka nilai =1 jika node sudah mati  maka set =0  SN(i).rop=0; %untuk mengetahui node berada pada round  berapa  SN(i).dtch=0; %jarak node dengan CH  SN(i).dts=0; %jarak node dengan Sink/BS  SN(i).tel=0; %sudah berapa kali node di set sebagai CH  SN(i).rn=0; %round ke berapa node terpilih sebagai CH  SN(i).chid=0; %id node terhubung ke CH mana  SN(i).rleft=0;%rounds left for node to become available for  Cluster Head election  SN(i).re=Eo; %residual energi pada node  SN(i).jch=0; %mengetahui berapa kali node sebagai CH  SN(i).status=1; %set nilai jika node hidup status=1 jika  mati status=0  %Ploting Topologi  figure(1)  plot(xm,ym,x,y,'ob',sinkx,sinky,'\*r');  title 'Wireless Sensor Network';  xlabel '(m)';  ylabel '(m)';  hold on;  grid on;  end |

Sesuai pada tabel 4.2 diatas, setelah dilakukan deklarasi variabel dilakukan proses pembentukan topologi, nilai posisi node yang sudah dibaca dari file node.xlsx di lakukan proses *looping* dimana data disimpan pada variabel struk SN beserta memberi nilai dasar pada variabel-variabel di masing-masing node yang ada pada topologi. Setelah itu dilakukan proses ploting untuk menampilkan topologi yang sudah dirancang, dimana untuk posisi node dilambangkan dengan simbol bulat biru yang tersebar pada luas area 100 x 100 dan untuk posisi sink dilambangkan dengan symbol plus merah yang dapat dilihat pada Gambar 4.1 berikut.



Gambar 4.1 Topologi WSN

### **Setup Phase K-Means**

Setup phase merupakan proses pembentukan klaster dan proses pemilihan *cluster head* pada LEACH protokol. Pada proses ini berdasarkan data pada topologi WSN dilakukan proses pembentukan klaster dan pemilihan CH dengan algoritma *K-Means Clustering*. Pada proses setup phase ini terdiri dari beberapa tahapan diantaranya:

1. Penentuan Titik Pusat Klaster

Pada proses ini dengan menggunakan algoritma *K-Means* pemilihan titik pusat klaster dilakukan secara acak, untuk itu perlu dilakukan pembangkitan bilangan acak seseuai dengan jumlah klaster dan jumlah round yang ditentukan. Pada penelitian ini bilangan acak sudah kita siapkan sebanyak 10x400 sesuai dengan jumlah klaster x jumlah round yang kita simpan pada file bilacak.xlsx. Alasan data bilangan acak dibaca dari file karena dalam pembangkitan bilangan acak harus bilangan yang unik atau tidak sama untuk mencegah *infinity loop* karena titik node yang dipilih sama. Berikut merupakan *source code* tahapannya pada tabel 4.3 berikut.

Table 4.3 Proses Penentuan Titik Pusat Klaster Algoritma K-Means

|  |
| --- |
| %%% Set Up Phase %%%  %%Proses K-Means%%  CLheads=0;  energy=0;  while rnd<=round  disp('Proses Kmeans')  rnd %Menampilkan nilai round saat ini  %Proses memilih cluster secara acak  startx=[];  starty=[];  center=[];  acak=xlsread('bilacak');  for i=1:k  startx(i)=x(acak(i,rnd));  starty(i)=y(acak(i,rnd));  center(i,1)=startx(i);  center(i,2)=starty(i);  end  %Ploting sebaran titik pusat klaster pada round terakhir  if(rnd==round)  figure(2)  disp(center);  plot(center(:,1),center(:,2),'ko','linewidth',3);  hold on;  grid on;  for i=1:n plot(xm,ym,SN(i).x,SN(i).y,'ob',sinkx,sinky,'\*r');  title 'Wireless Sensor Network';  xlabel '(m)';  ylabel '(m)';  end  end |

1. Pengelompokan Node Dengan Klaster

Pada proses ini setelah titik pusat klaster ditentukan selanjutnya adalah kita simpan pada variabel center. Kemudian kita siapkan variabel pendukung lainnya untuk proses ini kemudia lakukan perulangan while hingga titik pusat klaster tidak mengalami perubahan atau temp=1. Dalam proses perulangan hitung jarak setiap node dengan titik pusat klaster dengan menggunakan rumus *ecludien distance* sesuai dengan persamaan (3) yang kita simpan dalam variabel data diamana nilai data yang terendah kita simpan nilai jarak dan index node tersebut.

Kemudian data node pada klaster yang sama kita simpan dalam variabel outpux dan outputy. Setelah semua node dikelompokkan titik pusat klaster diperbaharui dengan mencari rata-rata tiap klaster sesuai dengan persamaan (4) atau pada matlab menggunakan function *Mean* dimana nilainya kita simpan ke variabel cx dan cy Kembali. Berikut merupakan *source code* tahapannya pada tabel 4.4 berikut.

Table 4.4 Proses Pengelompokan Klaster Algoritma K-means

|  |
| --- |
| %Proses pengelompokan node dengan cluster  cx=center(:,1); %Menyimpan nilai titik pusat x pada var cx  cy=center(:,2); %Menyimpan nilai titik pusat y pada var cy  mean\_oldx=cx; %menyimpan nilai x pusat klaster lama  mean\_newx=cx; %menyimpan nilai x pusat klaster yang baru  mean\_oldy=cy; %menyimpan nilai y pusat klaster lama  mean\_newy=cy; %menyimpan nilai y pusat klaster yang baru  outputx=cell(k,1); %menyimpan data node x sesuai klaster  outputy=cell(k,1); %menyimpan data node y sesuai klaster  temp=0;  iter=1;  %Perulangan pembentukan klaster hingga titik pusat tidak  mengalami perubahan  while(temp==0)  %menyamakan nilai pusat klaster lama dan baru  mean\_oldx=mean\_newx;  mean\_oldy=mean\_newy;  %perulangan mencari jarak setiap node  for i=1:n  data=[];  %mencari jarak setiap node dengan titik pusat  klaster  for j=1:length(cx)  data=[data sqrt((SN(i).x-cx(j))^2+(SN(i).y-cy(j))^2)];  end  %Mencari jarak terendah node dengan titik pusat klaster  [gc index]=min(data);  %Mengelompokkan data sesuai CH terdekat  outputx{index}=[outputx{index} SN(i).x];  outputy{index}=[outputy{index} SN(i).y];  SN(i).role=0; %reseting node role  SN(i).cluster=index; %Menyimpan nilai cluster  tiap node  SN(i).chid=index;  SN(i).dtch=min(data);  end  gmckx=[];  gmcky=[];  %Proses memperbarui nilai pusat CH  for i=1:k  gmckx=[gmckx mean(outputx{i})];  gmcky=[gmcky mean(outputy{i})];  end  cx=gmckx; %Menyimpan nilai terbaru CH  cy=gmcky;  mean\_newx=cx; %Menyimpan nilai terbaru CH  mean\_newy=cy; |

1. Pengecekan Titik Pusat Klaster

Setelah dicari nilai mean masing-masing klaster, simpan nilai pada variabel cx dan mean\_newx untuk titik nilai x dan xy dan mean\_newy untuk variabel y. Kemudian untuk melakukan pengecekan apakah iterasi dilanjutkan atau di hentikan disini dengan mengecek nilai mean\_newx == mean\_oldx dan apakah nilai mean\_newy == mean\_oldy. Dimana jika keduanya mengalami tidak mengalami perubahan kita set finalx dan finaly =1, kedua variabel ini kita cek kembali untuk memastikan apakah kedua titik pusat klaster x dan y tidak mengalami perubahan atau iya, jika tidak maka kita set nilai temp=1 untuk menghentikan perulangan while.

Apabila iya maka kita simpan kembali nilai titik x dan y ke variabel output dan output dimana variabel ini bertipe cell sesuai jumlah klaster yang ada, dan proses ini dilakukan secara berulang hingga titik pusat klaster tidak mengalami perubahan. Berikut merupakan *source code* tahapannya pada tabel 4.5 berikut.

Table 4.5 Pengecekan Titik Pusat Klaster Algoritma K-Means

|  |
| --- |
| finalx=0;  finaly=0;  %Mengecek apakah titik pusat klaster mengalami  perubahan atau tidak  if(mean\_newx==mean\_oldx)  finalx=1;  end  if(mean\_newy==mean\_oldy)  finaly=1;  end  Jika tidak mengalami perubahan maka akhiri  perulangan  if(finalx==1 && finaly==1) %  temp=1;  %Jika tidak sama maka simpan nilai pusat  klaster baru pada bvariabel outputx dan outputy  else  outputx=cell(k,1);  outputy=cell(k,1);  end  iter=iter+1;  end  jml\_iter(rnd)=iter;  celldisp(outputx);  celldisp(outputy); |

1. Proses Pemilihan Cluster Head

Setelah klaster sudah terbentuk tahap selanjutnya adalah pemilihan *cluster head.* Pada pemilihan CH ini dipilih node dengan residual energi tertinggi pada masing-masing klaster. Untuk menghitung residual energi pada masing-masing node menggunakan persamaan (2). Dalam pemilihan CH setiap node akan memiliki kondisi dimana node dapat terpilih kembali menjadi CH untuk memastikan setiap node pernah terpilih sebagai CH pada masing-masing klaster sesuai konsep LEACH protokol.

Apabila sebuah node terpilih menjadi CH maka akan di set role= 1 dan menghitung jarak setiap CH dengan sink/base station yang disimpan di variabel SN.dts untuk proses perhitungan energi nanti di fase *steady stat*e. Setelah CH setiap klaster terpilih, selanjutnya kita perbarui kembali jarak setiap node pada CH untuk memperbarui nilai jarak masing-masing node dengan CH yang disimpan di variabel SN.dtch. Setelah itu maka proses *setup phase* dengan algorima K-Means selesai dilakukan dan lanjut ke fase *steady state*. Berikut adalah source code proses pemilihan CH pada tabel 4.6 berikut.

Table 4.6 Pemilihan CH Algoritma K-Means

|  |
| --- |
| %Pemilihan CH  CLheads=0;  energy=0;  %Pemilihan CH tiap cluster  for i=1:k  for j=1:n  %memastikan node yg dipilih berada pada  klaster yang sama dan node bukan CH  if(SN(j).cluster==I && SN(j).role==0)  %Mengurangi probabilitas node yang sudah  pernah terpilih sebagai CH  if(SN(j).rleft>0)  SN(j).rleft=SN(j).rleft-1;  End  %Memastikan energi node >0 node berada di  kodisi dapat dipilih dan node bukan CH  if(SN(j).E>0 && SN(j).rleft==0 && SN(j).role==0)  for l=1:n  %jika residual energi node lebih besar dari node lain pada klaster yang sama  if(SN(j).re>=SN(l).re && SN(j).cluster==SN(l).cluster)  SN(j).role=1;  SN(j).jch=SN(j).jch+1;  SN(j).tel=SN(j).tel + 1;  SN(j).rleft=k/2;  SN(j).dts=sqrt((sinkx-SN(j).x)^2 + (sinky-SN(j).y)^2);  CLheads=CLheads+1;  SN(j).cluster=CLheads;  CL(CLheads).x=SN(j).x;  CL(CLheads).y=SN(j).y;  CL(CLheads).id=i;  break  end  end  %Untuk mengakhiri perulangan dan lanjut ke perulangan k selanjutnya jika sudah didapatkan CH tiap klaster  if(SN(j).role==1)  break  end  end  end  end  end  CL=CL(1:CLheads);  %Menghitung jarak node dengan masing" CH  for i=1:n  if(SN(i).role==0 && SN(i).E>0)  for j=1:CLheads  if(SN(i).cluster==j)  SN(i).dtch=sqrt((CL(j).x-SN(i).x)^2 + (CL(j).y-SN(i).y)^2);  end  end  else  SN(i).dtch=0;  end  end |

### **Setup Phase K-Means++**

Setup phase merupakan proses pembentukan klaster dan proses pemilihan *cluster head* pada LEACH protokol. Pada proses ini berdasarkan data pada topologi WSN dilakukan proses pembentukan klaster dan pemilihan CH dengan algoritma *K-Means++ Clustering*. Yang membedakan fase setup phase K-Means dan K-Means++ terletak pada bagian penentuan titik pusat klasternya diamana pada algoritma K-Means++ ini pemilihan titik pusat klaster tidak dilakukan secara acak namun menggunakan proses perhitungan sesuai dengan persamaan (5). Berikut adalah tahapannya.

1. Penentuan Titik Pusat Klaster

Pada proses ini dengan menggunakan algoritma *K-Means++* dimana pemilihan titik pusat klaster dilakukan tidak secara acak, namun berdasarkan rumus perhitungan. Dalam proses penentuan titik pusat klaster pertama berbeda dengan penentuan titik pusan klaster selanjutnya, yaitu pada pemilihan titik pusat pertama node yang dipilih sebagai titik pusat klaster berdasarkan jarak terjauh node dengan *sink/base station* menggunakan rumus persamaan (3). Setelah didapatkan jarak terjauh titik pusat klaster pertama atau C1 disimpan di variabel center. Berikut merupakan source code tahapannya pada tabel 4.7 berikut.

Table 4.7 Penentuan Titik Pusat Klaster Algoritma K-Means++

|  |
| --- |
| %%% Set Up Phase %%%  %%Proses K-Means++%%  CLheads=0;  energy=0;  while rnd<=round  disp('Proses Kmeans++')  rnd  %Penentua C1 Awal dengan mencari jarak terjauh  dengan BS  startx=[];  starty=[];  x=xlsread('node','A1:A100');  y=xlsread('node','B1:B100');  center=[startx starty]; %menyimpan nilai C1  d=[];  for i=1:n  SN(i).cluster=0;  SN(i).role=0; %reseting node role  SN(i).chid=0;  c=[SN(i).x SN(i).y]; %mengambil data node ke-i  %menghitung jarak C1 dengan setiap node  distance=sqrt((sinkx-c(:,1))^2+(sinky-c(:,2))^2);  d=[d distance]; %menyimpan index dan nilai  jarak masing" node dengan BS  end  [e s]=max(d); %mencari nilai terbesar dari distance  yang dipilih sebagai C1  center=[center;[x(s) y(s)]]; %menyimpan koordinat c1  dalam array  x(s)=[]; %mengosongkan kembali nilai x dan y  y(s)=[];  d=[];  %Proses mencari C selanjutnya  r=1;  while(r~=k)  for i=1:n  %Mengambil data kordinat node ke i  g=[SN(i).x SN(i).y];  %Mencari node terdekat dengan data pusat  klaster  ka=dsearchn(center,g);  %Mencari data terdekat pada nilai kordinat x  nearestx=center(ka,1);  %Mencari data terdekat pada nilai kordinat y  nearesty=center(ka,2);  %Menghitung jarak node dengan titik pusat  klaster  distance=sqrt((nearestx-g(:,1))^2+(nearesty-g(:,2))^2);  %Menyimpan nilai jarak dan indeksnya  d=[d distance];  end  %Mencari data jarak dengan nilai max/tertinggi  [e s]=max(d);  %Menyimpan data node dengan jarak tertinggi  pada aray CH  center=[center;[x(s) y(s)]];  SN(s).tel=SN(s).tel+1;  x(s)=[]; %mengosongkan kembali nilai x, y dan d  y(s)=[];  [r c]=size(center);  d=[];  end |

Kemudian dalam proses pencarian titik pusat klaster atau C selanjutnya, hitung jarak setiap node dengan titik pusat klaster yang ada di variabel center, kemudian pilih node sebagai titik pusat klaster selanjutnya berdasarkan probabilitas x dipilih proporsional dengan rumus persamaan (5), yaitu pilih nilai terjauh dari setiap titik pusat klaster sebagai C selanjutnya. Ulangi langkah tersebut hingga jumlah titik pusat klaster sesuai dengan jumlah klaster yang ditentukan maka proses ini selesai.

1. Pengelompokan Node Dengan Klaster

Pada proses ini setelah titik pusat klaster ditentukan selanjutnya adalah kita simpan pada variabel center. Kemudian kita siapkan variabel pendukung lainnya untuk proses ini kemudia lakukan perulangan while hingga titik pusat klaster tidak mengalami perubahan atau temp=1. Dalam proses perulangan hitung jarak setiap node dengan titik pusat klaster dengan menggunakan rumus ecludien distance sesuai dengan persamaan (3) yang kita simpan dalam variabel data diamana nilai data yang terendah kita simpan nilai jarak dan index node tersebut.

Kemudian data node pada klaster yang sama kita simpan dalam variabel outpux dan outputy. Setelah semua node dikelompokkan titik pusat klaster diperbaharui dengan mencari rata-rata tiap klaster sesuai dengan persamaan (4) atau pada matlab menggunakan function Mean dimana nilainya kita simpan ke variabel cx dan cy Kembali. Berikut merupakan source code tahapannya pada tabel 4.8 berikut.

Table 4.8 Proses Pengelompokan Klaster Algoritma K-Means++

|  |
| --- |
| %Proses pengelompokan node dengan cluster  cx=center(:,1); %Menyimpan nilai titik pusat x pada var cx  cy=center(:,2); %Menyimpan nilai titik pusat y pada var cy  mean\_oldx=cx; %menyimpan nilai x pusat klaster lama  mean\_newx=cx; %menyimpan nilai x pusat klaster yang baru  mean\_oldy=cy; %menyimpan nilai y pusat klaster lama  mean\_newy=cy; %menyimpan nilai y pusat klaster yang baru  outputx=cell(k,1); %menyimpan data node x sesuai klaster  outputy=cell(k,1); %menyimpan data node y sesuai klaster  temp=0;  iter=1;  %Perulangan pembentukan klaster hingga titik pusat tidak  mengalami perubahan  while(temp==0)  %menyamakan nilai pusat klaster lama dan baru  mean\_oldx=mean\_newx;  mean\_oldy=mean\_newy;  %perulangan mencari jarak setiap node  for i=1:n  data=[];  %mencari jarak setiap node dengan titik pusat  klaster  for j=1:length(cx)  data=[data sqrt((SN(i).x-cx(j))^2+(SN(i).y-cy(j))^2)];  end  %Mencari jarak terendah node dengan titik pusat klaster  [gc index]=min(data);  %Mengelompokkan data sesuai CH terdekat  outputx{index}=[outputx{index} SN(i).x];  outputy{index}=[outputy{index} SN(i).y];  SN(i).role=0; %reseting node role  SN(i).cluster=index; %Menyimpan nilai cluster  tiap node  SN(i).chid=index;  SN(i).dtch=min(data);  end  gmckx=[];  gmcky=[];  %Proses memperbarui nilai pusat CH  for i=1:k  gmckx=[gmckx mean(outputx{i})];  gmcky=[gmcky mean(outputy{i})];  end  cx=gmckx; %Menyimpan nilai terbaru CH  cy=gmcky;  mean\_newx=cx; %Menyimpan nilai terbaru CH  mean\_newy=cy; |

1. Pengecekan Titik Pusat Klaster

Setelah dicari nilai mean masing-masing klaster, simpan nilai pada variabel cx dan mean\_newx untuk titik nilai x dan xy dan mean\_newy untuk variabel y. Kemudian untuk melakukan pengecekan apakah iterasi dilanjutkan atau di hentikan disini dengan mengecek nilai mean\_newx == mean\_oldx dan apakah nilai mean\_newy == mean\_oldy. Dimana jika keduanya mengalami tidak mengalami perubahan kita set finalx dan finaly =1, kedua variabel ini kita cek kembali untuk memastikan apakah kedua titik pusat klaster x dan y tidak mengalami perubahan atau iya, jika tidak maka kita set nilai temp=1 untuk menghentikan perulangan while.

Apabila iya maka kita simpan kembali nilai titik x dan y ke variabel output dan output dimana variabel ini bertipe cell sesuai jumlah klaster yang ada, dan proses ini dilakukan secara berulang hingga titik pusat klaster tidak mengalami perubahan. Berikut merupakan source code tahapannya pada tabel 4.9 berikut.

Table 4.9 Pengecekan Titik Pusat Klaster Algoritma K-Means++

|  |
| --- |
| finalx=0;  finaly=0;  %Mengecek apakah titik pusat klaster mengalami  perubahan atau tidak  if(mean\_newx==mean\_oldx)  finalx=1;  end  if(mean\_newy==mean\_oldy)  finaly=1;  end  Jika tidak mengalami perubahan maka akhiri  perulangan  if(finalx==1 && finaly==1) %  temp=1;  %Jika tidak sama maka simpan nilai pusat  klaster baru pada bvariabel outputx dan outputy  else  outputx=cell(k,1);  outputy=cell(k,1);  end  iter=iter+1;  end  jml\_iter(rnd)=iter;  celldisp(outputx);  celldisp(outputy); |

1. Proses Pemilihan CH

Setelah klaster sudah terbentuk tahap selanjutnya adalah pemilihan cluster head. Pada pemilihan CH ini dipilih node dengan residual energi tertinggi pada masing-masing klaster. Untuk menghitung residual energi pada masing-masing node menggunakan persamaan (2). Dalam pemilihan CH setiap node akan memiliki kondisi dimana node dapat terpilih kembali menjadi CH untuk memastikan setiap node pernah terpilih sebagai CH pada masing-masing klaster sesuai konsep LEACH protokol.

Apabila sebuah node terpilih menjadi CH maka akan di set role=1 dan menghitung jarak setiap CH dengan sink/base station yang disimpan di variabel SN.dts untuk proses perhitungan energi nanti di fase steady state. Setelah CH setiap klaster terpilih, selanjutnya kita perbarui kembali jarak setiap node pada CH untuk memperbarui nilai jarak masing-masing node dengan CH yang disimpan di variabel SN.dtch. Setelah itu maka proses setup phase dengan algorima *K-Means++* selesai dilakukan dan lanjut ke fase *steady state*. Berikut adalah *source code* proses pemilihan CH pada tabel 4.10 berikut.

Table 4.10 Pemilihan CH algoritma K-Means++

|  |
| --- |
| %Pemilihan CH  CLheads=0;  energy=0;  %Pemilihan CH tiap cluster  for i=1:k  for j=1:n  %memastikan node yg dipilih berada pada  klaster yang sama dan node bukan CH  if(SN(j).cluster==I && SN(j).role==0)  %Mengurangi probabilitas node yang sudah  pernah terpilih sebagai CH  if(SN(j).rleft>0)  SN(j).rleft=SN(j).rleft-1;  End  %Memastikan energi node >0 node berada di  kodisi dapat dipilih dan node bukan CH  if(SN(j).E>0 && SN(j).rleft==0 && SN(j).role==0)  for l=1:n  %jika residual energi node lebih besar dari node lain pada klaster yang sama  if(SN(j).re>=SN(l).re && SN(j).cluster==SN(l).cluster)  SN(j).role=1;  SN(j).jch=SN(j).jch+1;  SN(j).tel=SN(j).tel + 1;  SN(j).rleft=k/2;  SN(j).dts=sqrt((sinkx-SN(j).x)^2 + (sinky-SN(j).y)^2);  CLheads=CLheads+1;  SN(j).cluster=CLheads;  CL(CLheads).x=SN(j).x;  CL(CLheads).y=SN(j).y;  CL(CLheads).id=i;  break  end  end  %Untuk mengakhiri perulangan dan lanjut ke perulangan k selanjutnya jika sudah didapatkan CH tiap klaster  if(SN(j).role==1)  break  end  end  end  end  end  CL=CL(1:CLheads);  %Menghitung jarak node dengan masing" CH  for i=1:n  if(SN(i).role==0 && SN(i).E>0)  for j=1:CLheads  if(SN(i).cluster==j)  SN(i).dtch=sqrt((CL(j).x-SN(i).x)^2 + (CL(j).y-SN(i).y)^2);  end  end  else  SN(i).dtch=0;  end  end |

### **Steady State Phase**

Steady state merupakan proses transmission energi dari *node* menuju *sink/base station*, dimana proses ini terjadi setiap roundnya saat node mengirimkan informasi menuju *sink/base station* dalam sebuah topologi WSN. Dalam proses transmission energinya, fase steady state dibagi menjadi 2 tahapan yaitu sebagai berikut.

1. Transmission Energi Pada Node Biasa

Pada proses ini merupakan proses bagaimana perhitungan energi yang dihabiskan setiap node yang ada dalam jaringan setiap proses transmission terjadi. Setelah klaster terbentuk dan masuk ke tahap *steady state* setiap *node* akan dihitung total energi yang dikeluarkan dengan menggunakan persamaan (6) yang disimpan di variabel Etx. Nilai dari variabel etx ini akan dikurangi dengan jumlah energi awal yang dimiliki setiap *node*. Variabel energi digunakan untuk menyimpan nilai total konsumsi energi setiap node yang ada setiap roundnya.

Kemudian ketika node mengirimkan data menuju CH, maka CH yang menerima data juga dihitung energi yang dikeluarkan saat menerima data dengan menggunakan persamaan (7) yang disimpan di variabel Erx. Kemudian energi yang ada pada CH akan dikurangi dengan nilai pada variabel Erx yang dimana proses ini dilakukan secara terus-menerus saat CH menerima data yang dikirimkan oleh node.

Kemudian dalam proses ini juga dilakukan pengecekan apakah dalam proses *node* mengirimkan energi dan saat CH menerima data energi yang dimiliki masih tersisa atau sama dengan 0. Jika energi yang dimiliki habis maka variabel status dan cond pada *node* di set 0 untuk menandakan bahwa *node* sudah dalam keadaan mati dan tidak bisa mengikuti proses transmisi energi. Saat *node* dalam keadaan mati maka nilai jumlah dead\_node ditambahkan 1 dan operating node dikurangi 1 setiap ditemukan kondisi *node* mati. Berikut adalah *source code* proses perhitungan energi pada node biasa pada tabel 4.11 berikut.

Table 4.11 Perhitungan Energi Pada Node Biasa

|  |
| --- |
| %%%Steady State Phase%%%  %Perhitungan energi pada node biasa  for i=1:n  if (SN(i).cond==1 && SN(i).role==0)  if (SN(i).E>0)  ETx= Eelec\*k + Eamp \* k \* SN(i).dtch^2;  SN(i).E=SN(i).E - ETx;  SN(i).re=SN(i).E;  energy=energy+ETx;  end  %Perhitungan energi saat CH menerima data yg  dikirim node  if SN(SN(i).chid).E>0 && SN(SN(i).chid).cond==1 && SN(SN(i).chid).role==1  ERx=(Eelec+EDA)\*k;  energy=energy+ERx;  SN(SN(i).chid).E=SN(SN(i).chid).E - ERx;  %Jika energi CH habis saat menerima data yang  dikirim node  if (SN(SN(i).chid).E<=0 && SN(SN(i).chid).status==1)  SN(SN(i).chid).cond=0;  SN(SN(i).chid).rop=rnd;  SN(i).E=0;  dead\_nodes=dead\_nodes +1;  operating\_nodes= operating\_nodes - 1;  SN(i).status=0;  end  end  end  %Jika energi node habis saat proses transmisi  if (SN(i).E<=0 && SN(i).status==1)  dead\_nodes=dead\_nodes +1;  operating\_nodes= operating\_nodes - 1;  SN(i).cond=0;  SN(i).chid=0;  SN(i).rop=rnd;  SN(i).E=0;  SN(i).status=0;  end  end |

1. Transmiision Energi Pada Node Sebagai CH

Pada proses ini tidak jauh berbeda dengan proses perhitungan energi pada node, yaitu yang membedakan adalah hanya jarak dimana jarak yang digunakan yaitu jarak CH dengan *sink/base station* yang nilainya berada di variabel SN.dts. Dimana akan dicek apakah SN.role==1 yaitu node sebagai CH maka akan masuk ke proses ini.

Dalam proses ini juga dicek kondisi CH apakah energi yang ada pada CH masih tersisa atau tidak, jika energi CH <=0 maka variabel status dan cond di set menjadi 0 untuk menandakan bahwa CH tersebut dalam kondisi mati. Jika CH berada dalam kondisi mati maka variabel death\_node ditambahkan 1 dan operating\_nodes dikurangi 1. Berikut adalah *source code* proses perhitungan energi pada node biasa pada tabel 4.12 berikut.

Table 4.12 Perhitungan Energi Node Sebagai CH

|  |
| --- |
| %Perhitungan energi pada Cluster Head  for i=1:n  if (SN(i).cond==1 && SN(i).role==1)  if (SN(i).E>0)  ETx= (Eelec+EDA)\*k + Eamp \* k \* SN(i).dts^2;  SN(i).E=SN(i).E - ETx;  SN(i).re=SN(i).E;  energy=energy+ETx;  end  %Jika energi CH habis saat proses transmisi  if (SN(i).E<=0 && SN(i).status==1)  dead\_nodes=dead\_nodes +1;  operating\_nodes= operating\_nodes - 1;  SN(i).cond=0;  SN(i).rop=rnd;  SN(i).E=0;  SN(i).status=0;  end  end  end |

1. Proses Perhitungan Total Energi, Node Mati & Node Hidup

Pada bagian akhir setelah proses perhitungan energi, node mati dan node hidup dilakukan pada masing-masing node dan CH maka nilai masing-masing disimpan dalam variabel array dengan indeks sesuai rnd atau putaran yang berlangsung. Dimana nantinya aray yang terbentuk sebanyak 400 baris sesuai dengan jumlah putaran yang kita definisikan diawal yaitu 400 round. Kemudian kita tambah nilai dari rnd untuk melanjutkan proses perulangan untuk mencapai 400 round.

Kemudian setelah perulangan round selesai, data disimpan di sebuah file bertipe txt sebagai proses dokumentasi data penelitian yang ingin didapat dalam proses simulasi ini. File yang dibuat terdiri dari 3 file yaitu total\_energi.txt yang menyimpan data total konsumsi energi node setiap roundnya, total\_dn.txt yang menimpan data total node mati setiap round dan file total\_na.txt yang menyimpan data total node hidup setiap round. Berikut adalah *source code* proses perhitungan total energi, node mati & node hidup pada tabel 4.13 berikut.

Table 4.13 Perhitungan Total Energi, Node Mati & Node Hidup

|  |
| --- |
| %Kalkulasi total energi, death node & node alive  total\_energi(rnd)=total\_energi2+energy;  total\_energi2=total\_energi(rnd);  total\_dn(rnd)=dead\_nodes;  total\_na(rnd)=operating\_nodes;    if(dead\_nodes>=n)  break  end  % Next Round %  rnd=rnd+1;  end  % Save data dalam bentuk file  writematrix(total\_energi,'TE-kmeansplus');  writematrix(total\_dn,'TDN-kmeansplus');  writematrix(total\_na,'TNA-kmeansplus'); |

## **Hasil dan Pengujian Sistem**

Pada subbab ini akan dijelaskan mengenai hasil dari pengujian simulasi yang dilakukan dalam penelitian ini. Hasil dalam penelitian ini yaitu plotting data perbandingan total konsumsi energi, jumlah node mati dan node hidup dari kedua metode yang dilakukan dalam proses simulasi dalam penelitian ini.

### **Hasil Clustering Metode K-Means**

Pada subbab ini menampilkan hasil proses *setup phase* yang dilakukan dengan menggunakan algoritma *K-Means Clustering.* Saat proses s*setup phase* dilangsungkan setiap roundnya akan menghasilkan 2 plotting data yaitu sebagai berikut.

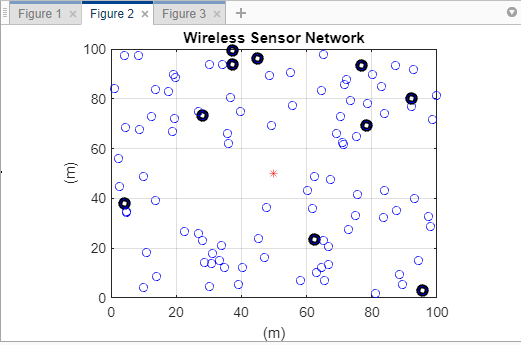
1. Ploting Titik Pusat Klaster Algoritma K-Means

Pada proses ini setelah titik pusat klaster dipilih secara acak dengan membaca data bilangan acak dari file bilacak.xlsx, maka data titik pusat klaster disimpan dalam variabel center. Dalam proses plotting data ini menampilkan salah satu cuplikan data yang ada, dimana disini manampilkan data round terakhir sebagai contoh. Berikut adalah *source code* proses plotting data titik pusat klaster algoritma *K-Means* pada tabel 4.14 berikut.

Table 4. 14 Data Titik Pusat Klaster Algoritma K-Means

|  |
| --- |
| %Plotting data round terakhir  if(rnd==round)  figure(2)  disp(center);  plot(center(:,1),center(:,2),'ko','linewidth',3);  hold on;  grid on;  for i=1:n  plot(xm,ym,SN(i).x,SN(i).y,'ob',sinkx,sinky,'\*r');  title 'Wireless Sensor Network';  xlabel '(m)';  ylabel '(m)';  end  end |

Setelah source code diatas dijalankan maka akan menampilkan hasil plotting seperti gambar 4.2 berikut.



Gambar 4.2 Plotting Titik Pusat Klaster Algoritma K-Means

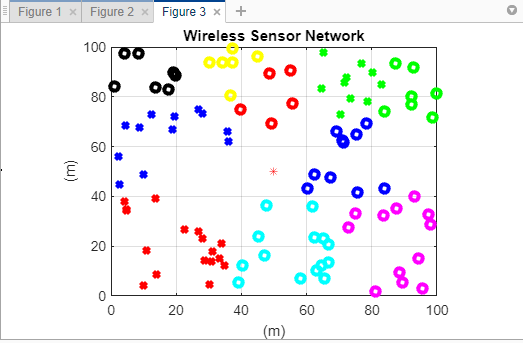
1. Ploting Hasil *Clustering* Algoritma *K-Means*

Pada proses ini setelah titik pusat klaster dipilih dan masing-masing node dikelompokkan hingga titik pusat klaster tidak mengalami perubahan maka akan didapatkan hasil akhir klaster node yang dibuat sesuai dengan jumlah klaster yang ditentukan. Data node tiap klasternya disimpan dalam variabel outputx dan output. Berikut adalah *source code* proses plotting data hasil *clustering* dengan algoritma *K-Means* pada tabel 4.15 berikut.

Table 4.15 Ploting Hasil Clustering Algoritma K-Means

|  |
| --- |
| %Menampilkan plotting data round terakhir  if(rnd==round)  figure;  for i=1:k  xf=outputx{i};  yf=outputy{i};  if(i==1)  plot(xf,yf,'bo','linewidth',3);  elseif(i==2)  plot(xf,yf,'go','linewidth',3);  elseif(i==3)  plot(xf,yf,'ro','linewidth',3);  elseif(i==4)  plot(xf,yf,'co','linewidth',3);  elseif(i==5)  plot(xf,yf,'mo','linewidth',3);  elseif(i==6)  plot(xf,yf,'yo','linewidth',3);  elseif(i==7)  plot(xf,yf,'ko','linewidth',3);  elseif(i==8)  plot(xf,yf,'rx','linewidth',3);  elseif(i==9)  plot(xf,yf,'bx','linewidth',3);  else  plot(xf,yf,'gx','linewidth',3);  end  hold on;  grid on;  end  figure(3)  plot(xm,ym,sinkx,sinky,'\*r');  title 'Wireless Sensor Network';  xlabel '(m)';  ylabel '(m)';  hold on;  end |

Setelah source code diatas dijalankan maka akan menampilkan hasil plotting seperti gambar 4.3 berikut



Gambar 4.3 Plotting Hasil Clustering Dengan Algoritma K-Means

### **Hasil Clustering Metode K-Means++**

Pada subbab ini menampilkan hasil proses *setup phase* yang dilakukan dengan menggunakan algoritma *K-Means++ Clustering.* Saat proses s*setup phase* dilangsungkan setiap roundnya akan menghasilkan 2 plotting data yaitu sebagai berikut.

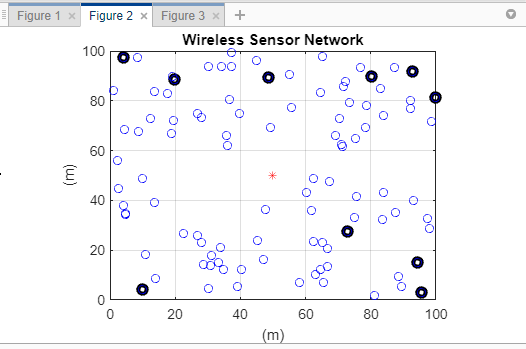
1. Ploting Titik Pusat Klaster Algoritma *K-Means++*

Pada proses ini setelah titik pusat klaster dipilih, maka data titik pusat klaster disimpan dalam variabel center. Dalam proses plotting data ini menampilkan salah satu cuplikan data yang ada, dimana disini manampilkan data round terakhir sebagai contoh. Berikut adalah *source code* proses plotting data titik pusat klaster algoritma *K-Means++* pada tabel 4.16 berikut

Table 4.16 Data Titik Pusat Klaster Algoritma K-Means++

|  |
| --- |
| %Plotting data round terakhir  if(rnd==round)  figure(2)  disp(center);  plot(center(:,1),center(:,2),'ko','linewidth',3);  hold on;  grid on;  for i=1:n  plot(xm,ym,SN(i).x,SN(i).y,'ob',sinkx,sinky,'\*r');  title 'Wireless Sensor Network';  xlabel '(m)';  ylabel '(m)';  end  end |

Setelah source code diatas dijalankan maka akan menampilkan hasil plotting seperti gambar 4.4 berikut



Gambar 4.4 Titik Pusat Klaster Algoritma K-Means++

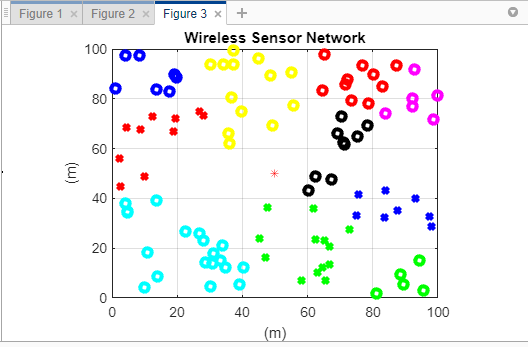
1. Ploting Hasil *Clustering* Algoritma *K-Means++*

Pada proses ini setelah titik pusat klaster dipilih dan masing-masing node dikelompokkan hingga titik pusat klaster tidak mengalami perubahan maka akan didapatkan hasil akhir klaster node yang dibuat sesuai dengan jumlah klaster yang ditentukan. Data node tiap klasternya disimpan dalam variabel outputx dan output. Berikut adalah *source code* proses plotting data hasil *clustering* dengan algoritma *K-Means++* pada tabel 4.17 berikut

Table 4.17 Ploting Hasil Clustering Algoritma K-Means++

|  |
| --- |
| %Menampilkan data round terakhir  if(rnd==round)  figure;  for i=1:k  xf=outputx{i};  yf=outputy{i};  if(i==1)  plot(xf,yf,'bo','linewidth',3);  elseif(i==2)  plot(xf,yf,'go','linewidth',3);  elseif(i==3)  plot(xf,yf,'ro','linewidth',3);  elseif(i==4)  plot(xf,yf,'co','linewidth',3);  elseif(i==5)  plot(xf,yf,'mo','linewidth',3);  elseif(i==6)  plot(xf,yf,'yo','linewidth',3);  elseif(i==7)  plot(xf,yf,'ko','linewidth',3);  elseif(i==8)  plot(xf,yf,'rx','linewidth',3);  elseif(i==9)  plot(xf,yf,'bx','linewidth',3);  else  plot(xf,yf,'gx','linewidth',3);  end  hold on;  grid on;  end  figure(3)  plot(xm,ym,sinkx,sinky,'\*r');  title 'Wireless Sensor Network';  xlabel '(m)';  ylabel '(m)';  hold on;  end |

Setelah source code diatas dijalankan maka akan menampilkan hasil plotting seperti gambar 4.5 berikut

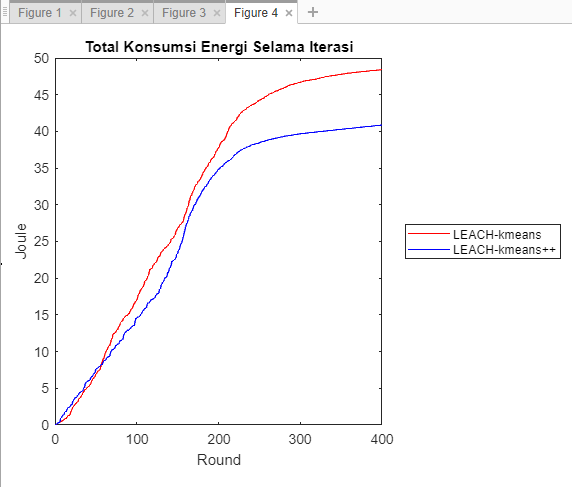


Gambar 4.5 Plotting Hasil Clustering Dengan Algoritma K-Means++

### **Pengujian Hasil Total Konsumsi Energi**

Total konsumsi energi merupakan jumlah energi yang dikeluarkan oleh setiap node saat proses pengiriman data yang ada dalam jaringan WSN. Dalam penelitian ini setiap total konsumsi energi yang dikeluarkan oleh 2 protokol jaringan yaitu LEACH yang dioptimasi dengan algoritma *K-Means* dengan LEACH yang dioptimasi dengan algoritma *K-Means++* telah disimpan dalam 2 file txt selama proses simulasi berlangsung.

Pada protokol LEACH dengan algoritma *K-Means* disimpan pada file TE-kmeans.txt dan protokol LEACH dengan algoritma *K-Means++*. Berdasarkan penelitian yang dilakukan selama 400 round, didapatkan hasil perbandingan total konsumsi energi seperti gambar 4.6 berikut.

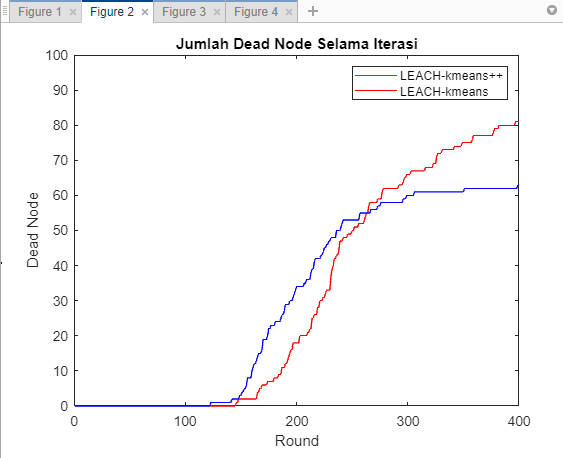


Gambar 4.6 Hasil Perbandingan Total Konsumsi Energi

Berdasarkan gambar 4.6 menunjukan bahwa protokol LEACH yang dimodifikasi dengan algoritma *K-Means++* atau KM-LEACH lebih hemat energi dibandingkan dengan LEACH yang dioptimasi dengan algoritma *K-Means,* dimana KM-LEACH menghabiskan total konsumsi energi sebesar 40.86 Joule. Lebih hemat dibandingkan LEACH dengan algoritma *K-Means* yang menghabiskan energi sebesar 48.40 Joule.

### **Pengujian Hasil Jumlah *Death Node***

*Death node* merupakan suatu kondisi dimana jika energi yang dimiliki oleh node kurang dari sama dengan 0 selama proses simulasi dalam topologi WSN. Dalam penelitian ini dilakukan perbandingan antara kedua protokol yang disimulasikan yaitu LEACH dengan algoritma *K-Means++* atau KM-LEACH dan LEACH yang dimodifikasi dengan algoritma *K-Mean,* dimana kedua hasil simulasi telah disimpan dalam 2 file txt yaitu TDN-kmeans.txt dan TDN-kmeansplus.txt. Berdasarkan penelitian yang dilakukan selama 400 round, didapatkan hasil perbandingan total *death node* seperti gambar 4.7 berikut.

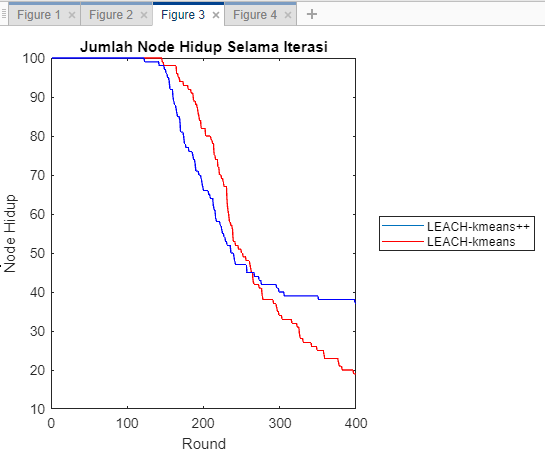


Gambar 4.7 Hasil Perbandingan Jumlah Dead Node

Berdasarkan gambar 4.7 menunjukan bahwa protokol LEACH yang dimodifikasi dengan algoritma *K-Means++* atau KM-LEACH lebih stabil dibandingkan dengan LEACH yang dioptimasi dengan algoritma *K-Means,* dimana KM-LEACH memiliki jumlah *death node* sebanyak 63 *node*. Lebih baik dibandingkan LEACH dengan algoritma *K-Means* yang memiliki jumlah *death node* sebanyak 81 *node*.

### **Pengujian Hasil Jumlah Node Hidup**

*Node alive* merupakan jumlah node hidup yang tersisa selama proses transmisi energi pada topologi WSN. Dalam penelitian ini dilakukan perbandingan antara kedua protokol yang disimulasikan yaitu LEACH dengan algoritma *K-Means++* atau KM-LEACH dan LEACH yang dimodifikasi dengan algoritma *K-Mean,* dimana kedua hasil simulasi telah disimpan dalam 2 file txt yaitu TNA-kmeans.txt dan TNA-kmeansplus.txt. Berdasarkan penelitian yang dilakukan selama 400 round, didapatkan hasil perbandingan total *node alive* seperti gambar 4.8 berikut.

****

Gambar 4.8 Perbandingan Jumlah Node Alive

Berdasarkan gambar 4.8 menunjukan bahwa protokol LEACH yang dimodifikasi dengan algoritma *K-Means++* atau KM-LEACH memiliki masa hidup lebih baik dibandingkan dengan LEACH yang dioptimasi dengan algoritma *K-Means,* dimana KM-LEACH memiliki jumlah *node alive* sebanyak 37 *node*. Lebih baik dibandingkan LEACH dengan algoritma *K-Means* yang memiliki jumlah *death node* sebanyak 19 *node*.

# **BAB V PENUTUP**

## **Kesimpulan**

Berdasarkan implementasi penelitian yang telah dilakukan dan hasil yang telah didapat, dapat ditarik kesimpulan sebagai berikut:

1. Algoritma yang diusulkan yaitu *K-Means++ Clustering* telah berhasil diimplementasikan pada LEACH protokol dengan menggunakan software Matlab 2015 A untuk mengoptimasi fase *setup phase* yang ada pada protokol ini.
2. Dari hasil yang didapat dan pengujian yang dilakukan dari segi total konsumsi energi yang dihasilkan terbukti KM-LEACH dengan algoritma *K-Means++* lebih efisien dan hemat energi dibandingkan dengan LEACH dengan algoritma K-Means yaitu sebesar 40.86 Joule berbanding 48.40 Joule selama proses simulasi berlangsung.
3. Dari hasil jumlah *death node* yang didapat juga menunjukkan bahwa KM-LEACH dengan algoritma ­­*K-Means++* yang memiliki total konsumsi energi yang lebih efisien dapat mengurangi jumlah *death node* yang dihasilkan dalam jaringan dibandingkan LEACH dengan algoritma *K-Means*, yaitu sebanyak 63 *node* berbanding 81 *node* selama proses simulasi berlangsung.
4. Efisiensi energi yang didapat juga berdampak pada jumlah *node alive* yang ada, dimana KM-LEACH dengan algoritma *K-Means++* dapat meningkatkan masa hidup node yang lebih baik dibandingkan LEACH dengan algoritma ­*K-Means,* yaitu sebesar 37 *node* berbanding 19 *node* selama proses simulasi berlangsung.

## **Saran**

Dari hasil penelitian yang didapatkan, penulis memberikan beberapa saran yang dapat digunakan untuk penelitian selanjutnya dengan menggunakan topik yang sama:

1. Algoritma *K-Means++* dapat menunjukkan hasil efisiensi energi lebih baik dibandingkan dengan algoritma *K-Means*, namun dari segi kompleksitasnya algoritma *K-Means++* lebih buruk karena proses iterasi yang dilakukan dalam pembentukan klaster lebih banyak dibandingkan dengan algoritma *K-Means* biasa yaitu dengan rata-rata 14 iterasi berbanding 8.35 iterasi sehingga jika diimplementasikan secara nyata pada node memerlukan spesifikasi memori node yg lebih besar. Dalam penelitian selanjutnya dapat menggunakan algoritma lain yang memiliki kompleksitas yg lebih rendah dibandingkan dengan metode yang diterapkan.
2. Pada penelitisn ini pembangkitan bilangan acak pada algoritma *K-Means* masih dilakukan secara manual melalui *read file* tidak secara otomatis pada program karena pada matlab belum memungkinkan membangkitkan bilangan acak unik berbeda dengan iterasi yang banyak. Sehingga dalam penilitian selanjutnya diharapkan pembangkitan bilangan acak dapat dilakukan secara otomatis melalui program.

# **DAFTAR PUSTAKA**

Arthur, D., and Sergei Vassilvitskii. (2007). k-means++: The Advantages of Careful Seeding. *Proc. of the annu. ACM-SIAM Symp. on Discrete Algorithm, Vol.8* hal.1027-1035. doi:10.1145/1283383.1283494

Heinzelman, W. B., Anantha P.Chandrakasan and Hari Balakrishnan. (2002). An Application-Specific Protocol Architecture for Wireless Microsensor Networks. *IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, VOL. 1, NO. 4, OCTOBER 2002*, hal.660-670.

Hibrian, M., Wardi and Agussalim. (2020). Peningkatan Network Lifetime Menggunakan Cluster Based Pada Wireless Sensor Network. *JITCE (Journal of Information Technology and Computer Engineering), Vol. 04 No. 01 (2020)*, hal.16-21.

Han, J., Micheline Kamber and Jian Pei (2011). Data Mining Concept and Techniques. USA, Elsevier Inc.

Novid, I., Delsina Faiza, Thamrin and Winda Agustiarmi. (2019). Analisa Perbandingan Routing Protokol Pada Wireless Sensor Networks (WSNs). *Jurnal Vokasional Teknik Elektronika dan Informatika, Vol. 7, No. 2, Juni 2019*.

Permana, M. A. (2015). Analisa Algoritma LEACH Pada Jaringan Sensor Nirkabel. *Proceeding Seminar Tugas Akhir Jurusan Teknik Elektro FTI-ITS*.

Ridwan, A., Rian Ferdian and Rahmadia Kurnia. (2020). Optimasi Protokol LEACH Untuk Meningkatkan Stabilitas Pada Wireless Sensor Network. *RESTI (Rekayasa Sistem dan Teknologi Informasi, Vol.4 No.1*, hal.193 -200.

Saheb, P., and Kanika Sharma. (2017). Improved LEACH Protocol Based on K-Means Clustering Algorithm for Wireless Sensor Network. *International Journal of Electronics & Communication Technology, 8*(4).

Salem, A. O. A., and Noor Shudifat. (2019). Enhanced LEACH protocol for increasing a lifetime of WSNs. *Springer*(23), hal.901–907.

Sohraby, K., Daniel Minoli and Taieb Znati (2007). Wireless Sensor Networks Technology, Protocols, and Applications. New Jersey, John Wiley & Sons, Inc.

Susanto, E. B. (2016). Evaluasi Hasil Klaster Pada Dataset Iris, Soybean-small, Wine Menggunakan Algoritma Fuzzy C-Means dan Kmeans++. *SURYA INFORMATIKA, Vol. 2 No. 1 - Mei 2016*.

Tonapa, J. R., and Indrastanti Ratna W. (2016). Analisis Ketahanan Energi Oleh Low Energy Adaptive Clustering Hierarchy (LEACH) Pada Cluster Head Wireless Sensor Network (WSN). *Semantic Scholar*.

# **LAMPIRAN**

Lampiran 1. Titik Kordinat Node

|  |  |  |
| --- | --- | --- |
| No | Kordinat x | Kordinat y |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100 | 78.287  0.98023  92.233  4.266  70.434  22.428  67.303  62.372  17.712  76.692  10.789  9.9095  19.325  9.909  55.73  31.194  33.896  55.015  62.892  39.085  60.128  99.756  48.565  13.755  92.736  71.357  34.329  12.477  64.648  39.828  83.522  65.226  74.931  61.947  75.651  49.235  97.273  83.78  95.417  35.687  28.15  71.113  69.061  4.7555  45.134  71.505  28.151  13.776  13.86  36.616  62.378  87.705  44.944  4.2298  18.921  8.644  36.102  81.115  8.3874  65.135  40.349  26.844  33.167  34.801  88.415  93.004  4.7401  73.597  4.491  89.363  30.366  19.548  72.175  58.243  92.274  28.595  98.478  83.897  47.062  26.909  75.389  30.775  47.557  78.811  66.851  2.1556  30.082  98.09  80.082  19.753  94.373  72.839  2.5857  64.63  37.231  82.953  37.253  87.255  66.846  65.385 | 69.379  84.321  77.095  37.819  72.951  26.905  47.749  23.644  82.964  93.448  18.223  48.976  89.589  4.4166  77.25  17.898  21.015  90.636  10.153  5.4617  43.172  81.16  89.445  39  91.749  61.834  93.603  73.059  83.315  74.982  32.246  97.913  33.042  36.064  41.39  69.474  32.775  73.907  3.1923  66.265  23.038  62.457  66.044  34.878  24.09  85.618  73.105  83.672  8.821  80.676  48.959  35.314  96.353  97.296  66.712  67.511  62.028  1.9257  97.48  23.124  12.202  25.785  15.223  12.166  9.4278  39.902  34.237  79.468  68.622  5.4792  4.6192  72.017  87.78  7.0684  80.037  14.366  71.568  43.326  16.071  74.902  64.681  13.872  36.246  78.03  13.35  55.984  93.941  28.662  89.611  88.402  14.916  27.676  44.653  12.12  93.713  84.909  99.318  93.35  20.678  7.2052 |

Lampiran 2. Data Bilangan Acak

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No Round | Klaster | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399  400 | 99  15  78  89  29  49  52  59  64  52  21  82  83  68  19  95  98  7  44  25  65  7  51  16  56  85  53  72  88  97  5  4  74  78  75  52  1  78  80  82  32  56  99  5  27  69  100  49  55  76  83  99  3  39  34  86  94  56  79  15  43  40  33  29  30  15  26  16  46  3  94  79  2  100  34  22  30  52  23  65  91  50  20  88  95  10  88  3  29  49  45  42  16  24  18  18  21  69  76  1  99  15  78  89  29  49  52  59  64  52  21  82  83  68  19  95  68  7  44  25  65  7  51  16  56  85  53  72  88  97  5  4  74  78  75  52  1  78  80  82  32  56  99  5  27  69  100  49  55  76  83  99  3  39  34  86  94  56  79  15  43  40  33  29  30  15  26  16  46  3  94  79  2  100  34  22  30  52  23  65  91  50  20  88  95  10  88  3  29  49  45  42  16  24  18  18  21  69  76  1  99  15  78  89  29  49  52  59  64  52  21  82  83  68  19  95  68  7  44  25  65  7  51  16  56  85  53  72  88  97  5  4  74  78  75  52  1  78  80  82  32  56  99  5  27  69  100  49  55  76  83  99  3  39  34  86  94  56  79  15  43  40  33  29  30  15  26  16  46  3  94  79  2  100  34  22  30  52  23  65  91  50  20  88  95  10  88  3  29  49  45  42  16  24  18  18  21  69  76  1  99  15  78  89  29  49  52  59  64  52  21  82  83  68  19  95  68  7  44  25  65  7  51  16  56  85  53  72  88  97  5  4  74  78  75  52  1  78  80  82  32  56  99  5  27  69  100  49  55  76  83  99  3  39  34  86  94  56  79  15  43  40  33  29  30  15  26  16  46  3  94  79  2  100  34  22  30  52  23  65  91  50  20  88  95  10  88  3  29  49  45  42  16  24  18  18  21  69  76  1 | 35  50  10  31  7  51  18  11  23  70  57  51  28  46  99  91  93  46  78  67  81  11  96  88  62  97  100  45  10  51  34  74  30  86  62  23  28  100  95  43  11  21  94  78  97  16  61  81  88  22  61  94  60  92  17  32  30  40  1  59  47  94  77  24  23  30  60  99  72  88  79  29  85  91  10  11  6  46  41  39  13  41  28  78  65  32  97  83  23  34  48  93  80  10  87  89  81  34  26  75  35  50  10  31  7  51  18  11  23  70  57  51  28  46  99  91  93  46  78  67  81  11  96  88  62  97  100  45  10  51  34  74  30  86  62  23  28  100  95  43  11  21  94  78  97  16  61  81  88  22  61  94  60  92  17  32  30  40  1  59  47  94  77  24  23  30  60  99  72  88  79  29  85  91  10  11  6  46  41  39  13  41  28  78  65  32  97  83  23  34  48  93  80  10  87  89  81  34  26  75  35  50  10  31  7  51  18  11  23  70  57  51  28  46  99  91  93  46  78  67  81  11  96  88  62  97  100  45  10  51  34  74  30  86  62  23  28  100  95  43  11  21  94  78  97  16  61  81  88  22  61  94  60  92  17  32  30  40  1  59  47  94  77  24  23  30  60  99  72  88  79  29  85  91  10  11  6  46  41  39  13  41  28  78  65  32  97  83  23  34  48  93  80  10  87  89  81  34  26  75  35  50  10  31  7  51  18  11  23  70  57  51  28  46  99  91  93  46  78  67  81  11  96  88  62  97  100  45  10  51  34  74  30  86  62  23  28  100  95  43  11  21  94  78  97  16  61  81  88  22  61  94  60  92  17  32  30  40  1  59  47  94  77  24  23  30  60  99  72  88  79  29  85  91  10  11  6  46  41  39  13  41  28  78  65  32  97  83  23  34  48  93  80  10  87  89  81  34  26  75 | 93  29  99  57  73  83  26  41  36  55  41  70  60  10  67  46  29  75  6  66  48  54  24  72  84  46  87  9  4  21  87  71  19  22  30  84  20  66  96  61  26  61  73  26  19  99  28  79  8  80  50  68  36  87  71  40  25  98  85  67  100  92  35  71  60  11  11  57  2  82  2  6  15  53  89  8  11  77  97  31  35  30  3  13  27  73  38  29  86  16  76  32  2  16  49  75  75  28  94  53  93  29  99  57  73  83  26  41  36  55  41  70  60  10  67  46  29  75  6  66  48  54  24  72  84  46  87  9  4  21  87  71  19  22  30  84  20  66  96  61  26  61  73  26  19  99  28  79  8  80  50  68  36  87  71  40  25  98  85  67  100  92  35  71  60  11  11  57  2  82  2  6  15  53  89  8  11  77  97  31  35  30  3  13  27  73  38  29  86  16  76  32  2  16  49  75  75  28  94  53  93  29  99  57  73  83  26  41  36  55  41  70  60  10  67  46  29  75  6  66  48  54  24  72  84  46  87  9  4  21  87  71  19  22  30  84  20  66  96  61  26  61  73  26  19  99  28  79  8  80  50  68  36  87  71  40  25  98  85  67  100  92  35  71  60  11  11  57  2  82  2  6  15  53  89  8  11  77  97  31  35  30  3  13  27  73  38  29  86  16  76  32  2  16  49  75  75  28  94  53  93  29  99  57  73  83  26  41  36  55  41  70  60  10  67  46  29  75  6  66  48  54  24  72  84  46  87  9  4  21  87  71  19  22  30  84  20  66  96  61  26  61  73  26  19  99  28  79  8  80  50  68  36  87  71  40  25  98  85  67  100  92  35  71  60  11  11  57  2  82  2  6  15  53  89  8  11  77  97  31  35  30  3  13  27  73  38  29  86  16  76  32  2  16  49  75  75  28  94  53 | 54  6  74  65  36  29  86  16  58  58  23  89  54  86  82  17  16  48  95  94  9  15  12  24  66  16  85  85  92  24  43  33  27  75  90  78  21  59  98  97  78  54  70  25  80  44  6  10  16  21  87  35  61  79  8  37  53  5  26  86  80  63  4  46  31  18  34  79  18  98  8  28  11  31  92  12  15  97  24  32  57  10  40  27  17  35  50  60  2  97  42  34  28  50  66  98  26  16  47  8  54  6  74  65  36  29  86  16  58  58  23  89  54  86  82  17  16  48  95  94  9  15  12  24  66  16  85  85  92  24  43  33  27  75  90  78  21  59  98  97  78  54  70  25  80  44  6  10  16  21  87  35  61  79  8  37  53  5  26  86  80  63  4  46  31  18  34  79  18  98  8  28  11  31  92  12  15  97  24  32  57  10  40  27  17  35  50  60  2  97  42  34  28  50  66  98  26  16  47  8  54  6  74  65  36  29  86  16  58  58  23  89  54  86  82  17  16  48  95  94  9  15  12  24  66  16  85  85  92  24  43  33  27  75  90  78  21  59  98  97  78  54  70  25  80  44  6  10  16  21  87  35  61  79  8  37  53  5  26  86  80  63  4  46  31  18  34  79  18  98  8  28  11  31  92  12  15  97  24  32  57  10  40  27  17  35  50  60  2  97  42  34  28  50  66  98  26  16  47  8  54  6  74  65  36  29  86  16  58  58  23  89  54  86  82  17  16  48  95  94  9  15  12  24  66  16  85  85  92  24  43  33  27  75  90  78  21  59  98  97  78  54  70  25  80  44  6  10  16  21  87  35  61  79  8  37  53  5  26  86  80  63  4  46  31  18  34  79  18  98  8  28  11  31  92  12  15  97  24  32  57  10  40  27  17  35  50  60  2  97  42  34  28  50  66  98  26  16  47  8 | 21  74  80  72  61  78  29  12  69  83  49  45  94  19  96  1  66  37  41  54  88  9  52  73  20  15  39  18  24  27  46  64  45  4  42  92  65  25  27  87  53  100  16  1  94  27  88  27  72  20  80  32  12  83  61  56  95  82  99  9  90  12  75  96  39  38  14  31  28  83  26  90  72  45  18  27  75  37  49  14  15  92  35  89  83  40  49  75  88  40  59  52  98  18  11  29  50  100  89  39  21  74  80  72  61  78  29  12  69  83  49  45  94  19  96  1  66  37  41  54  88  9  52  73  20  15  39  18  24  27  46  64  45  4  42  92  65  25  27  87  53  100  16  1  94  27  88  27  72  20  80  32  12  83  61  56  95  82  99  9  90  12  75  96  39  38  14  31  28  83  26  90  72  45  18  27  75  37  49  14  15  92  35  89  83  40  49  75  88  40  59  52  98  18  11  29  50  100  89  39  21  74  80  72  61  78  29  12  69  83  49  45  94  19  96  1  66  37  41  54  88  9  52  73  20  15  39  18  24  27  46  64  45  4  42  92  65  25  27  87  53  100  16  1  94  27  88  27  72  20  80  32  12  83  61  56  95  82  99  9  90  12  75  96  39  38  14  31  28  83  26  90  72  45  18  27  75  37  49  14  15  92  35  89  83  40  49  75  88  40  59  52  98  18  11  29  50  100  89  39  21  74  80  72  61  78  29  12  69  83  49  45  94  19  96  1  66  37  41  54  88  9  52  73  20  15  39  18  24  27  46  64  45  4  42  92  65  25  27  87  53  100  16  1  94  27  88  27  72  20  80  32  12  83  61  56  95  82  99  9  90  12  75  96  39  38  14  31  28  83  26  90  72  45  18  27  75  37  49  14  15  92  35  89  83  40  49  75  88  40  59  52  98  18  11  29  50  100  89  39 | 76  2  62  76  24  73  1  99  3  57  98  68  78  74  73  10  98  43  79  92  22  18  54  37  27  48  63  83  17  54  94  100  63  15  7  15  71  44  14  14  17  37  60  94  85  100  92  82  31  2  90  74  13  7  49  74  79  95  91  6  41  96  32  28  8  51  70  32  44  86  21  13  37  13  9  9  95  16  44  91  11  68  88  59  73  25  75  73  82  25  2  43  99  86  59  55  12  63  1  95  76  2  62  76  24  73  1  99  3  57  98  68  78  74  73  10  98  43  79  92  22  18  54  37  27  48  63  83  17  54  94  100  63  15  7  15  71  44  14  14  17  37  60  94  85  100  92  82  31  2  90  74  13  7  49  74  79  95  91  6  41  96  32  28  8  51  70  32  44  86  21  13  37  13  9  9  95  16  44  91  11  68  88  59  73  25  75  73  82  25  2  43  99  86  59  55  12  63  1  95  76  2  62  76  24  73  1  99  3  57  98  68  78  74  73  10  98  43  79  92  22  18  54  37  27  48  63  83  17  54  94  100  63  15  7  15  71  44  14  14  17  37  60  94  85  100  92  82  31  2  90  74  13  7  49  74  79  95  91  6  41  96  32  28  8  51  70  32  44  86  21  13  37  13  9  9  95  16  44  91  11  68  88  59  73  25  75  73  82  25  2  43  99  86  59  55  12  63  1  95  76  2  62  76  24  73  1  99  3  57  98  68  78  74  73  10  98  43  79  92  22  18  54  37  27  48  63  83  17  54  94  100  63  15  7  15  71  44  14  14  17  37  60  94  85  100  92  82  31  2  90  74  13  7  49  74  79  95  91  6  41  96  32  28  8  51  70  32  44  86  21  13  37  13  9  9  95  16  44  91  11  68  88  59  73  25  75  73  82  25  2  43  99  86  59  55  12  63  1  95 | 73  70  76  82  14  70  32  98  83  51  34  43  37  87  36  84  40  36  20  71  10  3  85  52  19  71  55  28  40  86  88  32  38  76  93  58  64  24  16  92  52  46  84  62  32  39  36  96  40  15  75  92  32  88  29  9  54  88  12  25  53  35  76  60  2  89  61  56  94  1  81  40  95  17  9  23  8  35  95  70  67  95  95  68  16  86  6  15  28  96  70  92  20  23  46  43  58  76  35  97  73  70  76  82  14  70  32  98  83  51  34  43  37  87  36  84  40  36  20  71  10  3  85  52  19  71  55  28  40  86  88  32  38  76  93  58  64  24  16  92  52  46  84  62  32  39  36  96  40  15  75  92  32  88  29  9  54  88  12  25  53  35  76  60  2  89  61  56  94  1  81  40  95  17  9  23  8  35  95  70  67  95  95  68  16  86  6  15  28  96  70  92  20  23  46  43  58  76  35  97  73  70  76  82  14  70  32  98  83  51  34  43  37  87  36  84  40  36  20  71  10  3  85  52  19  71  55  28  40  86  88  32  38  76  93  58  64  24  16  92  52  46  84  62  32  39  36  96  40  15  75  92  32  88  29  9  54  88  12  25  53  35  76  60  2  89  61  56  94  1  81  40  95  17  9  23  8  35  95  70  67  95  95  68  16  86  6  15  28  96  70  92  20  23  46  43  58  76  35  97  73  70  76  82  14  70  32  98  83  51  34  43  37  87  36  84  40  36  20  71  10  3  85  52  19  71  55  28  40  86  88  32  38  76  93  58  64  24  16  92  52  46  84  62  32  39  36  96  40  15  75  92  32  88  29  9  54  88  12  25  53  35  76  60  2  89  61  56  94  1  81  40  95  17  9  23  8  35  95  70  67  95  95  68  16  86  6  15  28  96  70  92  20  23  46  43  58  76  35  97 | 17  31  36  56  51  8  88  91  22  91  9  38  45  26  65  98  95  42  28  91  37  14  81  81  37  72  37  36  32  31  29  45  66  30  19  27  16  31  68  50  62  23  41  58  67  49  73  13  48  81  54  5  29  43  10  5  91  25  63  74  23  76  73  33  61  54  52  82  25  94  91  48  70  82  48  57  42  26  100  28  94  52  70  95  98  61  71  86  19  84  75  98  76  27  90  13  73  26  55  4  17  31  36  56  51  8  88  91  22  91  9  38  45  26  65  98  95  42  28  91  37  14  81  81  37  72  37  36  32  31  29  45  66  30  19  27  16  31  68  50  62  23  41  58  67  49  73  13  48  81  54  5  29  43  10  5  91  25  63  74  23  76  73  33  61  54  52  82  25  94  91  48  70  82  48  57  42  26  100  28  94  52  70  95  98  61  71  86  19  84  75  98  76  27  90  13  73  26  55  4  17  31  36  56  51  8  88  91  22  91  9  38  45  26  65  98  95  42  28  91  37  14  81  81  37  72  37  36  32  31  29  45  66  30  19  27  16  31  68  50  62  23  41  58  67  49  73  13  48  81  54  5  29  43  10  5  91  25  63  74  23  76  73  33  61  54  52  82  25  94  91  48  70  82  48  57  42  26  100  28  94  52  70  95  98  61  71  86  19  84  75  98  76  27  90  13  73  26  55  4  17  31  36  56  51  8  88  91  22  91  9  38  45  26  65  98  95  42  28  91  37  14  81  81  37  72  37  36  32  31  29  45  66  30  19  27  16  31  68  50  62  23  41  58  67  49  73  13  48  81  54  5  29  43  10  5  91  25  63  74  23  76  73  33  61  54  52  82  25  94  91  48  70  82  48  57  42  26  100  28  94  52  70  95  98  61  71  86  19  84  75  98  76  27  90  13  73  26  55  4 | 90  7  66  13  11  40  73  85  99  8  39  78  36  79  61  31  6  4  46  48  77  83  58  57  100  20  4  98  45  9  24  41  71  1  28  96  95  19  51  21  14  17  42  22  5  3  97  94  70  50  37  23  93  23  15  35  67  91  25  43  69  55  12  86  68  62  59  54  80  10  63  89  74  44  53  75  92  61  65  88  49  34  82  52  56  2  18  81  30  26  27  87  33  64  76  72  29  60  38  47  90  7  66  13  11  40  73  85  99  8  39  78  36  79  61  31  6  4  46  48  77  83  58  57  100  20  4  98  45  9  24  41  71  1  28  96  95  19  51  21  14  17  42  22  5  3  97  94  70  50  37  23  93  23  15  35  67  91  25  43  69  55  12  86  68  62  59  54  80  10  63  89  74  44  53  75  92  61  65  88  49  34  82  52  56  2  18  81  30  26  27  87  33  64  76  72  29  60  38  47  90  7  66  13  11  40  73  85  99  8  39  78  36  79  61  31  6  4  46  48  77  83  58  57  100  20  4  98  45  9  24  41  71  1  28  96  95  19  51  21  14  17  42  22  5  3  97  94  70  50  37  23  93  23  15  35  67  91  25  43  69  55  12  86  68  62  59  54  80  10  63  89  74  44  53  75  92  61  65  88  49  34  82  52  56  2  18  81  30  26  27  87  33  64  76  72  29  60  38  47  90  7  66  13  11  40  73  85  99  8  39  78  36  79  61  31  6  4  46  48  77  83  58  57  100  20  4  98  45  9  24  41  71  1  28  96  95  19  51  21  14  17  42  22  5  3  97  94  70  50  37  23  93  23  15  35  67  91  25  43  69  55  12  86  68  62  59  54  80  10  63  89  74  44  53  75  92  61  65  88  49  34  82  52  56  2  18  81  30  26  27  87  33  64  76  72  29  60  38  47 | 87  1  97  64  78  77  19  71  6  95  18  72  39  96  90  11  34  66  53  79  83  84  26  51  16  54  13  68  42  61  81  80  31  59  41  4  49  33  100  98  76  2  65  60  91  88  22  55  86  53  52  93  63  36  23  95  44  50  9  56  40  70  38  45  15  69  85  72  89  27  28  20  62  2  3  47  90  92  14  7  25  43  94  21  58  17  35  82  8  48  29  12  73  99  75  42  67  57  37  10  87  1  97  64  78  77  19  71  6  95  18  72  39  96  90  11  34  66  53  79  83  84  26  51  16  54  13  68  42  61  81  80  31  59  41  4  49  33  100  98  76  2  65  60  91  88  22  55  86  53  52  93  63  36  23  95  44  50  9  56  40  70  38  45  15  69  85  72  89  27  28  20  62  2  3  47  90  92  14  7  25  43  94  21  58  17  35  82  8  48  29  12  73  99  75  42  67  57  37  10  87  1  97  64  78  77  19  71  6  95  18  72  39  96  90  11  34  66  53  79  83  84  26  51  16  54  13  68  42  61  81  80  31  59  41  4  49  33  100  98  76  2  65  60  91  88  22  55  86  53  52  93  63  36  23  95  44  50  9  56  40  70  38  45  15  69  85  72  89  27  28  20  62  2  3  47  90  92  14  7  25  43  94  21  58  17  35  82  8  48  29  12  73  99  75  42  67  57  37  10  87  1  97  64  78  77  19  71  6  95  18  72  39  96  90  11  34  66  53  79  83  84  26  51  16  54  13  68  42  61  81  80  31  59  41  4  49  33  100  98  76  2  65  60  91  88  22  55  86  53  52  93  63  36  23  95  44  50  9  56  40  70  38  45  15  69  85  72  89  27  28  20  62  2  3  47  90  92  14  7  25  43  94  21  58  17  35  82  8  48  29  12  73  99  75  42  67  57  37  10 |

Lampiran 3. Data Total Konsumsi Energi

|  |  |  |  |
| --- | --- | --- | --- |
| Iterasi | Total Konsumsi Energi (Joule)  Algoritma K-Means | Iterasi | Total Konsumsi Energi (Joule)  Algoritma K-Means++ |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399  400 | 0.0598697133004236  0.137131060271127  0.20215929796957  0.268546991520651  0.323675029438688  0.388328626974386  0.442359242786202  0.517180166726883  0.576626545597496  0.641150132715089  0.735866422118487  0.79373159235674  0.868141498615963  0.936169206604998  1.15334795145812  1.22234421924249  1.2854873190906  1.34673791820745  1.67947982945695  1.99919774734145  2.18408893795787  2.50505156115539  2.57809211622866  2.74260674869996  2.89341009809755  2.9576782088264  3.12069721363103  3.19638266165039  3.50989248067046  3.58689131171225  3.66932303299555  3.9693534132312  4.03094893998105  4.32994993452154  4.50759243751887  4.74385866225823  4.84965882168958  4.96654561628828  5.0313870329901  5.09349726224153  5.19871527969977  5.40115702447516  5.45510289442919  5.71219076650448  5.973604643341  6.22735626357214  6.2913957386889  6.51971718308348  6.58954267802138  6.97563998128815  7.02618873337126  7.23533619704807  7.34586235518991  7.43006892114629  7.58013540238359  7.87751710904665  8.33908782264201  8.70983144478783  9.00298543260603  9.13462953448214  9.63899819510826  9.90712989529353  10.184422975173  10.2549513369432  10.6340058195878  10.706919945127  10.8723535078713  11.2138079852279  11.6797278260842  11.8613138480383  12.3904276102571  12.4543451013282  12.5354730493851  12.6192921520948  12.8363781877171  12.9229381559026  13.0773401520405  13.3692704426633  13.6019337083029  13.888368863783  13.9992065271618  14.1460684371958  14.2740464065388  14.5646625688395  14.6997850868991  14.7637951213181  14.821402032386  14.9210827331781  14.9806832778572  15.1686260826173  15.2343915552468  15.5269281656456  15.77763656073  15.8316908989359  15.9997115307767  16.3980835779043  16.5312912221452  16.681746701436  16.9505406210983  17.0191016315102  17.3578549403357  17.8235398580112  17.8772293734815  18.2488783357456  18.4018233911568  18.6405126842018  18.7705976579948  18.8509937119911  19.1962556595445  19.6279949719027  19.6830175238493  19.8189872850872  20.1263993794376  20.3677495877154  20.7015281494853  21.1693932816293  21.2303782666925  21.287364934998  21.4092249955038  21.7676675396078  21.8701504420991  22.0414268838631  22.1079457368127  22.2673718853579  22.4732761991034  22.825017593019  22.934671870189  22.9919901101517  23.3195461869013  23.5571629465436  23.6340152455018  23.7841617401838  23.9912288153016  24.0650555844926  24.1198799559854  24.1909951327269  24.3565567225032  24.4238769466392  24.6554056537035  24.7140489829383  24.7910806546673  25.2945104292732  25.3607653604478  25.4897892535251  25.551985336923  25.7361667386824  25.8424276091086  26.3809552522386  26.5547475610971  26.7960096393214  26.8462539207113  27.2125922252148  27.2711664210898  27.3992653099649  27.473107558202  27.5276418656067  27.8461966044484  28.3158367383619  28.7266805906962  28.8395789229025  29.2391992216289  29.6852466614417  29.8628812932926  30.4081426511018  30.7302087302324  31.1545111082076  31.2595029630489  31.6523105333144  31.9112018371682  31.9725583964969  32.4356865886082  32.6827838405316  32.7341243797258  32.9245849813172  33.2113072554083  33.2777098019366  33.3818002552757  33.6321722144179  33.7989474559372  34.0268748005878  34.3359762126407  34.513428732113  34.6338717187007  34.9325038867679  35.0374238304603  35.0896087250312  35.3449765661295  35.6336763014352  35.7088329263134  35.9060681547237  36.1647770403337  36.3196092587967  36.370131924959  36.4096429025663  36.7628290750501  36.9924519174457  37.3014025395705  37.3859953743831  37.6164332947734  37.8066812695596  37.9352390900027  38.2559454789496  38.4699996480979  38.5966733626147  38.6438429417696  38.7152797244389  38.8655649484913  39.0243289758112  39.2428486975696  39.5669005405967  39.9426138507598  40.1051984747419  40.4474485970684  40.6629055172138  40.7696892997564  40.935906802313  41.0476723275364  41.1615802797149  41.3210783976633  41.3732990353842  41.488208676589  41.6069835745209  41.8737508320825  42.0133021671181  42.1468074109381  42.3932967893669  42.4883349999111  42.5591314499215  42.6662766402265  42.7156477722787  42.8867638094586  42.9275652814628  42.994756744755  43.1074336994031  43.1690608255855  43.2843828294264  43.3173302543937  43.4524904897245  43.522578927283  43.5588486215353  43.5843285146051  43.6689645392235  43.721298823597  43.748454416059  43.8275132652613  43.9256269684656  43.9632049736821  44.1030940642861  44.1616502163374  44.2624407532841  44.295758575456  44.3479120978738  44.4006651975078  44.4602197827603  44.5062371697619  44.585952182964  44.6741784959299  44.7451163899939  44.8035602575894  44.8707675907697  44.9844083160898  45.0875964566329  45.1316232709036  45.1717268045888  45.20284657854  45.2371424162066  45.2832537960374  45.418033247699  45.4329935220794  45.451864978416  45.5145126933095  45.5344936329331  45.6021529902805  45.6568677340166  45.7027898110095  45.7643867884369  45.7869692952583  45.8287412593528  45.8446208990245  45.9093106801201  45.961154285461  46.0282601818532  46.0658816532152  46.1040647498739  46.1942070549528  46.2382538773542  46.3104401002932  46.3361268597699  46.3509014566663  46.3964402362764  46.4117037467892  46.4359773533058  46.4795718311086  46.5227354227707  46.5351798202351  46.5670463708566  46.620351548184  46.6431186767279  46.6604762197316  46.6870946200464  46.7170217170828  46.7467419991638  46.7966016897674  46.8234139192791  46.8550791087609  46.885514926083  46.898345983916  46.9100931574542  46.9314462745944  46.9427846277073  46.9608795035353  47.0008624916722  47.0130956412232  47.0282980635615  47.0589741863341  47.0733362407301  47.0834494008695  47.102365054511  47.1163749329371  47.1370426181996  47.1756407009163  47.1839017898057  47.2214792184186  47.2549657918181  47.2807216835909  47.3036912191509  47.3473376596571  47.3773323946991  47.3860788899623  47.4248143294237  47.4701300164758  47.4799522811248  47.503735385367  47.5123996540767  47.5207120074667  47.5544083624328  47.5612736059034  47.5723040597976  47.5811346612276  47.5971932893083  47.6205525159647  47.6371163136215  47.6568664712232  47.6687331374075  47.6776803169268  47.6969453450891  47.7245685438289  47.7511797455424  47.7669653953159  47.7726037508081  47.804129407496  47.822019837363  47.8313662433392  47.8412817706994  47.8494673183686  47.85747184552  47.8793070959423  47.9133791346325  47.9322911000602  47.941313509258  47.9469645086347  47.9529021944788  47.9791278590804  47.9901086189334  47.999025428686  48.0231258439002  48.0290456324025  48.0457441186891  48.0631689896426  48.069467656893  48.0753975093762  48.0850739080339  48.1084510000757  48.1134593694733  48.1215281962915  48.1373591881859  48.1545565829881  48.1800582837698  48.1897770602778  48.1960625107107  48.2031778629665  48.2063868855439  48.2132347639358  48.2326149819072  48.238503109886  48.2675146760599  48.274765375333  48.2844339078177  48.2885069093355  48.2936755605228  48.312199295777  48.3183710728409  48.3368357793049  48.3414354017487  48.3493578326883  48.3602188637481  48.3735330980707  48.384975080755  48.3930003293349  48.4016440535444 | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399  400 | 0.0776744585062477  0.137595102425465  0.203400539182279  0.274159114920395  0.326052725636506  0.403727184142754  0.93346496937776  0.993385613296978  1.27138561660108  1.3371910533579  1.41486551186414  1.74900950980787  1.81976808554599  1.87968872946521  2.26295835211351  2.34063281061975  2.39252642133586  2.45833185809268  2.66669745221186  2.72661809613107  2.80429255463732  3.33403033987233  3.40478891561044  3.63986282563367  3.70566826239048  3.78334272089673  3.84326336481595  4.12126336812006  4.17315697883617  4.38596135972984  4.46363581823609  4.53439439397421  4.59431503789342  4.66012047465024  4.99426447259397  5.07193893110021  5.60167671633522  5.82578512043689  5.8857057643561  5.93759937507221  6.01527383357846  6.08107927033528  6.15183784607339  6.53510746872169  6.59502811264091  6.67270257114716  6.95070257445126  7.02690136696561  7.09270680372242  7.62244458895743  7.70011904746367  7.76003969138289  7.83079826712101  7.88269187783712  8.0910574719563  8.16873193046254  8.23453736721936  8.29445801113858  8.6286020090823  8.70977844346484  8.78745290197109  8.8582114777092  9.13621148101331  9.19613212493253  9.26193756168934  9.33961202019559  9.3915056309117  9.92124341614671  10.1563173261699  10.2162379700891  10.2939124285954  10.3646710043335  10.4304764410903  10.8137460637386  10.8645505829005  10.9422250414067  11.0021456853259  11.054039296042  11.3881832939858  11.4539887307426  11.5316631892488  11.602421764987  11.6623424089062  12.1920801941412  12.4700801974453  12.5477546559515  12.7605590368452  12.826364473602  12.8862851175212  12.9381787282374  13.0158531867436  13.0866117624817  13.2949773566009  13.386487515029  13.4464081589482  13.5240826174544  13.5898880542112  14.1196258394463  14.5028954620946  14.5736540378327  14.6513284963389  14.7112491402581  14.7631427509742  14.8289481877311  15.1069481910352  15.1846226495414  15.5187666474851  15.5786872914044  15.802795695506  15.8735542712441  15.9512287297504  16.0170341665072  16.5467719517422  16.6066925956614  16.6585862063775  16.7362606648838  16.971334574907  17.0475092642861  17.1133147010429  17.1732353449622  17.2509098034684  17.3216683792065  17.5996683825106  17.796038438113  17.8477469532281  17.9242225698483  17.9832195187665  18.5277498487795  18.5887656693552  19.1380764958593  19.2145521124794  19.5383549747129  19.5973519236311  19.9742222348526  20.0404422325498  20.1169178491699  20.2738186182675  20.7811946178557  20.8401915667738  21.3895023932779  21.4659780098981  21.7363336404517  22.2685531417578  22.3294591084073  22.3883086143698  22.4642089969606  22.648493959746  22.7257305202693  23.2658105676403  23.3313228419482  23.8151616481374  23.8732199722772  24.3489978930447  24.6416616945928  24.8170865021437  25.267436870111  25.7166242616846  26.1506757713713  26.603289161196  27.0467867151742  27.4432145344413  27.630152112103  28.0205050533827  28.4078266227899  28.5530280392686  28.9078109712398  29.0764598862369  29.1785118984206  29.5396314223002  29.8717416093335  30.0471805990577  30.0965338971616  30.4222054377009  30.7120448155123  30.8654416108308  31.0224272744579  31.1916746477543  31.4018592518599  31.6904102698686  31.8724248680123  32.023481068786  32.1505868440006  32.3970179432611  32.4734956465896  32.5171674643877  32.7746460925895  32.907382497696  33.1780118300349  33.3256544817653  33.4272358135155  33.6320074032143  33.8376169778125  33.9195748923237  34.0163853518102  34.1307422838599  34.254345823484  34.4191938744787  34.5041720618067  34.7023854178178  34.8089412875796  34.9196809725123  35.0005261809302  35.103365628259  35.1971095264953  35.2705020041623  35.4105191196444  35.5324528359688  35.6634150782393  35.7657184792737  35.8256242652728  35.9311744038559  35.9684778367236  36.0350611788179  36.1644403871498  36.1757146955832  36.3345721619699  36.4631135639609  36.599398828448  36.7056002462248  36.8085780497984  36.9320260151219  37.0226355370392  37.0747099351846  37.184144774287  37.2429730462499  37.31294791354  37.4025959094127  37.4666070574742  37.5272097289117  37.5641071794066  37.6586313367139  37.6717303249824  37.7473493302083  37.7967737792352  37.8533994598107  37.9002458408428  37.9486550553922  38.0055992738541  38.0373143543528  38.1221362866435  38.1733121042909  38.1942065253452  38.2091784422713  38.22993129201  38.2903669138692  38.3167568064688  38.3353957106592  38.3504990971747  38.3726861725745  38.4451304190845  38.4933106802632  38.5547393849984  38.5717500826493  38.5876644875801  38.6255531573799  38.6519430499794  38.6969235678287  38.7440257629379  38.7591702872054  38.8056094826341  38.841817705379  38.8582244189633  38.8724934585382  38.8942279641838  38.9354396679439  38.9717815997435  38.9876640496082  38.9999472234613  39.0112056001161  39.0704699459604  39.094514224443  39.1126976437449  39.1495745161195  39.1639160652571  39.1996479518354  39.221760306762  39.2373228112843  39.2538436201205  39.2659265295902  39.3180175079686  39.3377775686352  39.3589765524204  39.3750815702564  39.3863434241478  39.4072612884286  39.4270213490952  39.4418704693458  39.4535047326403  39.4744165352325  39.4916965410059  39.5114566016726  39.5271381241743  39.5427351334466  39.5624221816019  39.5827051243155  39.6024651849822  39.622344679465  39.635580929611  39.6461001904693  39.6689270046354  39.6864445645964  39.698512920539  39.7086364838842  39.7191168681467  39.7323378271006  39.7498553870616  39.7602853887091  39.7736034506451  39.7845901472249  39.7972864826791  39.8109189695172  39.8260541550621  39.838869017855  39.8507741346886  39.8596308613873  39.8732633482254  39.8853048237113  39.8939097268933  39.9064380576025  39.9202292704643  39.9338617573024  39.947126794935  39.9568277645577  39.9660261419174  39.978796649428  39.9924291362661  40.0044905501531  40.0194694386446  40.0305297016171  40.0395582039548  40.0531906907928  40.0660055535858  40.0793236155217  40.0913320051053  40.1050878009805  40.1187202878186  40.1262761730871  40.1367061747346  40.1487830672072  40.159597988148  40.1732304749861  40.1873392860123  40.200008662162  40.2127791696727  40.2260972316086  40.2397297184467  40.2485864451454  40.2612827805996  40.2712640834213  40.2842352432676  40.2978677301057  40.310895248833  40.3193246266626  40.3286418743613  40.3394553849992  40.3533355610122  40.3676293522928  40.3810213646451  40.3926615108903  40.4008861137304  40.4147662897434  40.4232392945182  40.4306196544343  40.4436471731616  40.4546169808528  40.4684971568659  40.4797564365729  40.490614302493  40.4992590829068  40.5139077433081  40.5277879193211  40.5392562899273  40.5529813790295  40.5639511867208  40.5743823857376  40.5882625617507  40.6014982770496  40.6101430574633  40.6198019558229  40.6269741191674  40.6408542951805  40.65247896284  40.6671276232413  40.6779411338792  40.6872583815779  40.701138557591  40.7095679354206  40.7226205572378  40.7360125695901  40.7468704355102  40.7607506115233  40.7720098912303  40.780482896005  40.7927093517453  40.8057368704726  40.8196170464857  40.8312417141452  40.8415117623279  40.8521147369837  40.8599348698424 |

Lampiran 4. Data Jumlah Death Node

|  |  |  |  |
| --- | --- | --- | --- |
| Iterasi | Jumlah Death Node  Algoritma K-Means | Iterasi | Jumlah Death Node  Algoritma K-Means++ |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399  400 | 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  4  4  5  5  6  6  6  6  6  7  7  7  7  7  7  8  8  8  8  9  9  9  11  11  11  12  12  13  14  15  16  16  18  18  18  18  18  18  20  20  20  20  20  20  20  21  21  22  22  25  25  26  26  26  28  28  30  30  30  31  31  32  33  33  33  33  37  39  40  42  42  43  43  44  47  47  47  48  48  48  48  49  49  49  49  50  50  51  51  51  51  52  52  52  52  52  53  54  55  55  57  58  58  58  58  58  58  58  59  59  59  59  61  62  62  62  62  62  62  62  62  62  62  62  62  62  62  63  63  63  63  64  65  65  66  66  66  66  67  67  67  67  67  67  67  67  67  67  67  67  67  68  68  68  68  68  68  68  69  69  69  71  72  72  72  72  73  73  73  73  73  73  73  73  73  73  73  74  74  74  74  74  74  74  75  75  75  75  75  75  75  75  75  76  77  77  77  77  77  77  77  77  77  77  77  77  77  77  77  77  77  77  78  79  79  79  79  80  80  80  80  80  80  80  80  80  80  80  80  80  80  80  81  81  81  81 | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399  400 | 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  2  2  3  3  4  4  5  5  6  8  8  8  8  10  11  12  12  13  14  15  15  15  16  19  19  19  19  20  22  22  23  23  23  23  24  24  24  24  24  25  26  26  27  29  29  29  29  30  30  30  31  32  33  34  34  34  34  34  34  34  35  35  36  36  36  36  38  39  39  41  42  42  42  42  42  43  43  44  45  45  46  46  47  47  48  48  48  48  48  50  50  50  50  51  52  53  53  53  53  53  53  53  53  53  53  53  53  53  53  53  55  55  55  55  55  55  55  55  55  55  56  56  56  56  56  56  57  57  57  58  58  58  58  58  58  58  58  58  58  58  58  58  58  58  58  59  59  59  60  60  60  60  60  60  60  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  61  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  62  63  63 |

Lampiran 5. Data Jumlah Node Alive

|  |  |  |  |
| --- | --- | --- | --- |
| Iterasi | Jumlah Node Alive  Algoritma K-Means | Iterasi | Jumlah Node Alive  Algoritma K-Means++ |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399  400 | 100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  99  99  98  98  98  98  98  98  98  98  98  98  98  98  98  98  98  98  98  96  96  95  95  94  94  94  94  94  93  93  93  93  93  93  92  92  92  92  91  91  91  89  89  89  88  88  87  86  85  84  84  82  82  82  82  82  82  80  80  80  80  80  80  80  79  79  78  78  75  75  74  74  74  72  72  70  70  70  69  69  68  67  67  67  67  63  61  60  58  58  57  57  56  53  53  53  52  52  52  52  51  51  51  51  50  50  49  49  49  49  48  48  48  48  48  47  46  45  45  43  42  42  42  42  42  42  42  41  41  41  41  39  38  38  38  38  38  38  38  38  38  38  38  38  38  38  37  37  37  37  36  35  35  34  34  34  34  33  33  33  33  33  33  33  33  33  33  33  33  33  32  32  32  32  32  32  32  31  31  31  29  28  28  28  28  27  27  27  27  27  27  27  27  27  27  27  26  26  26  26  26  26  26  25  25  25  25  25  25  25  25  25  24  23  23  23  23  23  23  23  23  23  23  23  23  23  23  23  23  23  23  22  21  21  21  21  20  20  20  20  20  20  20  20  20  20  20  20  20  20  20  19  19  19  19 | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399  400 | 100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  100  99  99  99  99  99  99  99  99  99  99  99  99  99  99  99  99  99  99  99  98  98  98  98  98  98  98  97  97  96  96  95  95  94  92  92  92  92  90  89  88  88  87  86  85  85  85  84  81  81  81  81  80  78  78  77  77  77  77  76  76  76  76  76  75  74  74  73  71  71  71  71  70  70  70  69  68  67  66  66  66  66  66  66  66  65  65  64  64  64  64  62  61  61  59  58  58  58  58  58  57  57  56  55  55  54  54  53  53  52  52  52  52  52  50  50  50  50  49  48  47  47  47  47  47  47  47  47  47  47  47  47  47  47  47  45  45  45  45  45  45  45  45  45  45  44  44  44  44  44  44  43  43  43  42  42  42  42  42  42  42  42  42  42  42  42  42  42  42  42  42  42  42  42  41  41  41  40  40  40  40  40  40  40  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  39  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  38  37  37 |