# 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

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

*Wireless Sensor Network* merupakan sekumpulan node yang dapat memiliki kemampuan *sensing, processing data* dan *wireless communication*. Masing-masing *node* menggunakan baterai sebagai sumber energinya, sehingga penggunaan energi menjadi salah satu point penting dalam membangun WSN. Routing protokol LEACH merupakan salah satu solusi untuk mengatasi permasalahan energi pada WSN, namun LEACH memiliki beberapa kekurangan yaitu tidak mempertimbangkan sisa energi jarak *cluster head* pada *base station.*

Berdasarkan permasalahan tersebut pada penelitian ini menggunakan algoritma pengembangan *K-Means* yaitu *K-Means++* untuk memodifikasi LEACH menjadi KM-LEACH untuk lebih mengoptimalkan pemilihan *cluster head* yang akan dipilih berdasarkan energi yang tersisa. Algoritma ini akan memodifikasi fase *setup phase* pada protokol LEACH dan membandingkan dengan penelitian sebelumnya yang menggunakan algoritma *K-Means*.

Tahap pengujian dalam penelitian ini dilakukan dengan membandingkan 3 parameter uji yaitu total konsumsi energi (joule), jumlah *dead node* dan *node alive* yang dibandingkan dengan penelitian sebelumnya dengan menggunakan algoritma *K-Means.* Hasil yang didapat menunjukan bahwa KM-LEACH lebih hemat energi yaitu dengan total konsumsi energi 40.86 joule berbanding 48.40 joule dan jumlah *dead node* yang dihasilkan lebih sedikit yaitu 63 node berbanding 81 *node* dengan LEACH dengan algoritma *K-Means*.



**Kata Kunci:** *Wireless Sensor Network, K-Means++ clustering*, LEACH Protokol, Efisiensi Energi, Routing Protokol

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

Wireless Sensor Network is a group of nodes that can have sensing, data processing and wireless communication capabilities. Each node uses a battery as its energy source, so energy use is an important point in building a WSN. LEACH protocol routing is one solution to overcome energy problems in WSN, but LEACH has several disadvantages, namely it does not consider the remaining energy distance of the cluster head at the base station.

Based on these problems, this study uses the *K-Means* development algorithm, namely *K-Means* ++, to modify LEACH to KM-LEACH to further optimize the selection of the cluster head to be selected based on the remaining energy. This algorithm will modify the setup phase of the LEACH protocol and compare it with previous studies using the *K-Means* algorithm.

The testing phase in this study was carried out by comparing 3 test parameters, namely the total energy consumption (joules), the number of dead nodes and nodes alive which were compared with previous studies using the *K-Means* algorithm. The results obtained show that KM-LEACH is more energy efficient, with a total energy consumption of 40.86 joules versus 48.40 joules and the resulting number of dead nodes is less, namely 63 nodes compared to 81 nodes with LEACH with the *K-Means* algorithm.

**Keywords :** *Wireless Sensor Network, K-Means++ Clustering,* LEACH *Protocol, Energy Efficiency, Routing Protocol*

# 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

## DAFTAR TABEL

## DAFTAR GAMBAR

## DAFTAR LAMPIRAN

# BAB I
# PENDAHULUAN

## 1.1 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).

## 1.2 Rumusan Masalah

Rumusan masalah yang akan dijadikan acuan dalam penelitian ini yaitu:

a) Bagaimana pengaruh penerapan KM-LEACH terhadap efisiensi penggunaan energi (total konsumsi energi/Joule) pada WSN
b) 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.

## 1.3 Batasan Masalah

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

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

## 1.4 Tujuan Penelitian

Adapun tujuan dari dilakukannya penelitian ini yaitu:

a) Menerapkan algoritma *K-Means++ Clustering* sebagai optimalisasi dari Protokol LEACH serta mengetahui pengaruhnya terhadap efisiensi penggunaan energi (total konsumsi energi/Joule) pada WSN
b) 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.

## 1.5 Manfaat Penelitian

Manfaat yang diperoleh dari penelitian ini yaitu:

a) 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.*
b) 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.

## 1.6 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.

### 1.6.1 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.

### 1.6.2 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:

a. Total konsumsi energi (Joule)
b. Jumlah *node* hidup selama 400 round
c. Jumlah *dead 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 $m^2$untuk membuat skema topologi jaringan WSN.

### 1.6.3 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.

### 1.6.4 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.

### 1.6.5 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.

## 2.1 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:

### 2.1.1 *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 *node* 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 $m^2$. 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++*.

### 2.1.2 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.

### 2.1.3 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 $m^2$.

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++*.

### 2.1.4 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.

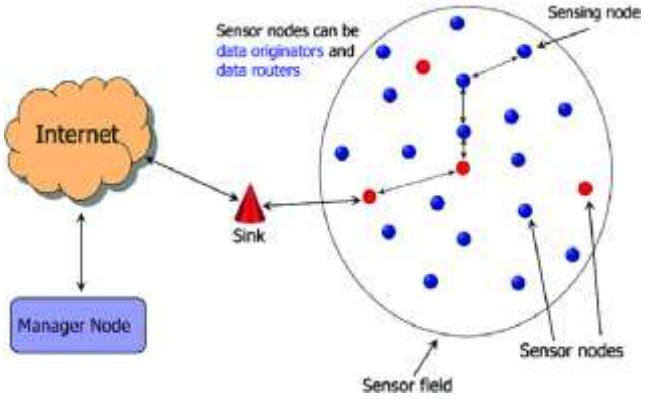
## 2.2 Tinjauan Teoritis

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

### 2.2.1 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.

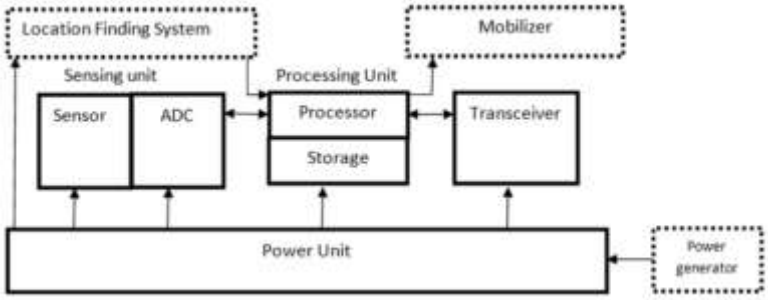



Gambar 2.1 Skema WSN

(Sumber: Mihobi dkk, 2017)

### 2.2.2 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):

A. *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.

B. 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.

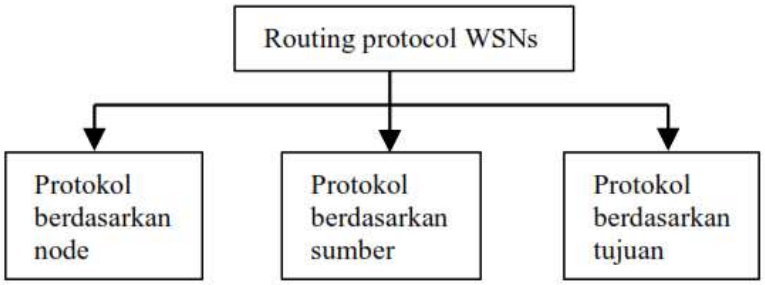
C. *Sink Node*

*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.

### 2.2.3 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)

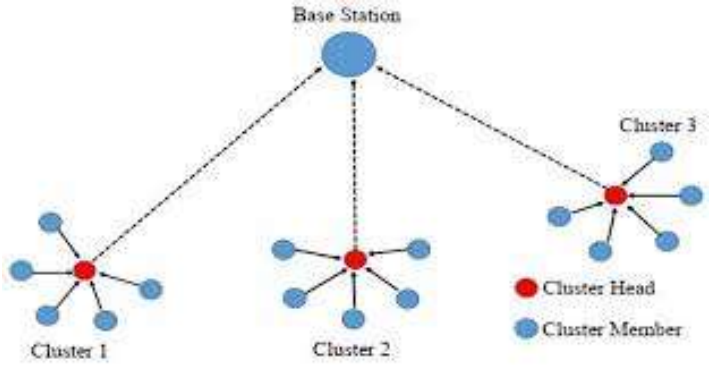
Pada WSN *routing protocol* dapat dibedakan menjadi 3 jenis yaitu *routing protocol* berdasarkan *node*, *routing protocol* berdasarkan sumber dan *routing protocol* berdasarkan tujuan (Novid dkk, 2019). Protokol berdasarkan *node* yang dimaksud adalah dalam penerapan routing protokol *node-node* yang 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).

### 2.2.4 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*.

### 2.2.5 Fitur-Fitur LEACH Protokol

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

A. *Data fusion*, yaitu penggabungan data sehingga mengurangi disipasi energi dan menambah lifetime jaringan.
B. *Adaptive*, yaitu mudah menyesuaikan diri saat pembentukan formasi klaster.
C. *Local Compression*, yaitu mengkompres data agar ukuran data yang dikirim ke *base station* lebih kecil.

D. *Randomization Rotation*, yaitu perputaran kedudukan *cluster head* secara acak.

E. *Self-Organizing*, yaitu tiap *node sensor* memiliki sikap pengambilan keputusan sendiri untuk menjadi *cluster head*.

### 2.2.6 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:

A. 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:

$$\boldsymbol{T(n)} = \begin{cases} \dfrac{\boldsymbol{P}}{\boldsymbol{1 - p * (r \, mod \dfrac{1}{p}}}, & \boldsymbol{if \; n \in G} \\ \boldsymbol{0}, & \boldsymbol{otherwise} \end{cases} \qquad (1)$$

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 :

$$\boldsymbol{Er(n) = E - Te} \qquad (2)$$

Dimana:

$\boldsymbol{Er}$ = Residual energi setiap *node* selama *round*

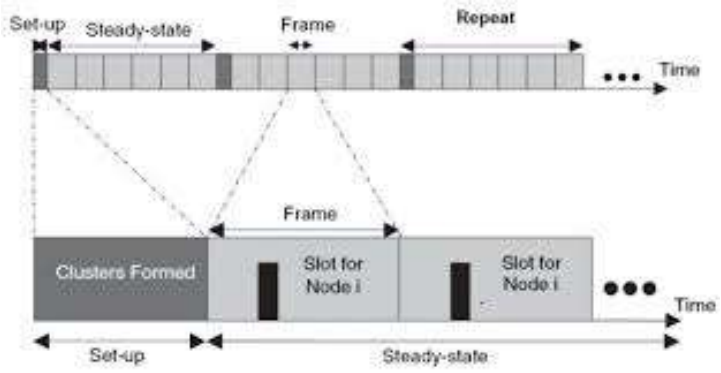
$\boldsymbol{Ei}$ = Energi setiap *node* selama *round*

$\boldsymbol{T_e}$ = 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*.

B. 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:



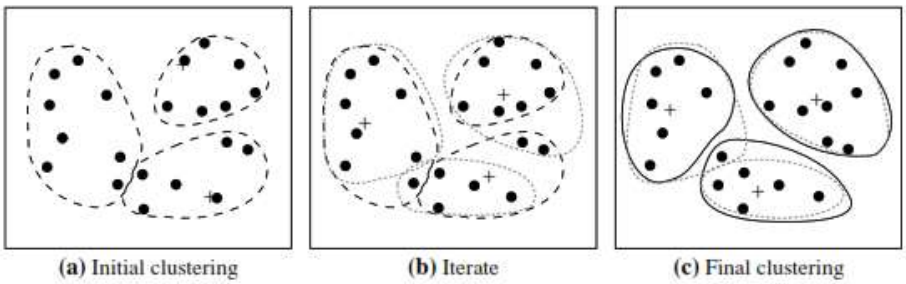

Gambar 2.5 Fase Pada LEACH

(Sumber: Ridwan dkk, 2020)

### 2.2.7 *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):

$$\boldsymbol{J = \sum_{i=1}^{k} \sum_{j=1}^{n} \left\| X_j - C_i \right\|^2} \qquad (3)$$

Dimana:

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

$X_j^{\,i}$ = *node* ke-j pada klaster ke-i

$C_i$ = *centroid* ke-i pada klaster ke-i

Berikut adalah Langkah-langkah algoritma *K-Means* (M. Ni'man Nasir & Irwan Budiman, 2017):

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

$$\boldsymbol{Ci} = \frac{\mathbf{1}}{|\boldsymbol{ki}|} \sum_{\boldsymbol{X_{j\in k}}}^{\boldsymbol{k}} \boldsymbol{Xj} \qquad (4)$$

G. 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 $m^2$ 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 (3) seperti berikut:

Jarak $node1$ dengan c1 $= \sqrt{(4-5)^2+(3-4)^2} = 1.414$ m

Jarak $node1$ dengan c2 $= \sqrt{(4-8)^2+(3-6)^2} = 5$ m

Jarak $node2$ dengan c1 $= \sqrt{(5-5)^2+(4-4)^2} = 0$ m

Jarak $node2$ dengan c2 $= \sqrt{(5-8)^2+(4-6)^2} = 3.605$ m

Jarak $node3$ dengan c1 $= \sqrt{(6-5)^2+(7-4)^2} = 3.162$ m

Jarak $node3$ dengan c2 $= \sqrt{(6-8)^2+(7-6)^2} = 2.236$ m

Jarak $node4$ dengan c1 $= \sqrt{(7-5)^2+(8-4)^2} = 4.472$ m

Jarak $node4$ dengan c2 $= \sqrt{(7-8)^2+(8-6)^2} = 2.236$ m

Jarak $node5$ dengan c1 $= \sqrt{(8-5)^2+(6-4)^2} = 3.605\ m$

Jarak $node5$ dengan c2 $= \sqrt{(8-8)^2+(6-6)^2} = 0$ m

Table 2.2 Perhitungan *Centroid* Terdekat Antar *Node*

| *Node* ke (i) | Jarak ke c1 | Jarak ke c2 | Jarak terdekat | Klaster |
|---|---|---|---|---|
| 1 | 1.414 | 5 | 1.414 | c1 |
| 2 | 0 | 3.605 | 0 | c1 |
| 3 | 3.162 | 2.236 | 2.236 | c2 |
| 4 | 4.472 | 2.236 | 2.236 | c2 |
| 5 | 3.605 | 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.

$c1\ untuk\ nilai\ x = \frac{(4+5)}{2} = 4.5 \qquad c1\ untuk\ nilai\ y = \frac{(4+3)}{2} = 3.5$

$c2\ untuk\ nilai\ x = \frac{(6+7+8)}{3} = 7 \qquad c2\ untuk\ nilai\ y = \frac{(7+8+6)}{3} = 7$

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.

### 2.2.8 *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++:*

$$\boldsymbol{K} = \frac{\boldsymbol{D}(\boldsymbol{x})}{\sum_{\boldsymbol{x}\epsilon\boldsymbol{X}} \boldsymbol{D}(\boldsymbol{x})} \qquad (5)$$

$\boldsymbol{D}(\boldsymbol{x})$ = Jarak *euclidean distance*

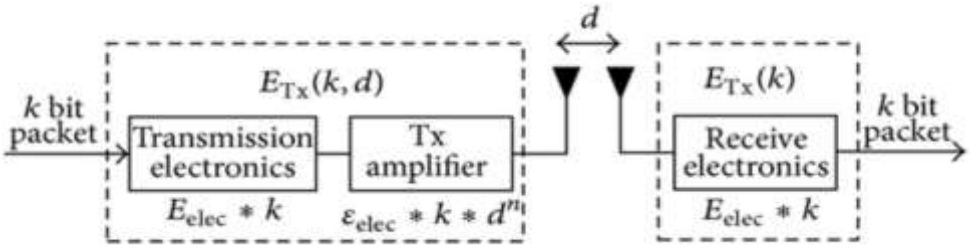
$\sum_{\boldsymbol{x}\epsilon\boldsymbol{X}} \boldsymbol{D}(\boldsymbol{x})$ = Jumlah jarak *euclidean distance*

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

A. Tentukan titik *centroid* pertama secara acak dari pada titik data *node.*
B. Untuk setiap *node* yang diamati, hitung jarak D(x) dengan *centroid* terdekat dengan rumus persamaan **Error! Reference source not found.**.
C. Pilih *centroid* baru dari antara titik *node*. dengan probabilitas x dipilih proporsional dengan rumus persamaan **Error! Reference source not found.**, dimana posisi *node* dengan nilai K terbesar dipilih sebagai *centroid.*
D. Ulangi langkah B dan C hingga jumlah *centorid* sama dengan jumlah klaster yang ditentukan.
E. Lanjut ke proses algoritma *K-Means* biasa.

### 2.2.9 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)

### 2.2.10 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

## 3.1 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:

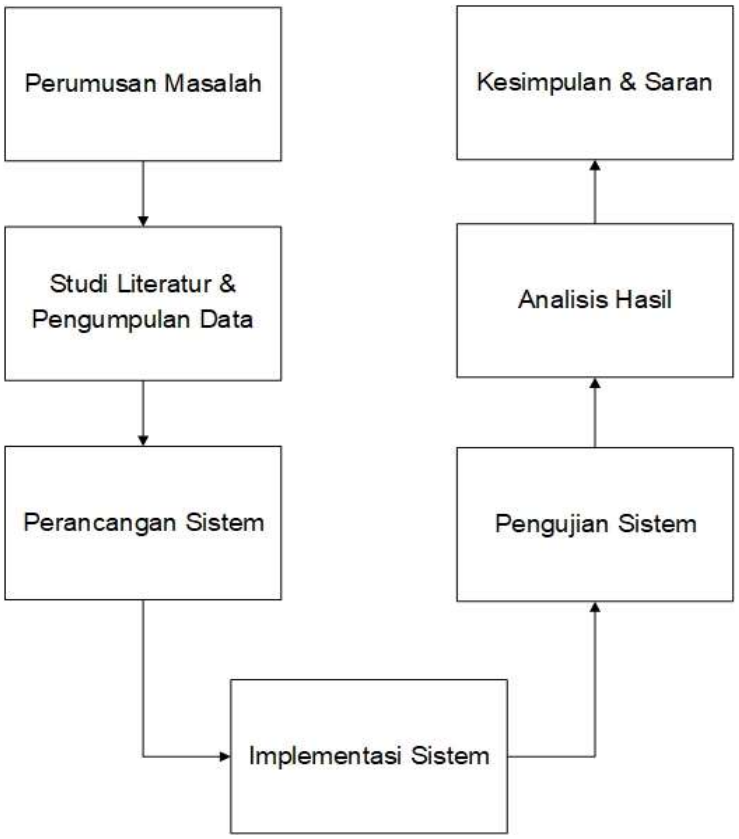
a. Total konsumsi energi (Joule)
b. Jumlah *node* hidup selama 400 round
c. Jumlah *dead node* selama 400 round

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 $m^2$ 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.

## 3.2 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.

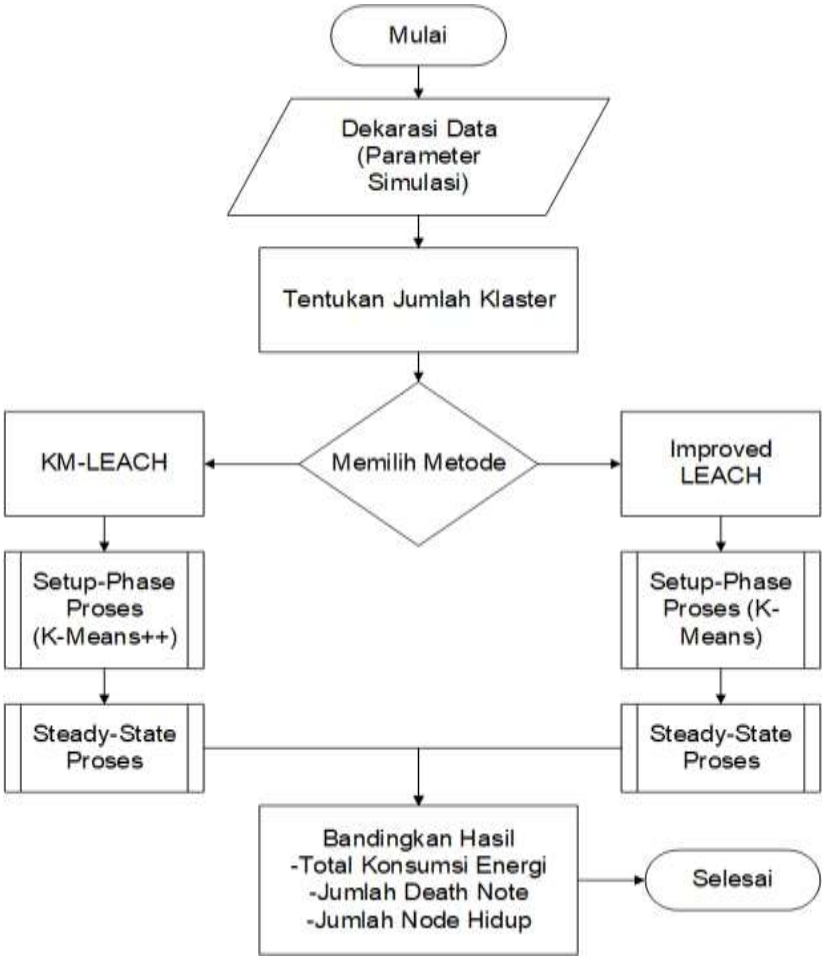
Barulah pada tahap keempat dilakukan implementasi sistem yang mana implementasi sistem akan dilakukan dalam bentuk simulasi dan kemudian dilakukan pengujian serta analisis sistem untuk mengetahui hasil akhir penelitian yang dilakukan, untuk mengetahui apakah penelitian yang dilakukan berhasil atau

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++.

## 3.3 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:

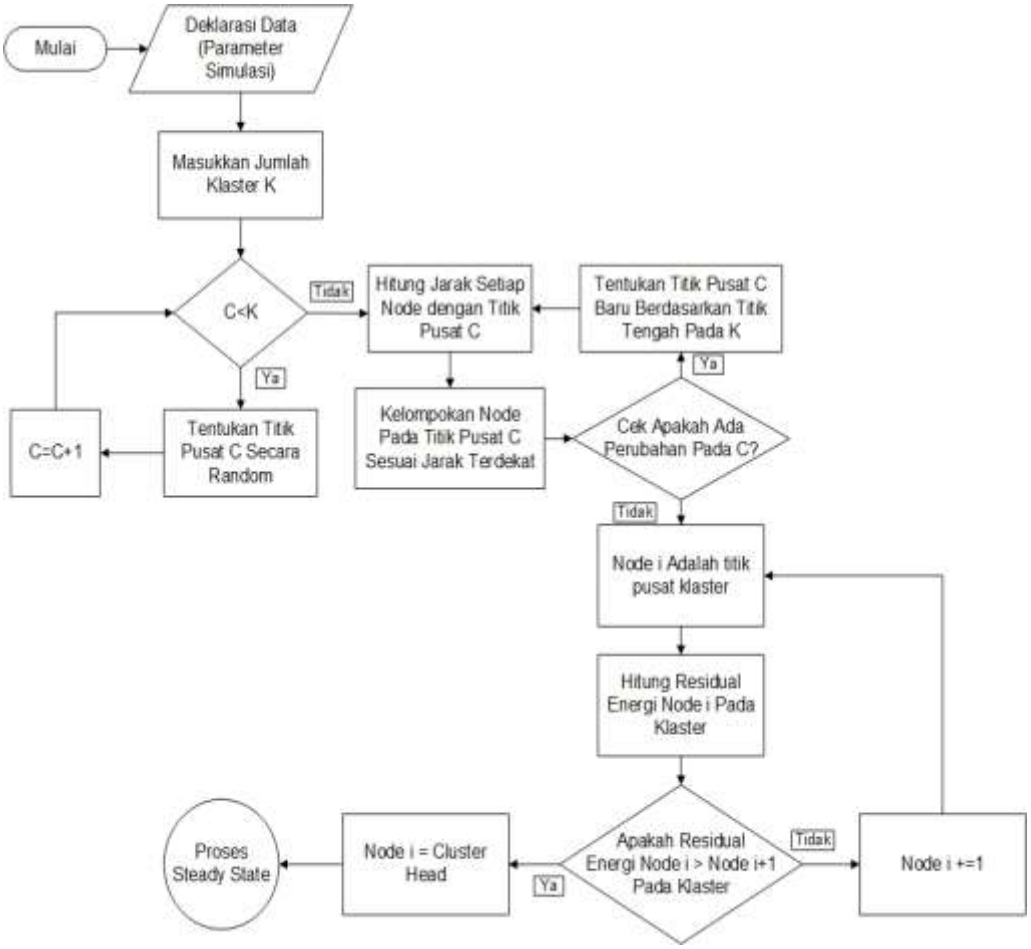



Gambar 3.2 Flowchart Rancangan Program

## 3.4 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:

A. *Setup Phase*



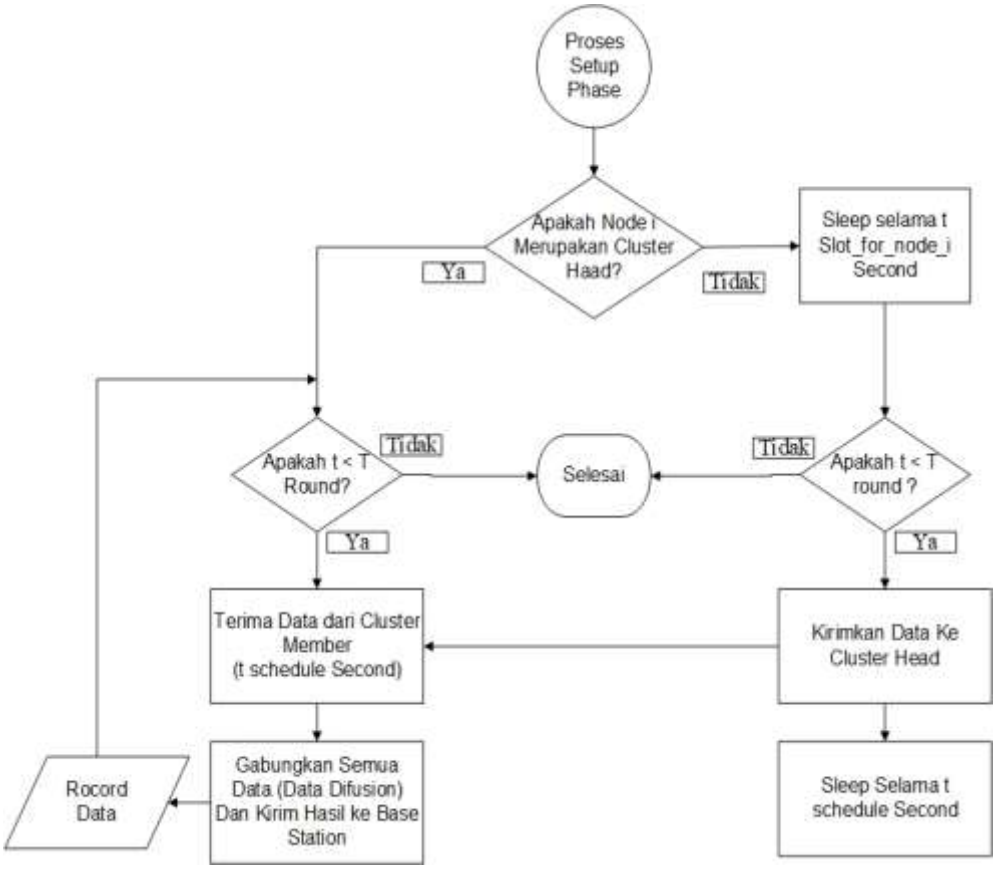

Gambar 3.3 *Flowchart Setup Phase* LEACH Dengan *K-Means*

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 *node* 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*

B. 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*:

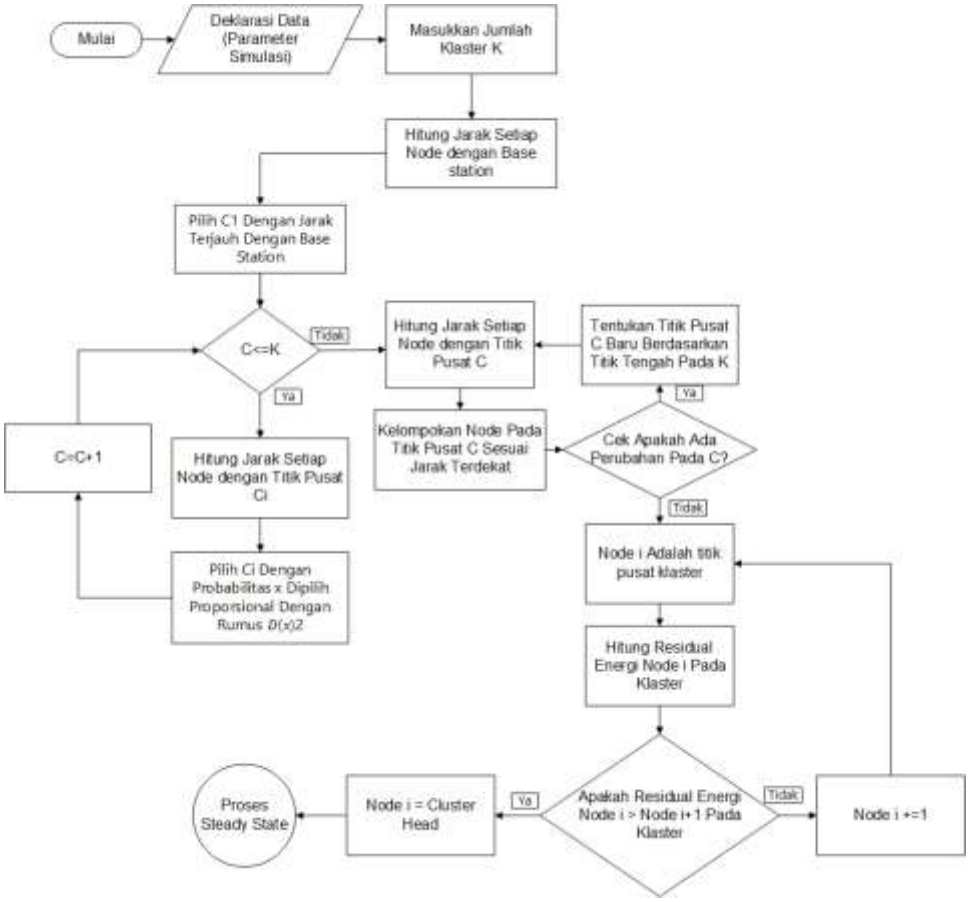
1. Cek apakah *node* merupakan sebuah *cluster head*
2. Apabila merupakan *cluster head* maka, *cluster head* akan menerima data dari semua anggota klaster.
3. *Node* yang bukan *cluster head* hanya akan mengirim data sesuai dengan jadwal TDMA yang sudah diterima.
4. *Node cluster head* akan mengumpulkan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

## 3.5 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:

A. 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 *node* 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*

## 3.6 Implementasi

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

A. 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

B. 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

### 3.7 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 $m^2$ |
| 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 |
| $\varepsilon$mp | 13nJ/bit/$m^4$ |
| $\varepsilon fs$ | 10nJ/bit/$m^2$ |
| 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.

### 3.8 Pengukuran dan Hasil Simulasi

Kualitas kinerja dari protokol KM-LEACH dengan protokol LEACH akan diukur dengan beberapa parameter seperti rata-rata konsumsi energi, banyaknya *dead 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:

$$\boldsymbol{E_{TX}(k,d) = (E_{elec} \times k) + \varepsilon_{fs} + k \times d^2} \qquad (6)$$

Dimana penjelas dari variabel rumus diatas yaitu:

$\boldsymbol{d}$ = *Threshold Distance,*

$\boldsymbol{E_{elec}}$ = *Electronic Energy*

$\boldsymbol{\varepsilon_{fs}}$ = *Parameter for Free Space Model*

$\boldsymbol{\varepsilon_{mp}}$ = *Parameter for Multi Path Model*

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

$$\boldsymbol{E_{RX} = E_{elec} * k} \qquad (7)$$

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

***Dead 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 *Dead 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

## 4.1. Implementasi Sistem

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

### 4.1.1 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');
```

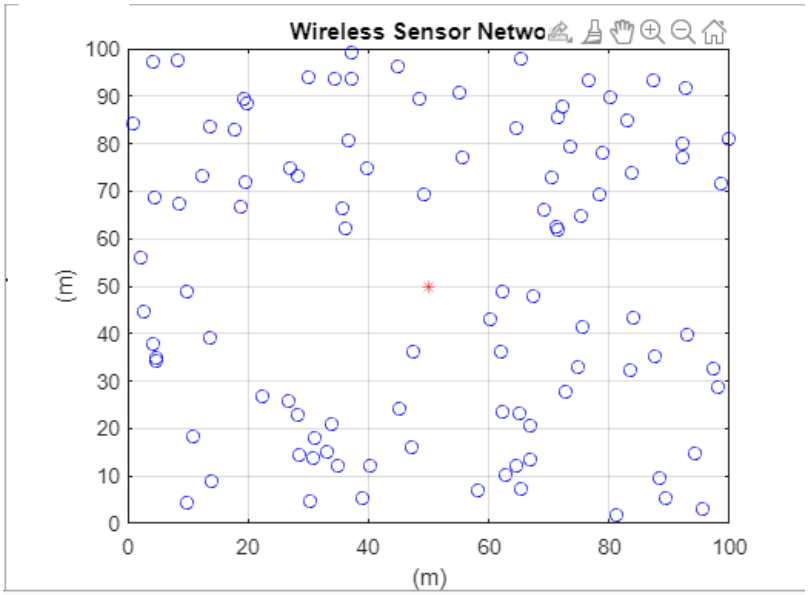
Pada tabel 4.1 diatas dilakukan beberapa pendeklarasian variabel seperti pemetaan sensor dalam satuan meter, variabel energi, penentuan klaster dan round dan proses pembacaan data titik koordinat *node*, dimana data *node* ini diambil dari file dengan nama *node*.xlsx yang sudah kita siapkan sebelumnya. Kemudian dilakukan proses pembentukan topologi jaringan awal yang *source code*nya 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 $m^2$ dan untuk posisi sink dilambangkan dengan symbol plus merah yang dapat dilihat pada Gambar 4.1 berikut.




Gambar 4.1 Topologi WSN

### 4.1.2 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:

A. 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
```

B. 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);
     cy=center(:,2);
     mean_oldx=cx;
     mean_newx=cx;
     mean_oldy=cy;
     mean_newy=cy;
     outputx=cell(k,1);
     outputy=cell(k,1);
     temp=0;
     iter=1;
     %Perulangan pembentukan klaster hingga titik pusat tidak
      mengalami perubahan
     while(temp==0)
           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;
                SN(i).cluster=index;
                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;
```

C. 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);
```

D. 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
```

### 4.1.3 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.

A. 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];
      d=[];
      for i=1:n
          SN(i).cluster=0;
          SN(i).role=0;
          SN(i).chid=0;
          c=[SN(i).x SN(i).y];
         distance=sqrt((sinkx-c(:,1))^2+(sinky-c(:,2))^2);
          d=[d distance];
      end
      [e s]=max(d);
      center=[center;[x(s) y(s)]];
      x(s)=[];
      y(s)=[];
      d=[];
      %Proses mencari C selanjutnya
      r=1;
      while(r~=k)
           for i=1:n
               g=[SN(i).x SN(i).y];
               ka=dsearchn(center,g);
               nearestx=center(ka,1);
               nearesty=center(ka,2);
               distance=sqrt((nearestx-
g(:,1))^2+(nearesty-g(:,2))^2);
               d=[d distance];
```

```
            end
            [e s]=max(d);
            center=[center;[x(s) y(s)]];
            SN(s).tel=SN(s).tel+1;
            x(s)=[];
            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 **Error! Reference source not found.**, 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.

B. 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);
      cy=center(:,2);
      mean_oldx=cx;
      mean_newx=cx;
      mean_oldy=cy;
      mean_newy=cy;
      outputx=cell(k,1);
      outputy=cell(k,1);
      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;
                 SN(i).cluster=index;
                 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;
            cy=gmcky;
            mean_newx=cx;
            mean_newy=cy;
```

C. 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);
```

D. 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
```

### 4.1.4 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.

A. 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
```

B. 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 *dead_node* ditambahkan 1 dan operating_*node*s 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
```

C. 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. Dilakukan juga perhitungan rata-rata iterasi yang dicapai dengan menjumlahkan variabel jml_iter setiap *round* dibagi jumlah putaran yang dilakukan, yang disimpan dalam variabel rata. Berikut adalah *source code* proses perhitungan total energi, *Dead node* & *node Alive* pada tabel 4.13 berikut.

Table 4.13 Perhitungan Total Energi, *Dead Node* & *Node Alive*

```
%Kalkulasi total energi, dead 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
% Menghitung rata-rata iterasi yang dilakukan
rata=sum(jml_iter)/400;
disp('Rata-rata iterasi yang dilakukan yaitu');disp(rata)
% Save data dalam bentuk file
writematrix(total_energi,'TE-kmeansplus');
writematrix(total_dn,'TDN-kmeansplus');
writematrix(total_na,'TNA-kmeansplus');
```

## 4.2. 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.

### 4.2.1 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.

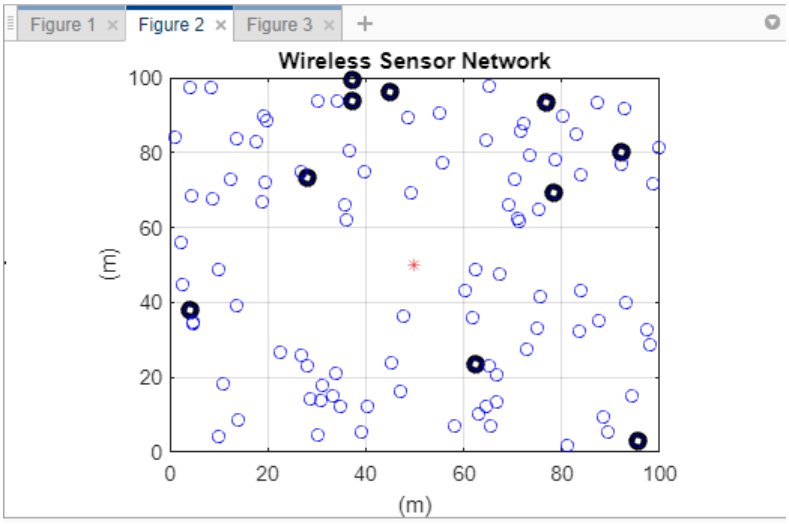
A. Ploting Titik Pusat Klaster Algoritma *K-Means*

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*

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

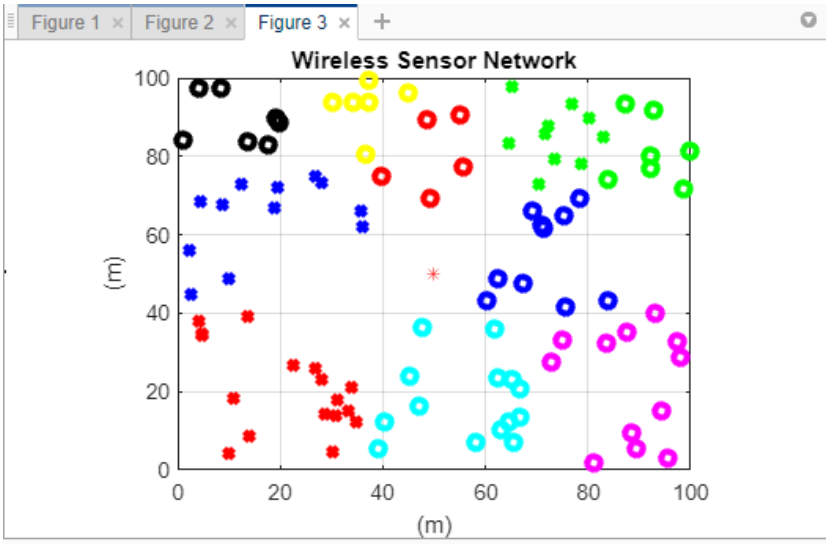
Pada proses ini setelah titik pusat klaster dipilih dan masing-masing *node* dikelompokkan hingga titik pusat klaster tidak mengalami perubahan maka akan didapatkan hasil akhir klaster *node* yang dibuat sesuai dengan jumlah klaster yang ditentukan. Data *node* tiap klasternya disimpan dalam variabel 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 dengan rata-rata iterasi yang didapat selama proses pembentukan klaster setiap roundnya sebanyak 8.35 kali.




Gambar 4.3 Plotting Hasil *Clustering* Dengan Algoritma *K-Means*

### 4.2.2 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.

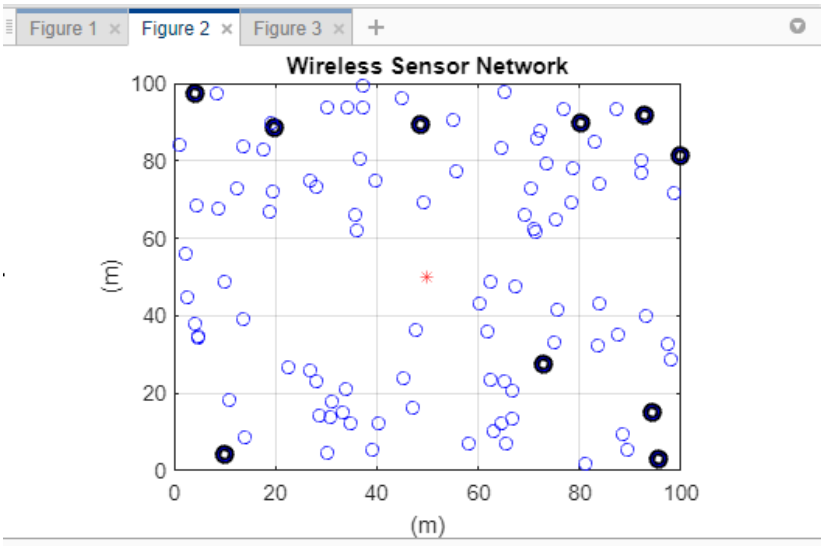
A. Ploting Titik Pusat Klaster Algoritma *K-Means++*

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++*

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

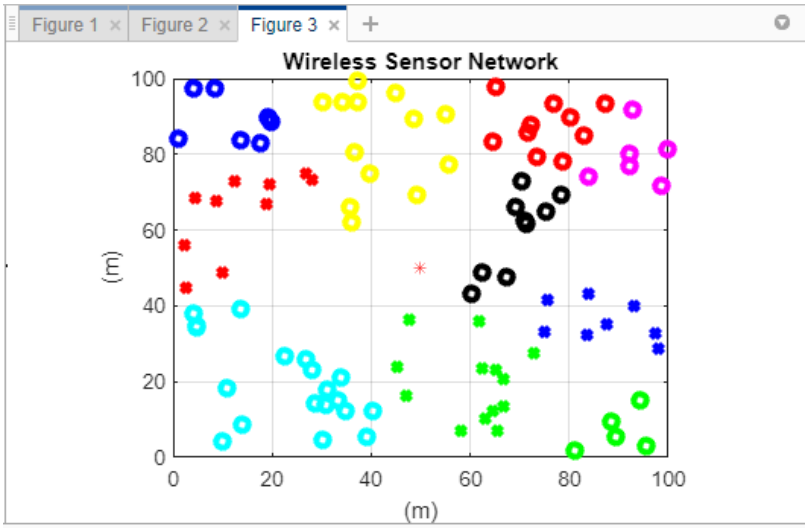
Pada proses ini setelah titik pusat klaster dipilih dan masing-masing *node* dikelompokkan hingga titik pusat klaster tidak mengalami perubahan maka akan didapatkan hasil akhir klaster *node* yang dibuat sesuai dengan jumlah klaster yang ditentukan. Data *node* tiap klasternya disimpan dalam variabel 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 dengan rata-rata iterasi yang didapat selama proses pembentukan klaster setiap roundnya sebanyak 14 kali.



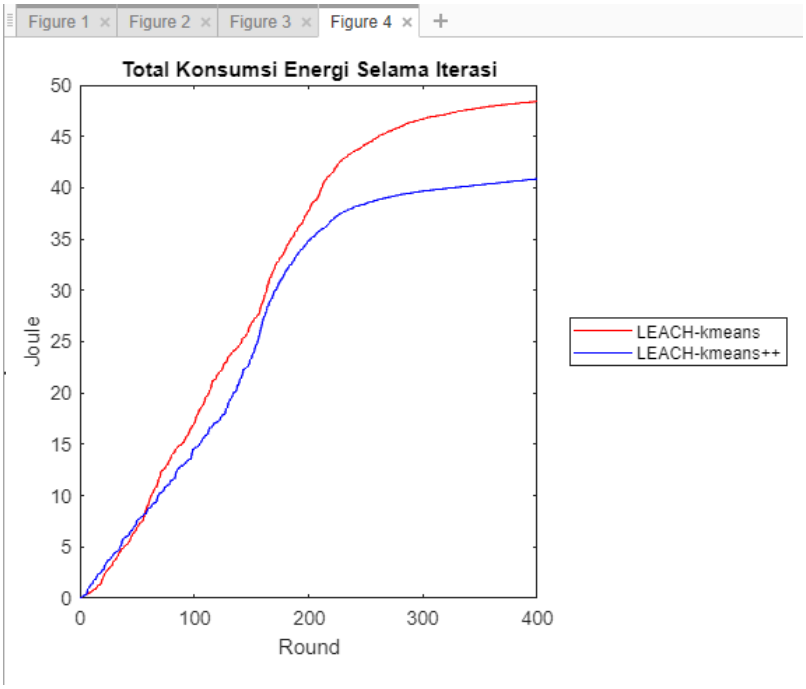

Gambar 4.5 Plotting Hasil *Clustering* Dengan Algoritma *K-Means++*

### 4.2.3 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.



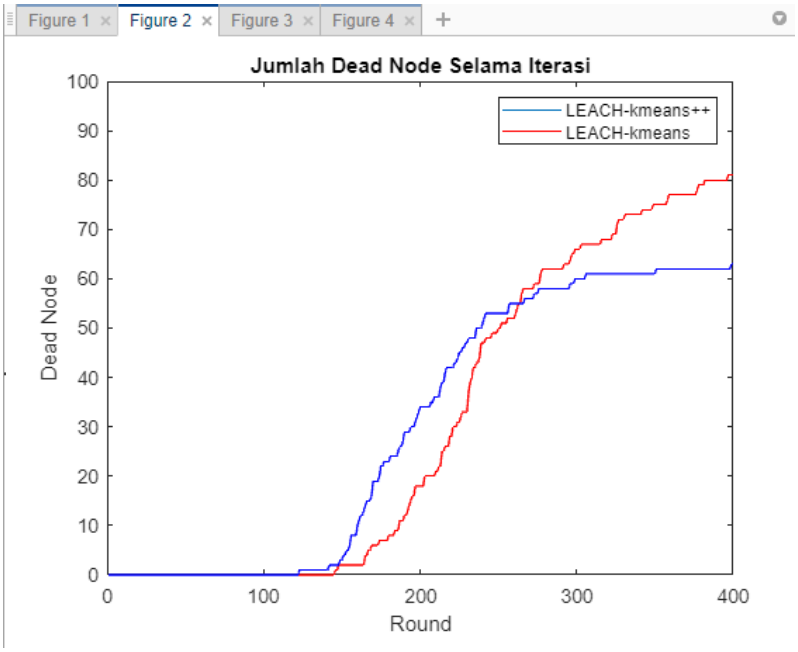

Gambar 4.6 Hasil Perbandingan Total Konsumsi Energi

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.

### 4.2.4 Pengujian Hasil Jumlah *Dead Node*

*Dead 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 *dead node* seperti gambar 4.7 berikut.



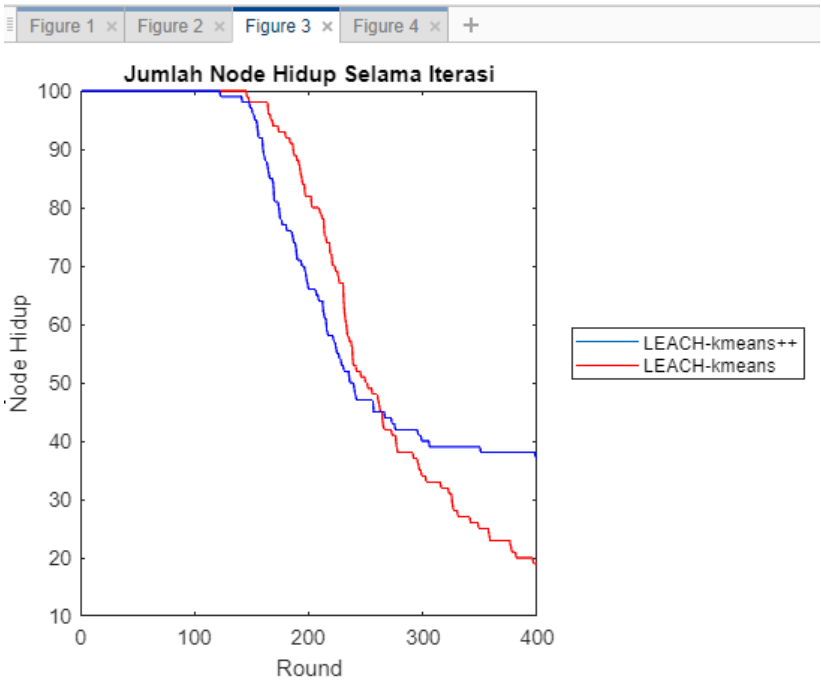

Gambar 4.7 Hasil Perbandingan Jumlah *Dead Node*

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 *dead node* sebanyak 63 *node*. Lebih baik dibandingkan LEACH dengan algoritma *K-Means* yang memiliki jumlah *dead node* sebanyak 81 *node*.

### 4.2.5 Pengujian Hasil Jumlah *Node* Alive

*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 *dead node* sebanyak 19 *node*.

# BAB V
# PENUTUP

## 5.1 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 *dead node* yang didapat juga menunjukkan bahwa KM-LEACH dengan algoritma *K-Means++* yang memiliki total konsumsi energi yang lebih efisien dapat mengurangi jumlah *dead 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.

## 5.2 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++* masih kurang 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 | 78.287 | 69.379 |
| 2 | 0.98023 | 84.321 |
| 3 | 92.233 | 77.095 |
| 4 | 4.266 | 37.819 |
| 5 | 70.434 | 72.951 |
| 6 | 22.428 | 26.905 |
| 7 | 67.303 | 47.749 |
| 8 | 62.372 | 23.644 |
| 9 | 17.712 | 82.964 |
| 10 | 76.692 | 93.448 |
| 11 | 10.789 | 18.223 |
| 12 | 9.9095 | 48.976 |
| 13 | 19.325 | 89.589 |
| 14 | 9.909 | 4.4166 |
| 15 | 55.73 | 77.25 |
| 16 | 31.194 | 17.898 |
| 17 | 33.896 | 21.015 |
| 18 | 55.015 | 90.636 |
| 19 | 62.892 | 10.153 |
| 20 | 39.085 | 5.4617 |
| 21 | 60.128 | 43.172 |
| 22 | 99.756 | 81.16 |
| 23 | 48.565 | 89.445 |
| 24 | 13.755 | 39 |
| 25 | 92.736 | 91.749 |
| 26 | 71.357 | 61.834 |

| | | |
|---|---|---|
| 27 | 34.329 | 93.603 |
| 28 | 12.477 | 73.059 |
| 29 | 64.648 | 83.315 |
| 30 | 39.828 | 74.982 |
| 31 | 83.522 | 32.246 |
| 32 | 65.226 | 97.913 |
| 33 | 74.931 | 33.042 |
| 34 | 61.947 | 36.064 |
| 35 | 75.651 | 41.39 |
| 36 | 49.235 | 69.474 |
| 37 | 97.273 | 32.775 |
| 38 | 83.78 | 73.907 |
| 39 | 95.417 | 3.1923 |
| 40 | 35.687 | 66.265 |
| 41 | 28.15 | 23.038 |
| 42 | 71.113 | 62.457 |
| 43 | 69.061 | 66.044 |
| 44 | 4.7555 | 34.878 |
| 45 | 45.134 | 24.09 |
| 46 | 71.505 | 85.618 |
| 47 | 28.151 | 73.105 |
| 48 | 13.776 | 83.672 |
| 49 | 13.86 | 8.821 |
| 50 | 36.616 | 80.676 |
| 51 | 62.378 | 48.959 |
| 52 | 87.705 | 35.314 |
| 53 | 44.944 | 96.353 |
| 54 | 4.2298 | 97.296 |
| 55 | 18.921 | 66.712 |
| 56 | 8.644 | 67.511 |
| 57 | 36.102 | 62.028 |

| | | |
|---|---|---|
| 58 | 81.115 | 1.9257 |
| 59 | 8.3874 | 97.48 |
| 60 | 65.135 | 23.124 |
| 61 | 40.349 | 12.202 |
| 62 | 26.844 | 25.785 |
| 63 | 33.167 | 15.223 |
| 64 | 34.801 | 12.166 |
| 65 | 88.415 | 9.4278 |
| 66 | 93.004 | 39.902 |
| 67 | 4.7401 | 34.237 |
| 68 | 73.597 | 79.468 |
| 69 | 4.491 | 68.622 |
| 70 | 89.363 | 5.4792 |
| 71 | 30.366 | 4.6192 |
| 72 | 19.548 | 72.017 |
| 73 | 72.175 | 87.78 |
| 74 | 58.243 | 7.0684 |
| 75 | 92.274 | 80.037 |
| 76 | 28.595 | 14.366 |
| 77 | 98.478 | 71.568 |
| 78 | 83.897 | 43.326 |
| 79 | 47.062 | 16.071 |
| 80 | 26.909 | 74.902 |
| 81 | 75.389 | 64.681 |
| 82 | 30.775 | 13.872 |
| 83 | 47.557 | 36.246 |
| 84 | 78.811 | 78.03 |
| 85 | 66.851 | 13.35 |
| 86 | 2.1556 | 55.984 |
| 87 | 30.082 | 93.941 |
| 88 | 98.09 | 28.662 |

| | | |
|---|---|---|
| 89 | 80.082 | 89.611 |
| 90 | 19.753 | 88.402 |
| 91 | 94.373 | 14.916 |
| 92 | 72.839 | 27.676 |
| 93 | 2.5857 | 44.653 |
| 94 | 64.63 | 12.12 |
| 95 | 37.231 | 93.713 |
| 96 | 82.953 | 84.909 |
| 97 | 37.253 | 99.318 |
| 98 | 87.255 | 93.35 |
| 99 | 66.846 | 20.678 |
| 100 | 65.385 | 7.2052 |

Lampiran 2. Data Bilangan Acak

| No Round | Klaster | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 99 | 35 | 93 | 54 | 21 | 76 | 73 | 17 | 90 | 87 |
| 2 | 15 | 50 | 29 | 6 | 74 | 2 | 70 | 31 | 7 | 1 |
| 3 | 78 | 10 | 99 | 74 | 80 | 62 | 76 | 36 | 66 | 97 |
| 4 | 89 | 31 | 57 | 65 | 72 | 76 | 82 | 56 | 13 | 64 |
| 5 | 29 | 7 | 73 | 36 | 61 | 24 | 14 | 51 | 11 | 78 |
| 6 | 49 | 51 | 83 | 29 | 78 | 73 | 70 | 8 | 40 | 77 |
| 7 | 52 | 18 | 26 | 86 | 29 | 1 | 32 | 88 | 73 | 19 |
| 8 | 59 | 11 | 41 | 16 | 12 | 99 | 98 | 91 | 85 | 71 |
| 9 | 64 | 23 | 36 | 58 | 69 | 3 | 83 | 22 | 99 | 6 |
| 10 | 52 | 70 | 55 | 58 | 83 | 57 | 51 | 91 | 8 | 95 |
| 11 | 21 | 57 | 41 | 23 | 49 | 98 | 34 | 9 | 39 | 18 |
| 12 | 82 | 51 | 70 | 89 | 45 | 68 | 43 | 38 | 78 | 72 |
| 13 | 83 | 28 | 60 | 54 | 94 | 78 | 37 | 45 | 36 | 39 |
| 14 | 68 | 46 | 10 | 86 | 19 | 74 | 87 | 26 | 79 | 96 |
| 15 | 19 | 99 | 67 | 82 | 96 | 73 | 36 | 65 | 61 | 90 |
| 16 | 95 | 91 | 46 | 17 | 1 | 10 | 84 | 98 | 31 | 11 |
| 17 | 98 | 93 | 29 | 16 | 66 | 98 | 40 | 95 | 6 | 34 |
| 18 | 7 | 46 | 75 | 48 | 37 | 43 | 36 | 42 | 4 | 66 |
| 19 | 44 | 78 | 6 | 95 | 41 | 79 | 20 | 28 | 46 | 53 |
| 20 | 25 | 67 | 66 | 94 | 54 | 92 | 71 | 91 | 48 | 79 |
| 21 | 65 | 81 | 48 | 9 | 88 | 22 | 10 | 37 | 77 | 83 |
| 22 | 7 | 11 | 54 | 15 | 9 | 18 | 3 | 14 | 83 | 84 |
| 23 | 51 | 96 | 24 | 12 | 52 | 54 | 85 | 81 | 58 | 26 |
| 24 | 16 | 88 | 72 | 24 | 73 | 37 | 52 | 81 | 57 | 51 |
| 25 | 56 | 62 | 84 | 66 | 20 | 27 | 19 | 37 | 100 | 16 |
| 26 | 85 | 97 | 46 | 16 | 15 | 48 | 71 | 72 | 20 | 54 |
| 27 | 53 | 100 | 87 | 85 | 39 | 63 | 55 | 37 | 4 | 13 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 28 | 72 | 45 | 9 | 85 | 18 | 83 | 28 | 36 | 98 | 68 |
| 29 | 88 | 10 | 4 | 92 | 24 | 17 | 40 | 32 | 45 | 42 |
| 30 | 97 | 51 | 21 | 24 | 27 | 54 | 86 | 31 | 9 | 61 |
| 31 | 5 | 34 | 87 | 43 | 46 | 94 | 88 | 29 | 24 | 81 |
| 32 | 4 | 74 | 71 | 33 | 64 | 100 | 32 | 45 | 41 | 80 |
| 33 | 74 | 30 | 19 | 27 | 45 | 63 | 38 | 66 | 71 | 31 |
| 34 | 78 | 86 | 22 | 75 | 4 | 15 | 76 | 30 | 1 | 59 |
| 35 | 75 | 62 | 30 | 90 | 42 | 7 | 93 | 19 | 28 | 41 |
| 36 | 52 | 23 | 84 | 78 | 92 | 15 | 58 | 27 | 96 | 4 |
| 37 | 1 | 28 | 20 | 21 | 65 | 71 | 64 | 16 | 95 | 49 |
| 38 | 78 | 100 | 66 | 59 | 25 | 44 | 24 | 31 | 19 | 33 |
| 39 | 80 | 95 | 96 | 98 | 27 | 14 | 16 | 68 | 51 | 100 |
| 40 | 82 | 43 | 61 | 97 | 87 | 14 | 92 | 50 | 21 | 98 |
| 41 | 32 | 11 | 26 | 78 | 53 | 17 | 52 | 62 | 14 | 76 |
| 42 | 56 | 21 | 61 | 54 | 100 | 37 | 46 | 23 | 17 | 2 |
| 43 | 99 | 94 | 73 | 70 | 16 | 60 | 84 | 41 | 42 | 65 |
| 44 | 5 | 78 | 26 | 25 | 1 | 94 | 62 | 58 | 22 | 60 |
| 45 | 27 | 97 | 19 | 80 | 94 | 85 | 32 | 67 | 5 | 91 |
| 46 | 69 | 16 | 99 | 44 | 27 | 100 | 39 | 49 | 3 | 88 |
| 47 | 100 | 61 | 28 | 6 | 88 | 92 | 36 | 73 | 97 | 22 |
| 48 | 49 | 81 | 79 | 10 | 27 | 82 | 96 | 13 | 94 | 55 |
| 49 | 55 | 88 | 8 | 16 | 72 | 31 | 40 | 48 | 70 | 86 |
| 50 | 76 | 22 | 80 | 21 | 20 | 2 | 15 | 81 | 50 | 53 |
| 51 | 83 | 61 | 50 | 87 | 80 | 90 | 75 | 54 | 37 | 52 |
| 52 | 99 | 94 | 68 | 35 | 32 | 74 | 92 | 5 | 23 | 93 |
| 53 | 3 | 60 | 36 | 61 | 12 | 13 | 32 | 29 | 93 | 63 |
| 54 | 39 | 92 | 87 | 79 | 83 | 7 | 88 | 43 | 23 | 36 |
| 55 | 34 | 17 | 71 | 8 | 61 | 49 | 29 | 10 | 15 | 23 |
| 56 | 86 | 32 | 40 | 37 | 56 | 74 | 9 | 5 | 35 | 95 |
| 57 | 94 | 30 | 25 | 53 | 95 | 79 | 54 | 91 | 67 | 44 |
| 58 | 56 | 40 | 98 | 5 | 82 | 95 | 88 | 25 | 91 | 50 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 59 | 79 | 1 | 85 | 26 | 99 | 91 | 12 | 63 | 25 | 9 |
| 60 | 15 | 59 | 67 | 86 | 9 | 6 | 25 | 74 | 43 | 56 |
| 61 | 43 | 47 | 100 | 80 | 90 | 41 | 53 | 23 | 69 | 40 |
| 62 | 40 | 94 | 92 | 63 | 12 | 96 | 35 | 76 | 55 | 70 |
| 63 | 33 | 77 | 35 | 4 | 75 | 32 | 76 | 73 | 12 | 38 |
| 64 | 29 | 24 | 71 | 46 | 96 | 28 | 60 | 33 | 86 | 45 |
| 65 | 30 | 23 | 60 | 31 | 39 | 8 | 2 | 61 | 68 | 15 |
| 66 | 15 | 30 | 11 | 18 | 38 | 51 | 89 | 54 | 62 | 69 |
| 67 | 26 | 60 | 11 | 34 | 14 | 70 | 61 | 52 | 59 | 85 |
| 68 | 16 | 99 | 57 | 79 | 31 | 32 | 56 | 82 | 54 | 72 |
| 69 | 46 | 72 | 2 | 18 | 28 | 44 | 94 | 25 | 80 | 89 |
| 70 | 3 | 88 | 82 | 98 | 83 | 86 | 1 | 94 | 10 | 27 |
| 71 | 94 | 79 | 2 | 8 | 26 | 21 | 81 | 91 | 63 | 28 |
| 72 | 79 | 29 | 6 | 28 | 90 | 13 | 40 | 48 | 89 | 20 |
| 73 | 2 | 85 | 15 | 11 | 72 | 37 | 95 | 70 | 74 | 62 |
| 74 | 100 | 91 | 53 | 31 | 45 | 13 | 17 | 82 | 44 | 2 |
| 75 | 34 | 10 | 89 | 92 | 18 | 9 | 9 | 48 | 53 | 3 |
| 76 | 22 | 11 | 8 | 12 | 27 | 9 | 23 | 57 | 75 | 47 |
| 77 | 30 | 6 | 11 | 15 | 75 | 95 | 8 | 42 | 92 | 90 |
| 78 | 52 | 46 | 77 | 97 | 37 | 16 | 35 | 26 | 61 | 92 |
| 79 | 23 | 41 | 97 | 24 | 49 | 44 | 95 | 100 | 65 | 14 |
| 80 | 65 | 39 | 31 | 32 | 14 | 91 | 70 | 28 | 88 | 7 |
| 81 | 91 | 13 | 35 | 57 | 15 | 11 | 67 | 94 | 49 | 25 |
| 82 | 50 | 41 | 30 | 10 | 92 | 68 | 95 | 52 | 34 | 43 |
| 83 | 20 | 28 | 3 | 40 | 35 | 88 | 95 | 70 | 82 | 94 |
| 84 | 88 | 78 | 13 | 27 | 89 | 59 | 68 | 95 | 52 | 21 |
| 85 | 95 | 65 | 27 | 17 | 83 | 73 | 16 | 98 | 56 | 58 |
| 86 | 10 | 32 | 73 | 35 | 40 | 25 | 86 | 61 | 2 | 17 |
| 87 | 88 | 97 | 38 | 50 | 49 | 75 | 6 | 71 | 18 | 35 |
| 88 | 3 | 83 | 29 | 60 | 75 | 73 | 15 | 86 | 81 | 82 |
| 89 | 29 | 23 | 86 | 2 | 88 | 82 | 28 | 19 | 30 | 8 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 90 | 49 | 34 | 16 | 97 | 40 | 25 | 96 | 84 | 26 | 48 |
| 91 | 45 | 48 | 76 | 42 | 59 | 2 | 70 | 75 | 27 | 29 |
| 92 | 42 | 93 | 32 | 34 | 52 | 43 | 92 | 98 | 87 | 12 |
| 93 | 16 | 80 | 2 | 28 | 98 | 99 | 20 | 76 | 33 | 73 |
| 94 | 24 | 10 | 16 | 50 | 18 | 86 | 23 | 27 | 64 | 99 |
| 95 | 18 | 87 | 49 | 66 | 11 | 59 | 46 | 90 | 76 | 75 |
| 96 | 18 | 89 | 75 | 98 | 29 | 55 | 43 | 13 | 72 | 42 |
| 97 | 21 | 81 | 75 | 26 | 50 | 12 | 58 | 73 | 29 | 67 |
| 98 | 69 | 34 | 28 | 16 | 100 | 63 | 76 | 26 | 60 | 57 |
| 99 | 76 | 26 | 94 | 47 | 89 | 1 | 35 | 55 | 38 | 37 |
| 100 | 1 | 75 | 53 | 8 | 39 | 95 | 97 | 4 | 47 | 10 |
| 101 | 99 | 35 | 93 | 54 | 21 | 76 | 73 | 17 | 90 | 87 |
| 102 | 15 | 50 | 29 | 6 | 74 | 2 | 70 | 31 | 7 | 1 |
| 103 | 78 | 10 | 99 | 74 | 80 | 62 | 76 | 36 | 66 | 97 |
| 104 | 89 | 31 | 57 | 65 | 72 | 76 | 82 | 56 | 13 | 64 |
| 105 | 29 | 7 | 73 | 36 | 61 | 24 | 14 | 51 | 11 | 78 |
| 106 | 49 | 51 | 83 | 29 | 78 | 73 | 70 | 8 | 40 | 77 |
| 107 | 52 | 18 | 26 | 86 | 29 | 1 | 32 | 88 | 73 | 19 |
| 108 | 59 | 11 | 41 | 16 | 12 | 99 | 98 | 91 | 85 | 71 |
| 109 | 64 | 23 | 36 | 58 | 69 | 3 | 83 | 22 | 99 | 6 |
| 110 | 52 | 70 | 55 | 58 | 83 | 57 | 51 | 91 | 8 | 95 |
| 111 | 21 | 57 | 41 | 23 | 49 | 98 | 34 | 9 | 39 | 18 |
| 112 | 82 | 51 | 70 | 89 | 45 | 68 | 43 | 38 | 78 | 72 |
| 113 | 83 | 28 | 60 | 54 | 94 | 78 | 37 | 45 | 36 | 39 |
| 114 | 68 | 46 | 10 | 86 | 19 | 74 | 87 | 26 | 79 | 96 |
| 115 | 19 | 99 | 67 | 82 | 96 | 73 | 36 | 65 | 61 | 90 |
| 116 | 95 | 91 | 46 | 17 | 1 | 10 | 84 | 98 | 31 | 11 |
| 117 | 68 | 93 | 29 | 16 | 66 | 98 | 40 | 95 | 6 | 34 |
| 118 | 7 | 46 | 75 | 48 | 37 | 43 | 36 | 42 | 4 | 66 |
| 119 | 44 | 78 | 6 | 95 | 41 | 79 | 20 | 28 | 46 | 53 |
| 120 | 25 | 67 | 66 | 94 | 54 | 92 | 71 | 91 | 48 | 79 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 121 | 65 | 81 | 48 | 9 | 88 | 22 | 10 | 37 | 77 | 83 |
| 122 | 7 | 11 | 54 | 15 | 9 | 18 | 3 | 14 | 83 | 84 |
| 123 | 51 | 96 | 24 | 12 | 52 | 54 | 85 | 81 | 58 | 26 |
| 124 | 16 | 88 | 72 | 24 | 73 | 37 | 52 | 81 | 57 | 51 |
| 125 | 56 | 62 | 84 | 66 | 20 | 27 | 19 | 37 | 100 | 16 |
| 126 | 85 | 97 | 46 | 16 | 15 | 48 | 71 | 72 | 20 | 54 |
| 127 | 53 | 100 | 87 | 85 | 39 | 63 | 55 | 37 | 4 | 13 |
| 128 | 72 | 45 | 9 | 85 | 18 | 83 | 28 | 36 | 98 | 68 |
| 129 | 88 | 10 | 4 | 92 | 24 | 17 | 40 | 32 | 45 | 42 |
| 130 | 97 | 51 | 21 | 24 | 27 | 54 | 86 | 31 | 9 | 61 |
| 131 | 5 | 34 | 87 | 43 | 46 | 94 | 88 | 29 | 24 | 81 |
| 132 | 4 | 74 | 71 | 33 | 64 | 100 | 32 | 45 | 41 | 80 |
| 133 | 74 | 30 | 19 | 27 | 45 | 63 | 38 | 66 | 71 | 31 |
| 134 | 78 | 86 | 22 | 75 | 4 | 15 | 76 | 30 | 1 | 59 |
| 135 | 75 | 62 | 30 | 90 | 42 | 7 | 93 | 19 | 28 | 41 |
| 136 | 52 | 23 | 84 | 78 | 92 | 15 | 58 | 27 | 96 | 4 |
| 137 | 1 | 28 | 20 | 21 | 65 | 71 | 64 | 16 | 95 | 49 |
| 138 | 78 | 100 | 66 | 59 | 25 | 44 | 24 | 31 | 19 | 33 |
| 139 | 80 | 95 | 96 | 98 | 27 | 14 | 16 | 68 | 51 | 100 |
| 140 | 82 | 43 | 61 | 97 | 87 | 14 | 92 | 50 | 21 | 98 |
| 141 | 32 | 11 | 26 | 78 | 53 | 17 | 52 | 62 | 14 | 76 |
| 142 | 56 | 21 | 61 | 54 | 100 | 37 | 46 | 23 | 17 | 2 |
| 143 | 99 | 94 | 73 | 70 | 16 | 60 | 84 | 41 | 42 | 65 |
| 144 | 5 | 78 | 26 | 25 | 1 | 94 | 62 | 58 | 22 | 60 |
| 145 | 27 | 97 | 19 | 80 | 94 | 85 | 32 | 67 | 5 | 91 |
| 146 | 69 | 16 | 99 | 44 | 27 | 100 | 39 | 49 | 3 | 88 |
| 147 | 100 | 61 | 28 | 6 | 88 | 92 | 36 | 73 | 97 | 22 |
| 148 | 49 | 81 | 79 | 10 | 27 | 82 | 96 | 13 | 94 | 55 |
| 149 | 55 | 88 | 8 | 16 | 72 | 31 | 40 | 48 | 70 | 86 |
| 150 | 76 | 22 | 80 | 21 | 20 | 2 | 15 | 81 | 50 | 53 |
| 151 | 83 | 61 | 50 | 87 | 80 | 90 | 75 | 54 | 37 | 52 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 152 | 99 | 94 | 68 | 35 | 32 | 74 | 92 | 5 | 23 | 93 |
| 153 | 3 | 60 | 36 | 61 | 12 | 13 | 32 | 29 | 93 | 63 |
| 154 | 39 | 92 | 87 | 79 | 83 | 7 | 88 | 43 | 23 | 36 |
| 155 | 34 | 17 | 71 | 8 | 61 | 49 | 29 | 10 | 15 | 23 |
| 156 | 86 | 32 | 40 | 37 | 56 | 74 | 9 | 5 | 35 | 95 |
| 157 | 94 | 30 | 25 | 53 | 95 | 79 | 54 | 91 | 67 | 44 |
| 158 | 56 | 40 | 98 | 5 | 82 | 95 | 88 | 25 | 91 | 50 |
| 159 | 79 | 1 | 85 | 26 | 99 | 91 | 12 | 63 | 25 | 9 |
| 160 | 15 | 59 | 67 | 86 | 9 | 6 | 25 | 74 | 43 | 56 |
| 161 | 43 | 47 | 100 | 80 | 90 | 41 | 53 | 23 | 69 | 40 |
| 162 | 40 | 94 | 92 | 63 | 12 | 96 | 35 | 76 | 55 | 70 |
| 163 | 33 | 77 | 35 | 4 | 75 | 32 | 76 | 73 | 12 | 38 |
| 164 | 29 | 24 | 71 | 46 | 96 | 28 | 60 | 33 | 86 | 45 |
| 165 | 30 | 23 | 60 | 31 | 39 | 8 | 2 | 61 | 68 | 15 |
| 166 | 15 | 30 | 11 | 18 | 38 | 51 | 89 | 54 | 62 | 69 |
| 167 | 26 | 60 | 11 | 34 | 14 | 70 | 61 | 52 | 59 | 85 |
| 168 | 16 | 99 | 57 | 79 | 31 | 32 | 56 | 82 | 54 | 72 |
| 169 | 46 | 72 | 2 | 18 | 28 | 44 | 94 | 25 | 80 | 89 |
| 170 | 3 | 88 | 82 | 98 | 83 | 86 | 1 | 94 | 10 | 27 |
| 171 | 94 | 79 | 2 | 8 | 26 | 21 | 81 | 91 | 63 | 28 |
| 172 | 79 | 29 | 6 | 28 | 90 | 13 | 40 | 48 | 89 | 20 |
| 173 | 2 | 85 | 15 | 11 | 72 | 37 | 95 | 70 | 74 | 62 |
| 174 | 100 | 91 | 53 | 31 | 45 | 13 | 17 | 82 | 44 | 2 |
| 175 | 34 | 10 | 89 | 92 | 18 | 9 | 9 | 48 | 53 | 3 |
| 176 | 22 | 11 | 8 | 12 | 27 | 9 | 23 | 57 | 75 | 47 |
| 177 | 30 | 6 | 11 | 15 | 75 | 95 | 8 | 42 | 92 | 90 |
| 178 | 52 | 46 | 77 | 97 | 37 | 16 | 35 | 26 | 61 | 92 |
| 179 | 23 | 41 | 97 | 24 | 49 | 44 | 95 | 100 | 65 | 14 |
| 180 | 65 | 39 | 31 | 32 | 14 | 91 | 70 | 28 | 88 | 7 |
| 181 | 91 | 13 | 35 | 57 | 15 | 11 | 67 | 94 | 49 | 25 |
| 182 | 50 | 41 | 30 | 10 | 92 | 68 | 95 | 52 | 34 | 43 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 183 | 20 | 28 | 3 | 40 | 35 | 88 | 95 | 70 | 82 | 94 |
| 184 | 88 | 78 | 13 | 27 | 89 | 59 | 68 | 95 | 52 | 21 |
| 185 | 95 | 65 | 27 | 17 | 83 | 73 | 16 | 98 | 56 | 58 |
| 186 | 10 | 32 | 73 | 35 | 40 | 25 | 86 | 61 | 2 | 17 |
| 187 | 88 | 97 | 38 | 50 | 49 | 75 | 6 | 71 | 18 | 35 |
| 188 | 3 | 83 | 29 | 60 | 75 | 73 | 15 | 86 | 81 | 82 |
| 189 | 29 | 23 | 86 | 2 | 88 | 82 | 28 | 19 | 30 | 8 |
| 190 | 49 | 34 | 16 | 97 | 40 | 25 | 96 | 84 | 26 | 48 |
| 191 | 45 | 48 | 76 | 42 | 59 | 2 | 70 | 75 | 27 | 29 |
| 192 | 42 | 93 | 32 | 34 | 52 | 43 | 92 | 98 | 87 | 12 |
| 193 | 16 | 80 | 2 | 28 | 98 | 99 | 20 | 76 | 33 | 73 |
| 194 | 24 | 10 | 16 | 50 | 18 | 86 | 23 | 27 | 64 | 99 |
| 195 | 18 | 87 | 49 | 66 | 11 | 59 | 46 | 90 | 76 | 75 |
| 196 | 18 | 89 | 75 | 98 | 29 | 55 | 43 | 13 | 72 | 42 |
| 197 | 21 | 81 | 75 | 26 | 50 | 12 | 58 | 73 | 29 | 67 |
| 198 | 69 | 34 | 28 | 16 | 100 | 63 | 76 | 26 | 60 | 57 |
| 199 | 76 | 26 | 94 | 47 | 89 | 1 | 35 | 55 | 38 | 37 |
| 200 | 1 | 75 | 53 | 8 | 39 | 95 | 97 | 4 | 47 | 10 |
| 201 | 99 | 35 | 93 | 54 | 21 | 76 | 73 | 17 | 90 | 87 |
| 202 | 15 | 50 | 29 | 6 | 74 | 2 | 70 | 31 | 7 | 1 |
| 203 | 78 | 10 | 99 | 74 | 80 | 62 | 76 | 36 | 66 | 97 |
| 204 | 89 | 31 | 57 | 65 | 72 | 76 | 82 | 56 | 13 | 64 |
| 205 | 29 | 7 | 73 | 36 | 61 | 24 | 14 | 51 | 11 | 78 |
| 206 | 49 | 51 | 83 | 29 | 78 | 73 | 70 | 8 | 40 | 77 |
| 207 | 52 | 18 | 26 | 86 | 29 | 1 | 32 | 88 | 73 | 19 |
| 208 | 59 | 11 | 41 | 16 | 12 | 99 | 98 | 91 | 85 | 71 |
| 209 | 64 | 23 | 36 | 58 | 69 | 3 | 83 | 22 | 99 | 6 |
| 210 | 52 | 70 | 55 | 58 | 83 | 57 | 51 | 91 | 8 | 95 |
| 211 | 21 | 57 | 41 | 23 | 49 | 98 | 34 | 9 | 39 | 18 |
| 212 | 82 | 51 | 70 | 89 | 45 | 68 | 43 | 38 | 78 | 72 |
| 213 | 83 | 28 | 60 | 54 | 94 | 78 | 37 | 45 | 36 | 39 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 214 | 68 | 46 | 10 | 86 | 19 | 74 | 87 | 26 | 79 | 96 |
| 215 | 19 | 99 | 67 | 82 | 96 | 73 | 36 | 65 | 61 | 90 |
| 216 | 95 | 91 | 46 | 17 | 1 | 10 | 84 | 98 | 31 | 11 |
| 217 | 68 | 93 | 29 | 16 | 66 | 98 | 40 | 95 | 6 | 34 |
| 218 | 7 | 46 | 75 | 48 | 37 | 43 | 36 | 42 | 4 | 66 |
| 219 | 44 | 78 | 6 | 95 | 41 | 79 | 20 | 28 | 46 | 53 |
| 220 | 25 | 67 | 66 | 94 | 54 | 92 | 71 | 91 | 48 | 79 |
| 221 | 65 | 81 | 48 | 9 | 88 | 22 | 10 | 37 | 77 | 83 |
| 222 | 7 | 11 | 54 | 15 | 9 | 18 | 3 | 14 | 83 | 84 |
| 223 | 51 | 96 | 24 | 12 | 52 | 54 | 85 | 81 | 58 | 26 |
| 224 | 16 | 88 | 72 | 24 | 73 | 37 | 52 | 81 | 57 | 51 |
| 225 | 56 | 62 | 84 | 66 | 20 | 27 | 19 | 37 | 100 | 16 |
| 226 | 85 | 97 | 46 | 16 | 15 | 48 | 71 | 72 | 20 | 54 |
| 227 | 53 | 100 | 87 | 85 | 39 | 63 | 55 | 37 | 4 | 13 |
| 228 | 72 | 45 | 9 | 85 | 18 | 83 | 28 | 36 | 98 | 68 |
| 229 | 88 | 10 | 4 | 92 | 24 | 17 | 40 | 32 | 45 | 42 |
| 230 | 97 | 51 | 21 | 24 | 27 | 54 | 86 | 31 | 9 | 61 |
| 231 | 5 | 34 | 87 | 43 | 46 | 94 | 88 | 29 | 24 | 81 |
| 232 | 4 | 74 | 71 | 33 | 64 | 100 | 32 | 45 | 41 | 80 |
| 233 | 74 | 30 | 19 | 27 | 45 | 63 | 38 | 66 | 71 | 31 |
| 234 | 78 | 86 | 22 | 75 | 4 | 15 | 76 | 30 | 1 | 59 |
| 235 | 75 | 62 | 30 | 90 | 42 | 7 | 93 | 19 | 28 | 41 |
| 236 | 52 | 23 | 84 | 78 | 92 | 15 | 58 | 27 | 96 | 4 |
| 237 | 1 | 28 | 20 | 21 | 65 | 71 | 64 | 16 | 95 | 49 |
| 238 | 78 | 100 | 66 | 59 | 25 | 44 | 24 | 31 | 19 | 33 |
| 239 | 80 | 95 | 96 | 98 | 27 | 14 | 16 | 68 | 51 | 100 |
| 240 | 82 | 43 | 61 | 97 | 87 | 14 | 92 | 50 | 21 | 98 |
| 241 | 32 | 11 | 26 | 78 | 53 | 17 | 52 | 62 | 14 | 76 |
| 242 | 56 | 21 | 61 | 54 | 100 | 37 | 46 | 23 | 17 | 2 |
| 243 | 99 | 94 | 73 | 70 | 16 | 60 | 84 | 41 | 42 | 65 |
| 244 | 5 | 78 | 26 | 25 | 1 | 94 | 62 | 58 | 22 | 60 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 245 | 27 | 97 | 19 | 80 | 94 | 85 | 32 | 67 | 5 | 91 |
| 246 | 69 | 16 | 99 | 44 | 27 | 100 | 39 | 49 | 3 | 88 |
| 247 | 100 | 61 | 28 | 6 | 88 | 92 | 36 | 73 | 97 | 22 |
| 248 | 49 | 81 | 79 | 10 | 27 | 82 | 96 | 13 | 94 | 55 |
| 249 | 55 | 88 | 8 | 16 | 72 | 31 | 40 | 48 | 70 | 86 |
| 250 | 76 | 22 | 80 | 21 | 20 | 2 | 15 | 81 | 50 | 53 |
| 251 | 83 | 61 | 50 | 87 | 80 | 90 | 75 | 54 | 37 | 52 |
| 252 | 99 | 94 | 68 | 35 | 32 | 74 | 92 | 5 | 23 | 93 |
| 253 | 3 | 60 | 36 | 61 | 12 | 13 | 32 | 29 | 93 | 63 |
| 254 | 39 | 92 | 87 | 79 | 83 | 7 | 88 | 43 | 23 | 36 |
| 255 | 34 | 17 | 71 | 8 | 61 | 49 | 29 | 10 | 15 | 23 |
| 256 | 86 | 32 | 40 | 37 | 56 | 74 | 9 | 5 | 35 | 95 |
| 257 | 94 | 30 | 25 | 53 | 95 | 79 | 54 | 91 | 67 | 44 |
| 258 | 56 | 40 | 98 | 5 | 82 | 95 | 88 | 25 | 91 | 50 |
| 259 | 79 | 1 | 85 | 26 | 99 | 91 | 12 | 63 | 25 | 9 |
| 260 | 15 | 59 | 67 | 86 | 9 | 6 | 25 | 74 | 43 | 56 |
| 261 | 43 | 47 | 100 | 80 | 90 | 41 | 53 | 23 | 69 | 40 |
| 262 | 40 | 94 | 92 | 63 | 12 | 96 | 35 | 76 | 55 | 70 |
| 263 | 33 | 77 | 35 | 4 | 75 | 32 | 76 | 73 | 12 | 38 |
| 264 | 29 | 24 | 71 | 46 | 96 | 28 | 60 | 33 | 86 | 45 |
| 265 | 30 | 23 | 60 | 31 | 39 | 8 | 2 | 61 | 68 | 15 |
| 266 | 15 | 30 | 11 | 18 | 38 | 51 | 89 | 54 | 62 | 69 |
| 267 | 26 | 60 | 11 | 34 | 14 | 70 | 61 | 52 | 59 | 85 |
| 268 | 16 | 99 | 57 | 79 | 31 | 32 | 56 | 82 | 54 | 72 |
| 269 | 46 | 72 | 2 | 18 | 28 | 44 | 94 | 25 | 80 | 89 |
| 270 | 3 | 88 | 82 | 98 | 83 | 86 | 1 | 94 | 10 | 27 |
| 271 | 94 | 79 | 2 | 8 | 26 | 21 | 81 | 91 | 63 | 28 |
| 272 | 79 | 29 | 6 | 28 | 90 | 13 | 40 | 48 | 89 | 20 |
| 273 | 2 | 85 | 15 | 11 | 72 | 37 | 95 | 70 | 74 | 62 |
| 274 | 100 | 91 | 53 | 31 | 45 | 13 | 17 | 82 | 44 | 2 |
| 275 | 34 | 10 | 89 | 92 | 18 | 9 | 9 | 48 | 53 | 3 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 276 | 22 | 11 | 8 | 12 | 27 | 9 | 23 | 57 | 75 | 47 |
| 277 | 30 | 6 | 11 | 15 | 75 | 95 | 8 | 42 | 92 | 90 |
| 278 | 52 | 46 | 77 | 97 | 37 | 16 | 35 | 26 | 61 | 92 |
| 279 | 23 | 41 | 97 | 24 | 49 | 44 | 95 | 100 | 65 | 14 |
| 280 | 65 | 39 | 31 | 32 | 14 | 91 | 70 | 28 | 88 | 7 |
| 281 | 91 | 13 | 35 | 57 | 15 | 11 | 67 | 94 | 49 | 25 |
| 282 | 50 | 41 | 30 | 10 | 92 | 68 | 95 | 52 | 34 | 43 |
| 283 | 20 | 28 | 3 | 40 | 35 | 88 | 95 | 70 | 82 | 94 |
| 284 | 88 | 78 | 13 | 27 | 89 | 59 | 68 | 95 | 52 | 21 |
| 285 | 95 | 65 | 27 | 17 | 83 | 73 | 16 | 98 | 56 | 58 |
| 286 | 10 | 32 | 73 | 35 | 40 | 25 | 86 | 61 | 2 | 17 |
| 287 | 88 | 97 | 38 | 50 | 49 | 75 | 6 | 71 | 18 | 35 |
| 288 | 3 | 83 | 29 | 60 | 75 | 73 | 15 | 86 | 81 | 82 |
| 289 | 29 | 23 | 86 | 2 | 88 | 82 | 28 | 19 | 30 | 8 |
| 290 | 49 | 34 | 16 | 97 | 40 | 25 | 96 | 84 | 26 | 48 |
| 291 | 45 | 48 | 76 | 42 | 59 | 2 | 70 | 75 | 27 | 29 |
| 292 | 42 | 93 | 32 | 34 | 52 | 43 | 92 | 98 | 87 | 12 |
| 293 | 16 | 80 | 2 | 28 | 98 | 99 | 20 | 76 | 33 | 73 |
| 294 | 24 | 10 | 16 | 50 | 18 | 86 | 23 | 27 | 64 | 99 |
| 295 | 18 | 87 | 49 | 66 | 11 | 59 | 46 | 90 | 76 | 75 |
| 296 | 18 | 89 | 75 | 98 | 29 | 55 | 43 | 13 | 72 | 42 |
| 297 | 21 | 81 | 75 | 26 | 50 | 12 | 58 | 73 | 29 | 67 |
| 298 | 69 | 34 | 28 | 16 | 100 | 63 | 76 | 26 | 60 | 57 |
| 299 | 76 | 26 | 94 | 47 | 89 | 1 | 35 | 55 | 38 | 37 |
| 300 | 1 | 75 | 53 | 8 | 39 | 95 | 97 | 4 | 47 | 10 |
| 301 | 99 | 35 | 93 | 54 | 21 | 76 | 73 | 17 | 90 | 87 |
| 302 | 15 | 50 | 29 | 6 | 74 | 2 | 70 | 31 | 7 | 1 |
| 303 | 78 | 10 | 99 | 74 | 80 | 62 | 76 | 36 | 66 | 97 |
| 304 | 89 | 31 | 57 | 65 | 72 | 76 | 82 | 56 | 13 | 64 |
| 305 | 29 | 7 | 73 | 36 | 61 | 24 | 14 | 51 | 11 | 78 |
| 306 | 49 | 51 | 83 | 29 | 78 | 73 | 70 | 8 | 40 | 77 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 307 | 52 | 18 | 26 | 86 | 29 | 1 | 32 | 88 | 73 | 19 |
| 308 | 59 | 11 | 41 | 16 | 12 | 99 | 98 | 91 | 85 | 71 |
| 309 | 64 | 23 | 36 | 58 | 69 | 3 | 83 | 22 | 99 | 6 |
| 310 | 52 | 70 | 55 | 58 | 83 | 57 | 51 | 91 | 8 | 95 |
| 311 | 21 | 57 | 41 | 23 | 49 | 98 | 34 | 9 | 39 | 18 |
| 312 | 82 | 51 | 70 | 89 | 45 | 68 | 43 | 38 | 78 | 72 |
| 313 | 83 | 28 | 60 | 54 | 94 | 78 | 37 | 45 | 36 | 39 |
| 314 | 68 | 46 | 10 | 86 | 19 | 74 | 87 | 26 | 79 | 96 |
| 315 | 19 | 99 | 67 | 82 | 96 | 73 | 36 | 65 | 61 | 90 |
| 316 | 95 | 91 | 46 | 17 | 1 | 10 | 84 | 98 | 31 | 11 |
| 317 | 68 | 93 | 29 | 16 | 66 | 98 | 40 | 95 | 6 | 34 |
| 318 | 7 | 46 | 75 | 48 | 37 | 43 | 36 | 42 | 4 | 66 |
| 319 | 44 | 78 | 6 | 95 | 41 | 79 | 20 | 28 | 46 | 53 |
| 320 | 25 | 67 | 66 | 94 | 54 | 92 | 71 | 91 | 48 | 79 |
| 321 | 65 | 81 | 48 | 9 | 88 | 22 | 10 | 37 | 77 | 83 |
| 322 | 7 | 11 | 54 | 15 | 9 | 18 | 3 | 14 | 83 | 84 |
| 323 | 51 | 96 | 24 | 12 | 52 | 54 | 85 | 81 | 58 | 26 |
| 324 | 16 | 88 | 72 | 24 | 73 | 37 | 52 | 81 | 57 | 51 |
| 325 | 56 | 62 | 84 | 66 | 20 | 27 | 19 | 37 | 100 | 16 |
| 326 | 85 | 97 | 46 | 16 | 15 | 48 | 71 | 72 | 20 | 54 |
| 327 | 53 | 100 | 87 | 85 | 39 | 63 | 55 | 37 | 4 | 13 |
| 328 | 72 | 45 | 9 | 85 | 18 | 83 | 28 | 36 | 98 | 68 |
| 329 | 88 | 10 | 4 | 92 | 24 | 17 | 40 | 32 | 45 | 42 |
| 330 | 97 | 51 | 21 | 24 | 27 | 54 | 86 | 31 | 9 | 61 |
| 331 | 5 | 34 | 87 | 43 | 46 | 94 | 88 | 29 | 24 | 81 |
| 332 | 4 | 74 | 71 | 33 | 64 | 100 | 32 | 45 | 41 | 80 |
| 333 | 74 | 30 | 19 | 27 | 45 | 63 | 38 | 66 | 71 | 31 |
| 334 | 78 | 86 | 22 | 75 | 4 | 15 | 76 | 30 | 1 | 59 |
| 335 | 75 | 62 | 30 | 90 | 42 | 7 | 93 | 19 | 28 | 41 |
| 336 | 52 | 23 | 84 | 78 | 92 | 15 | 58 | 27 | 96 | 4 |
| 337 | 1 | 28 | 20 | 21 | 65 | 71 | 64 | 16 | 95 | 49 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 338 | 78 | 100 | 66 | 59 | 25 | 44 | 24 | 31 | 19 | 33 |
| 339 | 80 | 95 | 96 | 98 | 27 | 14 | 16 | 68 | 51 | 100 |
| 340 | 82 | 43 | 61 | 97 | 87 | 14 | 92 | 50 | 21 | 98 |
| 341 | 32 | 11 | 26 | 78 | 53 | 17 | 52 | 62 | 14 | 76 |
| 342 | 56 | 21 | 61 | 54 | 100 | 37 | 46 | 23 | 17 | 2 |
| 343 | 99 | 94 | 73 | 70 | 16 | 60 | 84 | 41 | 42 | 65 |
| 344 | 5 | 78 | 26 | 25 | 1 | 94 | 62 | 58 | 22 | 60 |
| 345 | 27 | 97 | 19 | 80 | 94 | 85 | 32 | 67 | 5 | 91 |
| 346 | 69 | 16 | 99 | 44 | 27 | 100 | 39 | 49 | 3 | 88 |
| 347 | 100 | 61 | 28 | 6 | 88 | 92 | 36 | 73 | 97 | 22 |
| 348 | 49 | 81 | 79 | 10 | 27 | 82 | 96 | 13 | 94 | 55 |
| 349 | 55 | 88 | 8 | 16 | 72 | 31 | 40 | 48 | 70 | 86 |
| 350 | 76 | 22 | 80 | 21 | 20 | 2 | 15 | 81 | 50 | 53 |
| 351 | 83 | 61 | 50 | 87 | 80 | 90 | 75 | 54 | 37 | 52 |
| 352 | 99 | 94 | 68 | 35 | 32 | 74 | 92 | 5 | 23 | 93 |
| 353 | 3 | 60 | 36 | 61 | 12 | 13 | 32 | 29 | 93 | 63 |
| 354 | 39 | 92 | 87 | 79 | 83 | 7 | 88 | 43 | 23 | 36 |
| 355 | 34 | 17 | 71 | 8 | 61 | 49 | 29 | 10 | 15 | 23 |
| 356 | 86 | 32 | 40 | 37 | 56 | 74 | 9 | 5 | 35 | 95 |
| 357 | 94 | 30 | 25 | 53 | 95 | 79 | 54 | 91 | 67 | 44 |
| 358 | 56 | 40 | 98 | 5 | 82 | 95 | 88 | 25 | 91 | 50 |
| 359 | 79 | 1 | 85 | 26 | 99 | 91 | 12 | 63 | 25 | 9 |
| 360 | 15 | 59 | 67 | 86 | 9 | 6 | 25 | 74 | 43 | 56 |
| 361 | 43 | 47 | 100 | 80 | 90 | 41 | 53 | 23 | 69 | 40 |
| 362 | 40 | 94 | 92 | 63 | 12 | 96 | 35 | 76 | 55 | 70 |
| 363 | 33 | 77 | 35 | 4 | 75 | 32 | 76 | 73 | 12 | 38 |
| 364 | 29 | 24 | 71 | 46 | 96 | 28 | 60 | 33 | 86 | 45 |
| 365 | 30 | 23 | 60 | 31 | 39 | 8 | 2 | 61 | 68 | 15 |
| 366 | 15 | 30 | 11 | 18 | 38 | 51 | 89 | 54 | 62 | 69 |
| 367 | 26 | 60 | 11 | 34 | 14 | 70 | 61 | 52 | 59 | 85 |
| 368 | 16 | 99 | 57 | 79 | 31 | 32 | 56 | 82 | 54 | 72 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 369 | 46 | 72 | 2 | 18 | 28 | 44 | 94 | 25 | 80 | 89 |
| 370 | 3 | 88 | 82 | 98 | 83 | 86 | 1 | 94 | 10 | 27 |
| 371 | 94 | 79 | 2 | 8 | 26 | 21 | 81 | 91 | 63 | 28 |
| 372 | 79 | 29 | 6 | 28 | 90 | 13 | 40 | 48 | 89 | 20 |
| 373 | 2 | 85 | 15 | 11 | 72 | 37 | 95 | 70 | 74 | 62 |
| 374 | 100 | 91 | 53 | 31 | 45 | 13 | 17 | 82 | 44 | 2 |
| 375 | 34 | 10 | 89 | 92 | 18 | 9 | 9 | 48 | 53 | 3 |
| 376 | 22 | 11 | 8 | 12 | 27 | 9 | 23 | 57 | 75 | 47 |
| 377 | 30 | 6 | 11 | 15 | 75 | 95 | 8 | 42 | 92 | 90 |
| 378 | 52 | 46 | 77 | 97 | 37 | 16 | 35 | 26 | 61 | 92 |
| 379 | 23 | 41 | 97 | 24 | 49 | 44 | 95 | 100 | 65 | 14 |
| 380 | 65 | 39 | 31 | 32 | 14 | 91 | 70 | 28 | 88 | 7 |
| 381 | 91 | 13 | 35 | 57 | 15 | 11 | 67 | 94 | 49 | 25 |
| 382 | 50 | 41 | 30 | 10 | 92 | 68 | 95 | 52 | 34 | 43 |
| 383 | 20 | 28 | 3 | 40 | 35 | 88 | 95 | 70 | 82 | 94 |
| 384 | 88 | 78 | 13 | 27 | 89 | 59 | 68 | 95 | 52 | 21 |
| 385 | 95 | 65 | 27 | 17 | 83 | 73 | 16 | 98 | 56 | 58 |
| 386 | 10 | 32 | 73 | 35 | 40 | 25 | 86 | 61 | 2 | 17 |
| 387 | 88 | 97 | 38 | 50 | 49 | 75 | 6 | 71 | 18 | 35 |
| 388 | 3 | 83 | 29 | 60 | 75 | 73 | 15 | 86 | 81 | 82 |
| 389 | 29 | 23 | 86 | 2 | 88 | 82 | 28 | 19 | 30 | 8 |
| 390 | 49 | 34 | 16 | 97 | 40 | 25 | 96 | 84 | 26 | 48 |
| 391 | 45 | 48 | 76 | 42 | 59 | 2 | 70 | 75 | 27 | 29 |
| 392 | 42 | 93 | 32 | 34 | 52 | 43 | 92 | 98 | 87 | 12 |
| 393 | 16 | 80 | 2 | 28 | 98 | 99 | 20 | 76 | 33 | 73 |
| 394 | 24 | 10 | 16 | 50 | 18 | 86 | 23 | 27 | 64 | 99 |
| 395 | 18 | 87 | 49 | 66 | 11 | 59 | 46 | 90 | 76 | 75 |
| 396 | 18 | 89 | 75 | 98 | 29 | 55 | 43 | 13 | 72 | 42 |
| 397 | 21 | 81 | 75 | 26 | 50 | 12 | 58 | 73 | 29 | 67 |
| 398 | 69 | 34 | 28 | 16 | 100 | 63 | 76 | 26 | 60 | 57 |
| 399 | 76 | 26 | 94 | 47 | 89 | 1 | 35 | 55 | 38 | 37 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 400 | 1 | 75 | 53 | 8 | 39 | 95 | 97 | 4 | 47 | 10 |

Lampiran 3. Data Total Konsumsi Energi

| Iterasi | Total Konsumsi Energi (Joule) Algoritma *K-Means* | Iterasi | Total Konsumsi Energi (Joule) Algoritma *K-Means++* |
|---|---|---|---|
| 1 | 0.0598697133004236 | 1 | 0.0776744585062477 |
| 2 | 0.137131060271127 | 2 | 0.137595102425465 |
| 3 | 0.20215929796957 | 3 | 0.203400539182279 |
| 4 | 0.268546991520651 | 4 | 0.274159114920395 |
| 5 | 0.323675029438688 | 5 | 0.326052725636506 |
| 6 | 0.388328626974386 | 6 | 0.403727184142754 |
| 7 | 0.442359242786202 | 7 | 0.93346496937776 |
| 8 | 0.517180166726883 | 8 | 0.993385613296978 |
| 9 | 0.576626545597496 | 9 | 1.27138561660108 |
| 10 | 0.641150132715089 | 10 | 1.3371910533579 |
| 11 | 0.735866422118487 | 11 | 1.41486551186414 |
| 12 | 0.79373159235674 | 12 | 1.74900950980787 |
| 13 | 0.868141498615963 | 13 | 1.81976808554599 |
| 14 | 0.936169206604998 | 14 | 1.87968872946521 |
| 15 | 1.15334795145812 | 15 | 2.26295835211351 |
| 16 | 1.22234421924249 | 16 | 2.34063281061975 |
| 17 | 1.2854873190906 | 17 | 2.39252642133586 |
| 18 | 1.34673791820745 | 18 | 2.45833185809268 |
| 19 | 1.67947982945695 | 19 | 2.66669745221186 |
| 20 | 1.99919774734145 | 20 | 2.72661809613107 |
| 21 | 2.18408893795787 | 21 | 2.80429255463732 |
| 22 | 2.50505156115539 | 22 | 3.33403033987233 |
| 23 | 2.57809211622866 | 23 | 3.40478891561044 |
| 24 | 2.74260674869996 | 24 | 3.63986282563367 |
| 25 | 2.89341009809755 | 25 | 3.70566826239048 |
| 26 | 2.9576782088264 | 26 | 3.78334272089673 |

| | | | |
|---|---|---|---|
| 27 | 3.12069721363103 | 27 | 3.84326336481595 |
| 28 | 3.19638266165039 | 28 | 4.12126336812006 |
| 29 | 3.50989248067046 | 29 | 4.17315697883617 |
| 30 | 3.58689131171225 | 30 | 4.38596135972984 |
| 31 | 3.66932303299555 | 31 | 4.46363581823609 |
| 32 | 3.9693534132312 | 32 | 4.53439439397421 |
| 33 | 4.03094893998105 | 33 | 4.59431503789342 |
| 34 | 4.32994993452154 | 34 | 4.66012047465024 |
| 35 | 4.50759243751887 | 35 | 4.99426447259397 |
| 36 | 4.74385866225823 | 36 | 5.07193893110021 |
| 37 | 4.84965882168958 | 37 | 5.60167671633522 |
| 38 | 4.96654561628828 | 38 | 5.82578512043689 |
| 39 | 5.0313870329901 | 39 | 5.8857057643561 |
| 40 | 5.09349726224153 | 40 | 5.93759937507221 |
| 41 | 5.19871527969977 | 41 | 6.01527383357846 |
| 42 | 5.40115702447516 | 42 | 6.08107927033528 |
| 43 | 5.45510289442919 | 43 | 6.15183784607339 |
| 44 | 5.71219076650448 | 44 | 6.53510746872169 |
| 45 | 5.973604643341 | 45 | 6.59502811264091 |
| 46 | 6.22735626357214 | 46 | 6.67270257114716 |
| 47 | 6.2913957386889 | 47 | 6.95070257445126 |
| 48 | 6.51971718308348 | 48 | 7.02690136696561 |
| 49 | 6.58954267802138 | 49 | 7.09270680372242 |
| 50 | 6.97563998128815 | 50 | 7.62244458895743 |
| 51 | 7.02618873337126 | 51 | 7.70011904746367 |
| 52 | 7.23533619704807 | 52 | 7.76003969138289 |
| 53 | 7.34586235518991 | 53 | 7.83079826712101 |
| 54 | 7.43006892114629 | 54 | 7.88269187783712 |
| 55 | 7.58013540238359 | 55 | 8.0910574719563 |
| 56 | 7.87751710904665 | 56 | 8.16873193046254 |
| 57 | 8.33908782264201 | 57 | 8.23453736721936 |

| | | | |
|---|---|---|---|
| 58 | 8.70983144478783 | 58 | 8.29445801113858 |
| 59 | 9.00298543260603 | 59 | 8.6286020090823 |
| 60 | 9.13462953448214 | 60 | 8.70977844346484 |
| 61 | 9.63899819510826 | 61 | 8.78745290197109 |
| 62 | 9.90712989529353 | 62 | 8.8582114777092 |
| 63 | 10.184422975173 | 63 | 9.13621148101331 |
| 64 | 10.2549513369432 | 64 | 9.19613212493253 |
| 65 | 10.6340058195878 | 65 | 9.26193756168934 |
| 66 | 10.706919945127 | 66 | 9.33961202019559 |
| 67 | 10.8723535078713 | 67 | 9.3915056309117 |
| 68 | 11.2138079852279 | 68 | 9.92124341614671 |
| 69 | 11.6797278260842 | 69 | 10.1563173261699 |
| 70 | 11.8613138480383 | 70 | 10.2162379700891 |
| 71 | 12.3904276102571 | 71 | 10.2939124285954 |
| 72 | 12.4543451013282 | 72 | 10.3646710043335 |
| 73 | 12.5354730493851 | 73 | 10.4304764410903 |
| 74 | 12.6192921520948 | 74 | 10.8137460637386 |
| 75 | 12.8363781877171 | 75 | 10.8645505829005 |
| 76 | 12.9229381559026 | 76 | 10.9422250414067 |
| 77 | 13.0773401520405 | 77 | 11.0021456853259 |
| 78 | 13.3692704426633 | 78 | 11.054039296042 |
| 79 | 13.6019337083029 | 79 | 11.3881832939858 |
| 80 | 13.888368863783 | 80 | 11.4539887307426 |
| 81 | 13.9992065271618 | 81 | 11.5316631892488 |
| 82 | 14.1460684371958 | 82 | 11.602421764987 |
| 83 | 14.2740464065388 | 83 | 11.6623424089062 |
| 84 | 14.5646625688395 | 84 | 12.1920801941412 |
| 85 | 14.6997850868991 | 85 | 12.4700801974453 |
| 86 | 14.7637951213181 | 86 | 12.5477546559515 |
| 87 | 14.821402032386 | 87 | 12.7605590368452 |
| 88 | 14.9210827331781 | 88 | 12.826364473602 |

| | | | |
|---|---|---|---|
| 89 | 14.9806832778572 | 89 | 12.8862851175212 |
| 90 | 15.1686260826173 | 90 | 12.9381787282374 |
| 91 | 15.2343915552468 | 91 | 13.0158531867436 |
| 92 | 15.5269281656456 | 92 | 13.0866117624817 |
| 93 | 15.77763656073 | 93 | 13.2949773566009 |
| 94 | 15.8316908989359 | 94 | 13.386487515029 |
| 95 | 15.9997115307767 | 95 | 13.4464081589482 |
| 96 | 16.3980835779043 | 96 | 13.5240826174544 |
| 97 | 16.5312912221452 | 97 | 13.5898880542112 |
| 98 | 16.681746701436 | 98 | 14.1196258394463 |
| 99 | 16.9505406210983 | 99 | 14.5028954620946 |
| 100 | 17.0191016315102 | 100 | 14.5736540378327 |
| 101 | 17.3578549403357 | 101 | 14.6513284963389 |
| 102 | 17.8235398580112 | 102 | 14.7112491402581 |
| 103 | 17.8772293734815 | 103 | 14.7631427509742 |
| 104 | 18.2488783357456 | 104 | 14.8289481877311 |
| 105 | 18.4018233911568 | 105 | 15.1069481910352 |
| 106 | 18.6405126842018 | 106 | 15.1846226495414 |
| 107 | 18.7705976579948 | 107 | 15.5187666474851 |
| 108 | 18.8509937119911 | 108 | 15.5786872914044 |
| 109 | 19.1962556595445 | 109 | 15.802795695506 |
| 110 | 19.6279949719027 | 110 | 15.8735542712441 |
| 111 | 19.6830175238493 | 111 | 15.9512287297504 |
| 112 | 19.8189872850872 | 112 | 16.0170341665072 |
| 113 | 20.1263993794376 | 113 | 16.5467719517422 |
| 114 | 20.3677495877154 | 114 | 16.6066925956614 |
| 115 | 20.7015281494853 | 115 | 16.6585862063775 |
| 116 | 21.1693932816293 | 116 | 16.7362606648838 |
| 117 | 21.2303782666925 | 117 | 16.971334574907 |
| 118 | 21.287364934998 | 118 | 17.0475092642861 |
| 119 | 21.4092249955038 | 119 | 17.1133147010429 |

| | | | |
|---|---|---|---|
| 120 | 21.7676675396078 | 120 | 17.1732353449622 |
| 121 | 21.8701504420991 | 121 | 17.2509098034684 |
| 122 | 22.0414268838631 | 122 | 17.3216683792065 |
| 123 | 22.1079457368127 | 123 | 17.5996683825106 |
| 124 | 22.2673718853579 | 124 | 17.796038438113 |
| 125 | 22.4732761991034 | 125 | 17.8477469532281 |
| 126 | 22.825017593019 | 126 | 17.9242225698483 |
| 127 | 22.934671870189 | 127 | 17.9832195187665 |
| 128 | 22.9919901101517 | 128 | 18.5277498487795 |
| 129 | 23.3195461869013 | 129 | 18.5887656693552 |
| 130 | 23.5571629465436 | 130 | 19.1380764958593 |
| 131 | 23.6340152455018 | 131 | 19.2145521124794 |
| 132 | 23.7841617401838 | 132 | 19.5383549747129 |
| 133 | 23.9912288153016 | 133 | 19.5973519236311 |
| 134 | 24.0650555844926 | 134 | 19.9742222348526 |
| 135 | 24.1198799559854 | 135 | 20.0404422325498 |
| 136 | 24.1909951327269 | 136 | 20.1169178491699 |
| 137 | 24.3565567225032 | 137 | 20.2738186182675 |
| 138 | 24.4238769466392 | 138 | 20.7811946178557 |
| 139 | 24.6554056537035 | 139 | 20.8401915667738 |
| 140 | 24.7140489829383 | 140 | 21.3895023932779 |
| 141 | 24.7910806546673 | 141 | 21.4659780098981 |
| 142 | 25.2945104292732 | 142 | 21.7363336404517 |
| 143 | 25.3607653604478 | 143 | 22.2685531417578 |
| 144 | 25.4897892535251 | 144 | 22.3294591084073 |
| 145 | 25.551985336923 | 145 | 22.3883086143698 |
| 146 | 25.7361667386824 | 146 | 22.4642089969606 |
| 147 | 25.8424276091086 | 147 | 22.648493959746 |
| 148 | 26.3809552522386 | 148 | 22.7257305202693 |
| 149 | 26.5547475610971 | 149 | 23.2658105676403 |
| 150 | 26.7960096393214 | 150 | 23.3313228419482 |

| | | | |
|---|---|---|---|
| 151 | 26.8462539207113 | 151 | 23.8151616481374 |
| 152 | 27.2125922252148 | 152 | 23.8732199722772 |
| 153 | 27.2711664210898 | 153 | 24.3489978930447 |
| 154 | 27.3992653099649 | 154 | 24.6416616945928 |
| 155 | 27.473107558202 | 155 | 24.8170865021437 |
| 156 | 27.5276418656067 | 156 | 25.267436870111 |
| 157 | 27.8461966044484 | 157 | 25.7166242616846 |
| 158 | 28.3158367383619 | 158 | 26.1506757713713 |
| 159 | 28.7266805906962 | 159 | 26.603289161196 |
| 160 | 28.8395789229025 | 160 | 27.0467867151742 |
| 161 | 29.2391992216289 | 161 | 27.4432145344413 |
| 162 | 29.6852466614417 | 162 | 27.630152112103 |
| 163 | 29.8628812932926 | 163 | 28.0205050533827 |
| 164 | 30.4081426511018 | 164 | 28.4078266227899 |
| 165 | 30.7302087302324 | 165 | 28.5530280392686 |
| 166 | 31.1545111082076 | 166 | 28.9078109712398 |
| 167 | 31.2595029630489 | 167 | 29.0764598862369 |
| 168 | 31.6523105333144 | 168 | 29.1785118984206 |
| 169 | 31.9112018371682 | 169 | 29.5396314223002 |
| 170 | 31.9725583964969 | 170 | 29.8717416093335 |
| 171 | 32.4356865886082 | 171 | 30.0471805990577 |
| 172 | 32.6827838405316 | 172 | 30.0965338971616 |
| 173 | 32.7341243797258 | 173 | 30.4222054377009 |
| 174 | 32.9245849813172 | 174 | 30.7120448155123 |
| 175 | 33.2113072554083 | 175 | 30.8654416108308 |
| 176 | 33.2777098019366 | 176 | 31.0224272744579 |
| 177 | 33.3818002552757 | 177 | 31.1916746477543 |
| 178 | 33.6321722144179 | 178 | 31.4018592518599 |
| 179 | 33.7989474559372 | 179 | 31.6904102698686 |
| 180 | 34.0268748005878 | 180 | 31.8724248680123 |
| 181 | 34.3359762126407 | 181 | 32.023481068786 |

| | | | |
|---|---|---|---|
| 182 | 34.513428732113 | 182 | 32.1505868440006 |
| 183 | 34.6338717187007 | 183 | 32.3970179432611 |
| 184 | 34.9325038867679 | 184 | 32.4734956465896 |
| 185 | 35.0374238304603 | 185 | 32.5171674643877 |
| 186 | 35.0896087250312 | 186 | 32.7746460925895 |
| 187 | 35.3449765661295 | 187 | 32.907382497696 |
| 188 | 35.6336763014352 | 188 | 33.1780118300349 |
| 189 | 35.7088329263134 | 189 | 33.3256544817653 |
| 190 | 35.9060681547237 | 190 | 33.4272358135155 |
| 191 | 36.1647770403337 | 191 | 33.6320074032143 |
| 192 | 36.3196092587967 | 192 | 33.8376169778125 |
| 193 | 36.370131924959 | 193 | 33.9195748923237 |
| 194 | 36.4096429025663 | 194 | 34.0163853518102 |
| 195 | 36.7628290750501 | 195 | 34.1307422838599 |
| 196 | 36.9924519174457 | 196 | 34.254345823484 |
| 197 | 37.3014025395705 | 197 | 34.4191938744787 |
| 198 | 37.3859953743831 | 198 | 34.5041720618067 |
| 199 | 37.6164332947734 | 199 | 34.7023854178178 |
| 200 | 37.8066812695596 | 200 | 34.8089412875796 |
| 201 | 37.9352390900027 | 201 | 34.9196809725123 |
| 202 | 38.2559454789496 | 202 | 35.0005261809302 |
| 203 | 38.4699996480979 | 203 | 35.103365628259 |
| 204 | 38.5966733626147 | 204 | 35.1971095264953 |
| 205 | 38.6438429417696 | 205 | 35.2705020041623 |
| 206 | 38.7152797244389 | 206 | 35.4105191196444 |
| 207 | 38.8655649484913 | 207 | 35.5324528359688 |
| 208 | 39.0243289758112 | 208 | 35.6634150782393 |
| 209 | 39.2428486975696 | 209 | 35.7657184792737 |
| 210 | 39.5669005405967 | 210 | 35.8256242652728 |
| 211 | 39.9426138507598 | 211 | 35.9311744038559 |
| 212 | 40.1051984747419 | 212 | 35.9684778367236 |

| | | | |
|---|---|---|---|
| 213 | 40.4474485970684 | 213 | 36.0350611788179 |
| 214 | 40.6629055172138 | 214 | 36.1644403871498 |
| 215 | 40.7696892997564 | 215 | 36.1757146955832 |
| 216 | 40.935906802313 | 216 | 36.3345721619699 |
| 217 | 41.0476723275364 | 217 | 36.4631135639609 |
| 218 | 41.1615802797149 | 218 | 36.599398828448 |
| 219 | 41.3210783976633 | 219 | 36.7056002462248 |
| 220 | 41.3732990353842 | 220 | 36.8085780497984 |
| 221 | 41.488208676589 | 221 | 36.9320260151219 |
| 222 | 41.6069835745209 | 222 | 37.0226355370392 |
| 223 | 41.8737508320825 | 223 | 37.0747099351846 |
| 224 | 42.0133021671181 | 224 | 37.184144774287 |
| 225 | 42.1468074109381 | 225 | 37.2429730462499 |
| 226 | 42.3932967893669 | 226 | 37.31294791354 |
| 227 | 42.4883349999111 | 227 | 37.4025959094127 |
| 228 | 42.5591314499215 | 228 | 37.4666070574742 |
| 229 | 42.6662766402265 | 229 | 37.5272097289117 |
| 230 | 42.7156477722787 | 230 | 37.5641071794066 |
| 231 | 42.8867638094586 | 231 | 37.6586313367139 |
| 232 | 42.9275652814628 | 232 | 37.6717303249824 |
| 233 | 42.994756744755 | 233 | 37.7473493302083 |
| 234 | 43.1074336994031 | 234 | 37.7967737792352 |
| 235 | 43.1690608255855 | 235 | 37.8533994598107 |
| 236 | 43.2843828294264 | 236 | 37.9002458408428 |
| 237 | 43.3173302543937 | 237 | 37.9486550553922 |
| 238 | 43.4524904897245 | 238 | 38.0055992738541 |
| 239 | 43.522578927283 | 239 | 38.0373143543528 |
| 240 | 43.5588486215353 | 240 | 38.1221362866435 |
| 241 | 43.5843285146051 | 241 | 38.1733121042909 |
| 242 | 43.6689645392235 | 242 | 38.1942065253452 |
| 243 | 43.721298823597 | 243 | 38.2091784422713 |

| | | | |
|---|---|---|---|
| 244 | 43.748454416059 | 244 | 38.22993129201 |
| 245 | 43.8275132652613 | 245 | 38.2903669138692 |
| 246 | 43.9256269684656 | 246 | 38.3167568064688 |
| 247 | 43.9632049736821 | 247 | 38.3353957106592 |
| 248 | 44.1030940642861 | 248 | 38.3504990971747 |
| 249 | 44.1616502163374 | 249 | 38.3726861725745 |
| 250 | 44.2624407532841 | 250 | 38.4451304190845 |
| 251 | 44.295758575456 | 251 | 38.4933106802632 |
| 252 | 44.3479120978738 | 252 | 38.5547393849984 |
| 253 | 44.4006651975078 | 253 | 38.5717500826493 |
| 254 | 44.4602197827603 | 254 | 38.5876644875801 |
| 255 | 44.5062371697619 | 255 | 38.6255531573799 |
| 256 | 44.585952182964 | 256 | 38.6519430499794 |
| 257 | 44.6741784959299 | 257 | 38.6969235678287 |
| 258 | 44.7451163899939 | 258 | 38.7440257629379 |
| 259 | 44.8035602575894 | 259 | 38.7591702872054 |
| 260 | 44.8707675907697 | 260 | 38.8056094826341 |
| 261 | 44.9844083160898 | 261 | 38.841817705379 |
| 262 | 45.0875964566329 | 262 | 38.8582244189633 |
| 263 | 45.1316232709036 | 263 | 38.8724934585382 |
| 264 | 45.1717268045888 | 264 | 38.8942279641838 |
| 265 | 45.20284657854 | 265 | 38.9354396679439 |
| 266 | 45.2371424162066 | 266 | 38.9717815997435 |
| 267 | 45.2832537960374 | 267 | 38.9876640496082 |
| 268 | 45.418033247699 | 268 | 38.9999472234613 |
| 269 | 45.4329935220794 | 269 | 39.0112056001161 |
| 270 | 45.451864978416 | 270 | 39.0704699459604 |
| 271 | 45.5145126933095 | 271 | 39.094514224443 |
| 272 | 45.5344936329331 | 272 | 39.1126976437449 |
| 273 | 45.6021529902805 | 273 | 39.1495745161195 |
| 274 | 45.6568677340166 | 274 | 39.1639160652571 |

| | | | |
|---|---|---|---|
| 275 | 45.7027898110095 | 275 | 39.1996479518354 |
| 276 | 45.7643867884369 | 276 | 39.221760306762 |
| 277 | 45.7869692952583 | 277 | 39.2373228112843 |
| 278 | 45.8287412593528 | 278 | 39.2538436201205 |
| 279 | 45.8446208990245 | 279 | 39.2659265295902 |
| 280 | 45.9093106801201 | 280 | 39.3180175079686 |
| 281 | 45.961154285461 | 281 | 39.3377775686352 |
| 282 | 46.0282601818532 | 282 | 39.3589765524204 |
| 283 | 46.0658816532152 | 283 | 39.3750815702564 |
| 284 | 46.1040647498739 | 284 | 39.3863434241478 |
| 285 | 46.1942070549528 | 285 | 39.4072612884286 |
| 286 | 46.2382538773542 | 286 | 39.4270213490952 |
| 287 | 46.3104401002932 | 287 | 39.4418704693458 |
| 288 | 46.3361268597699 | 288 | 39.4535047326403 |
| 289 | 46.3509014566663 | 289 | 39.4744165352325 |
| 290 | 46.3964402362764 | 290 | 39.4916965410059 |
| 291 | 46.4117037467892 | 291 | 39.5114566016726 |
| 292 | 46.4359773533058 | 292 | 39.5271381241743 |
| 293 | 46.4795718311086 | 293 | 39.5427351334466 |
| 294 | 46.5227354227707 | 294 | 39.5624221816019 |
| 295 | 46.5351798202351 | 295 | 39.5827051243155 |
| 296 | 46.5670463708566 | 296 | 39.6024651849822 |
| 297 | 46.620351548184 | 297 | 39.622344679465 |
| 298 | 46.6431186767279 | 298 | 39.635580929611 |
| 299 | 46.6604762197316 | 299 | 39.6461001904693 |
| 300 | 46.6870946200464 | 300 | 39.6689270046354 |
| 301 | 46.7170217170828 | 301 | 39.6864445645964 |
| 302 | 46.7467419991638 | 302 | 39.698512920539 |
| 303 | 46.7966016897674 | 303 | 39.7086364838842 |
| 304 | 46.8234139192791 | 304 | 39.7191168681467 |
| 305 | 46.8550791087609 | 305 | 39.7323378271006 |

| | | | |
|---|---|---|---|
| 306 | 46.885514926083 | 306 | 39.7498553870616 |
| 307 | 46.898345983916 | 307 | 39.7602853887091 |
| 308 | 46.9100931574542 | 308 | 39.7736034506451 |
| 309 | 46.9314462745944 | 309 | 39.7845901472249 |
| 310 | 46.9427846277073 | 310 | 39.7972864826791 |
| 311 | 46.9608795035353 | 311 | 39.8109189695172 |
| 312 | 47.0008624916722 | 312 | 39.8260541550621 |
| 313 | 47.0130956412232 | 313 | 39.838869017855 |
| 314 | 47.0282980635615 | 314 | 39.8507741346886 |
| 315 | 47.0589741863341 | 315 | 39.8596308613873 |
| 316 | 47.0733362407301 | 316 | 39.8732633482254 |
| 317 | 47.0834494008695 | 317 | 39.8853048237113 |
| 318 | 47.102365054511 | 318 | 39.8939097268933 |
| 319 | 47.1163749329371 | 319 | 39.9064380576025 |
| 320 | 47.1370426181996 | 320 | 39.9202292704643 |
| 321 | 47.1756407009163 | 321 | 39.9338617573024 |
| 322 | 47.1839017898057 | 322 | 39.947126794935 |
| 323 | 47.2214792184186 | 323 | 39.9568277645577 |
| 324 | 47.2549657918181 | 324 | 39.9660261419174 |
| 325 | 47.2807216835909 | 325 | 39.978796649428 |
| 326 | 47.3036912191509 | 326 | 39.9924291362661 |
| 327 | 47.3473376596571 | 327 | 40.0044905501531 |
| 328 | 47.3773323946991 | 328 | 40.0194694386446 |
| 329 | 47.3860788899623 | 329 | 40.0305297016171 |
| 330 | 47.4248143294237 | 330 | 40.0395582039548 |
| 331 | 47.4701300164758 | 331 | 40.0531906907928 |
| 332 | 47.4799522811248 | 332 | 40.0660055535858 |
| 333 | 47.503735385367 | 333 | 40.0793236155217 |
| 334 | 47.5123996540767 | 334 | 40.0913320051053 |
| 335 | 47.5207120074667 | 335 | 40.1050878009805 |
| 336 | 47.5544083624328 | 336 | 40.1187202878186 |

| | | | |
|---|---|---|---|
| 337 | 47.5612736059034 | 337 | 40.1262761730871 |
| 338 | 47.5723040597976 | 338 | 40.1367061747346 |
| 339 | 47.5811346612276 | 339 | 40.1487830672072 |
| 340 | 47.5971932893083 | 340 | 40.159597988148 |
| 341 | 47.6205525159647 | 341 | 40.1732304749861 |
| 342 | 47.6371163136215 | 342 | 40.1873392860123 |
| 343 | 47.6568664712232 | 343 | 40.200008662162 |
| 344 | 47.6687331374075 | 344 | 40.2127791696727 |
| 345 | 47.6776803169268 | 345 | 40.2260972316086 |
| 346 | 47.6969453450891 | 346 | 40.2397297184467 |
| 347 | 47.7245685438289 | 347 | 40.2485864451454 |
| 348 | 47.7511797455424 | 348 | 40.2612827805996 |
| 349 | 47.7669653953159 | 349 | 40.2712640834213 |
| 350 | 47.7726037508081 | 350 | 40.2842352432676 |
| 351 | 47.804129407496 | 351 | 40.2978677301057 |
| 352 | 47.822019837363 | 352 | 40.310895248833 |
| 353 | 47.8313662433392 | 353 | 40.3193246266626 |
| 354 | 47.8412817706994 | 354 | 40.3286418743613 |
| 355 | 47.8494673183686 | 355 | 40.3394553849992 |
| 356 | 47.85747184552 | 356 | 40.3533355610122 |
| 357 | 47.8793070959423 | 357 | 40.3676293522928 |
| 358 | 47.9133791346325 | 358 | 40.3810213646451 |
| 359 | 47.9322911000602 | 359 | 40.3926615108903 |
| 360 | 47.941313509258 | 360 | 40.4008861137304 |
| 361 | 47.9469645086347 | 361 | 40.4147662897434 |
| 362 | 47.9529021944788 | 362 | 40.4232392945182 |
| 363 | 47.9791278590804 | 363 | 40.4306196544343 |
| 364 | 47.9901086189334 | 364 | 40.4436471731616 |
| 365 | 47.999025428686 | 365 | 40.4546169808528 |
| 366 | 48.0231258439002 | 366 | 40.4684971568659 |
| 367 | 48.0290456324025 | 367 | 40.4797564365729 |

| | | | |
|---|---|---|---|
| 368 | 48.0457441186891 | 368 | 40.490614302493 |
| 369 | 48.0631689896426 | 369 | 40.4992590829068 |
| 370 | 48.069467656893 | 370 | 40.5139077433081 |
| 371 | 48.0753975093762 | 371 | 40.5277879193211 |
| 372 | 48.0850739080339 | 372 | 40.5392562899273 |
| 373 | 48.1084510000757 | 373 | 40.5529813790295 |
| 374 | 48.1134593694733 | 374 | 40.5639511867208 |
| 375 | 48.1215281962915 | 375 | 40.5743823857376 |
| 376 | 48.1373591881859 | 376 | 40.5882625617507 |
| 377 | 48.1545565829881 | 377 | 40.6014982770496 |
| 378 | 48.1800582837698 | 378 | 40.6101430574633 |
| 379 | 48.1897770602778 | 379 | 40.6198019558229 |
| 380 | 48.1960625107107 | 380 | 40.6269741191674 |
| 381 | 48.2031778629665 | 381 | 40.6408542951805 |
| 382 | 48.2063868855439 | 382 | 40.65247896284 |
| 383 | 48.2132347639358 | 383 | 40.6671276232413 |
| 384 | 48.2326149819072 | 384 | 40.6779411338792 |
| 385 | 48.238503109886 | 385 | 40.6872583815779 |
| 386 | 48.2675146760599 | 386 | 40.701138557591 |
| 387 | 48.274765375333 | 387 | 40.7095679354206 |
| 388 | 48.2844339078177 | 388 | 40.7226205572378 |
| 389 | 48.2885069093355 | 389 | 40.7360125695901 |
| 390 | 48.2936755605228 | 390 | 40.7468704355102 |
| 391 | 48.312199295777 | 391 | 40.7607506115233 |
| 392 | 48.3183710728409 | 392 | 40.7720098912303 |
| 393 | 48.3368357793049 | 393 | 40.780482896005 |
| 394 | 48.3414354017487 | 394 | 40.7927093517453 |
| 395 | 48.3493578326883 | 395 | 40.8057368704726 |
| 396 | 48.3602188637481 | 396 | 40.8196170464857 |
| 397 | 48.3735330980707 | 397 | 40.8312417141452 |
| 398 | 48.384975080755 | 398 | 40.8415117623279 |

| | | | |
|---|---|---|---|
| 399 | 48.3930003293349 | 399 | 40.8521147369837 |
| 400 | 48.4016440535444 | 400 | 40.8599348698424 |

Lampiran 4. Data Jumlah *Dead Node*

| Iterasi | Jumlah *Dead Node* Algoritma *K-Means* | Iterasi | Jumlah *Dead Node* Algoritma *K-Means++* |
|---|---|---|---|
| 1 | 0 | 1 | 0 |
| 2 | 0 | 2 | 0 |
| 3 | 0 | 3 | 0 |
| 4 | 0 | 4 | 0 |
| 5 | 0 | 5 | 0 |
| 6 | 0 | 6 | 0 |
| 7 | 0 | 7 | 0 |
| 8 | 0 | 8 | 0 |
| 9 | 0 | 9 | 0 |
| 10 | 0 | 10 | 0 |
| 11 | 0 | 11 | 0 |
| 12 | 0 | 12 | 0 |
| 13 | 0 | 13 | 0 |
| 14 | 0 | 14 | 0 |
| 15 | 0 | 15 | 0 |
| 16 | 0 | 16 | 0 |
| 17 | 0 | 17 | 0 |
| 18 | 0 | 18 | 0 |
| 19 | 0 | 19 | 0 |
| 20 | 0 | 20 | 0 |
| 21 | 0 | 21 | 0 |
| 22 | 0 | 22 | 0 |
| 23 | 0 | 23 | 0 |
| 24 | 0 | 24 | 0 |
| 25 | 0 | 25 | 0 |
| 26 | 0 | 26 | 0 |
| 27 | 0 | 27 | 0 |

| | | | |
|---|---|---|---|
| 28 | 0 | 28 | 0 |
| 29 | 0 | 29 | 0 |
| 30 | 0 | 30 | 0 |
| 31 | 0 | 31 | 0 |
| 32 | 0 | 32 | 0 |
| 33 | 0 | 33 | 0 |
| 34 | 0 | 34 | 0 |
| 35 | 0 | 35 | 0 |
| 36 | 0 | 36 | 0 |
| 37 | 0 | 37 | 0 |
| 38 | 0 | 38 | 0 |
| 39 | 0 | 39 | 0 |
| 40 | 0 | 40 | 0 |
| 41 | 0 | 41 | 0 |
| 42 | 0 | 42 | 0 |
| 43 | 0 | 43 | 0 |
| 44 | 0 | 44 | 0 |
| 45 | 0 | 45 | 0 |
| 46 | 0 | 46 | 0 |
| 47 | 0 | 47 | 0 |
| 48 | 0 | 48 | 0 |
| 49 | 0 | 49 | 0 |
| 50 | 0 | 50 | 0 |
| 51 | 0 | 51 | 0 |
| 52 | 0 | 52 | 0 |
| 53 | 0 | 53 | 0 |
| 54 | 0 | 54 | 0 |
| 55 | 0 | 55 | 0 |
| 56 | 0 | 56 | 0 |
| 57 | 0 | 57 | 0 |
| 58 | 0 | 58 | 0 |

| | | | |
|---|---|---|---|
| 59 | 0 | 59 | 0 |
| 60 | 0 | 60 | 0 |
| 61 | 0 | 61 | 0 |
| 62 | 0 | 62 | 0 |
| 63 | 0 | 63 | 0 |
| 64 | 0 | 64 | 0 |
| 65 | 0 | 65 | 0 |
| 66 | 0 | 66 | 0 |
| 67 | 0 | 67 | 0 |
| 68 | 0 | 68 | 0 |
| 69 | 0 | 69 | 0 |
| 70 | 0 | 70 | 0 |
| 71 | 0 | 71 | 0 |
| 72 | 0 | 72 | 0 |
| 73 | 0 | 73 | 0 |
| 74 | 0 | 74 | 0 |
| 75 | 0 | 75 | 0 |
| 76 | 0 | 76 | 0 |
| 77 | 0 | 77 | 0 |
| 78 | 0 | 78 | 0 |
| 79 | 0 | 79 | 0 |
| 80 | 0 | 80 | 0 |
| 81 | 0 | 81 | 0 |
| 82 | 0 | 82 | 0 |
| 83 | 0 | 83 | 0 |
| 84 | 0 | 84 | 0 |
| 85 | 0 | 85 | 0 |
| 86 | 0 | 86 | 0 |
| 87 | 0 | 87 | 0 |
| 88 | 0 | 88 | 0 |
| 89 | 0 | 89 | 0 |

| | | | |
|---|---|---|---|
| 90 | 0 | 90 | 0 |
| 91 | 0 | 91 | 0 |
| 92 | 0 | 92 | 0 |
| 93 | 0 | 93 | 0 |
| 94 | 0 | 94 | 0 |
| 95 | 0 | 95 | 0 |
| 96 | 0 | 96 | 0 |
| 97 | 0 | 97 | 0 |
| 98 | 0 | 98 | 0 |
| 99 | 0 | 99 | 0 |
| 100 | 0 | 100 | 0 |
| 101 | 0 | 101 | 0 |
| 102 | 0 | 102 | 0 |
| 103 | 0 | 103 | 0 |
| 104 | 0 | 104 | 0 |
| 105 | 0 | 105 | 0 |
| 106 | 0 | 106 | 0 |
| 107 | 0 | 107 | 0 |
| 108 | 0 | 108 | 0 |
| 109 | 0 | 109 | 0 |
| 110 | 0 | 110 | 0 |
| 111 | 0 | 111 | 0 |
| 112 | 0 | 112 | 0 |
| 113 | 0 | 113 | 0 |
| 114 | 0 | 114 | 0 |
| 115 | 0 | 115 | 0 |
| 116 | 0 | 116 | 0 |
| 117 | 0 | 117 | 0 |
| 118 | 0 | 118 | 0 |
| 119 | 0 | 119 | 0 |
| 120 | 0 | 120 | 0 |

| | | | |
|---|---|---|---|
| 121 | 0 | 121 | 0 |
| 122 | 0 | 122 | 0 |
| 123 | 0 | 123 | 0 |
| 124 | 0 | 124 | 0 |
| 125 | 0 | 125 | 0 |
| 126 | 0 | 126 | 0 |
| 127 | 0 | 127 | 1 |
| 128 | 0 | 128 | 1 |
| 129 | 0 | 129 | 1 |
| 130 | 0 | 130 | 1 |
| 131 | 0 | 131 | 1 |
| 132 | 0 | 132 | 1 |
| 133 | 0 | 133 | 1 |
| 134 | 0 | 134 | 1 |
| 135 | 0 | 135 | 1 |
| 136 | 0 | 136 | 1 |
| 137 | 0 | 137 | 1 |
| 138 | 0 | 138 | 1 |
| 139 | 0 | 139 | 1 |
| 140 | 0 | 140 | 1 |
| 141 | 0 | 141 | 1 |
| 142 | 0 | 142 | 1 |
| 143 | 0 | 143 | 1 |
| 144 | 0 | 144 | 1 |
| 145 | 0 | 145 | 1 |
| 146 | 1 | 146 | 2 |
| 147 | 1 | 147 | 2 |
| 148 | 2 | 148 | 2 |
| 149 | 2 | 149 | 2 |
| 150 | 2 | 150 | 2 |
| 151 | 2 | 151 | 2 |

| | | | |
|---|---|---|---|
| 152 | 2 | 152 | 2 |
| 153 | 2 | 153 | 3 |
| 154 | 2 | 154 | 3 |
| 155 | 2 | 155 | 4 |
| 156 | 2 | 156 | 4 |
| 157 | 2 | 157 | 5 |
| 158 | 2 | 158 | 5 |
| 159 | 2 | 159 | 6 |
| 160 | 2 | 160 | 8 |
| 161 | 2 | 161 | 8 |
| 162 | 2 | 162 | 8 |
| 163 | 2 | 163 | 8 |
| 164 | 2 | 164 | 10 |
| 165 | 4 | 165 | 11 |
| 166 | 4 | 166 | 12 |
| 167 | 5 | 167 | 12 |
| 168 | 5 | 168 | 13 |
| 169 | 6 | 169 | 14 |
| 170 | 6 | 170 | 15 |
| 171 | 6 | 171 | 15 |
| 172 | 6 | 172 | 15 |
| 173 | 6 | 173 | 16 |
| 174 | 7 | 174 | 19 |
| 175 | 7 | 175 | 19 |
| 176 | 7 | 176 | 19 |
| 177 | 7 | 177 | 19 |
| 178 | 7 | 178 | 20 |
| 179 | 7 | 179 | 22 |
| 180 | 8 | 180 | 22 |
| 181 | 8 | 181 | 23 |
| 182 | 8 | 182 | 23 |

| | | | |
|---|---|---|---|
| 183 | 8 | 183 | 23 |
| 184 | 9 | 184 | 23 |
| 185 | 9 | 185 | 24 |
| 186 | 9 | 186 | 24 |
| 187 | 11 | 187 | 24 |
| 188 | 11 | 188 | 24 |
| 189 | 11 | 189 | 24 |
| 190 | 12 | 190 | 25 |
| 191 | 12 | 191 | 26 |
| 192 | 13 | 192 | 26 |
| 193 | 14 | 193 | 27 |
| 194 | 15 | 194 | 29 |
| 195 | 16 | 195 | 29 |
| 196 | 16 | 196 | 29 |
| 197 | 18 | 197 | 29 |
| 198 | 18 | 198 | 30 |
| 199 | 18 | 199 | 30 |
| 200 | 18 | 200 | 30 |
| 201 | 18 | 201 | 31 |
| 202 | 18 | 202 | 32 |
| 203 | 20 | 203 | 33 |
| 204 | 20 | 204 | 34 |
| 205 | 20 | 205 | 34 |
| 206 | 20 | 206 | 34 |
| 207 | 20 | 207 | 34 |
| 208 | 20 | 208 | 34 |
| 209 | 20 | 209 | 34 |
| 210 | 21 | 210 | 34 |
| 211 | 21 | 211 | 35 |
| 212 | 22 | 212 | 35 |
| 213 | 22 | 213 | 36 |

| | | | |
|---|---|---|---|
| 214 | 25 | 214 | 36 |
| 215 | 25 | 215 | 36 |
| 216 | 26 | 216 | 36 |
| 217 | 26 | 217 | 38 |
| 218 | 26 | 218 | 39 |
| 219 | 28 | 219 | 39 |
| 220 | 28 | 220 | 41 |
| 221 | 30 | 221 | 42 |
| 222 | 30 | 222 | 42 |
| 223 | 30 | 223 | 42 |
| 224 | 31 | 224 | 42 |
| 225 | 31 | 225 | 42 |
| 226 | 32 | 226 | 43 |
| 227 | 33 | 227 | 43 |
| 228 | 33 | 228 | 44 |
| 229 | 33 | 229 | 45 |
| 230 | 33 | 230 | 45 |
| 231 | 37 | 231 | 46 |
| 232 | 39 | 232 | 46 |
| 233 | 40 | 233 | 47 |
| 234 | 42 | 234 | 47 |
| 235 | 42 | 235 | 48 |
| 236 | 43 | 236 | 48 |
| 237 | 43 | 237 | 48 |
| 238 | 44 | 238 | 48 |
| 239 | 47 | 239 | 48 |
| 240 | 47 | 240 | 50 |
| 241 | 47 | 241 | 50 |
| 242 | 48 | 242 | 50 |
| 243 | 48 | 243 | 50 |
| 244 | 48 | 244 | 51 |

| | | | |
|---|---|---|---|
| 245 | 48 | 245 | 52 |
| 246 | 49 | 246 | 53 |
| 247 | 49 | 247 | 53 |
| 248 | 49 | 248 | 53 |
| 249 | 49 | 249 | 53 |
| 250 | 50 | 250 | 53 |
| 251 | 50 | 251 | 53 |
| 252 | 51 | 252 | 53 |
| 253 | 51 | 253 | 53 |
| 254 | 51 | 254 | 53 |
| 255 | 51 | 255 | 53 |
| 256 | 52 | 256 | 53 |
| 257 | 52 | 257 | 53 |
| 258 | 52 | 258 | 53 |
| 259 | 52 | 259 | 53 |
| 260 | 52 | 260 | 53 |
| 261 | 53 | 261 | 55 |
| 262 | 54 | 262 | 55 |
| 263 | 55 | 263 | 55 |
| 264 | 55 | 264 | 55 |
| 265 | 57 | 265 | 55 |
| 266 | 58 | 266 | 55 |
| 267 | 58 | 267 | 55 |
| 268 | 58 | 268 | 55 |
| 269 | 58 | 269 | 55 |
| 270 | 58 | 270 | 55 |
| 271 | 58 | 271 | 56 |
| 272 | 58 | 272 | 56 |
| 273 | 59 | 273 | 56 |
| 274 | 59 | 274 | 56 |
| 275 | 59 | 275 | 56 |

| | | | |
|---|---|---|---|
| 276 | 59 | 276 | 56 |
| 277 | 61 | 277 | 57 |
| 278 | 62 | 278 | 57 |
| 279 | 62 | 279 | 57 |
| 280 | 62 | 280 | 58 |
| 281 | 62 | 281 | 58 |
| 282 | 62 | 282 | 58 |
| 283 | 62 | 283 | 58 |
| 284 | 62 | 284 | 58 |
| 285 | 62 | 285 | 58 |
| 286 | 62 | 286 | 58 |
| 287 | 62 | 287 | 58 |
| 288 | 62 | 288 | 58 |
| 289 | 62 | 289 | 58 |
| 290 | 62 | 290 | 58 |
| 291 | 62 | 291 | 58 |
| 292 | 63 | 292 | 58 |
| 293 | 63 | 293 | 58 |
| 294 | 63 | 294 | 58 |
| 295 | 63 | 295 | 58 |
| 296 | 64 | 296 | 59 |
| 297 | 65 | 297 | 59 |
| 298 | 65 | 298 | 59 |
| 299 | 66 | 299 | 60 |
| 300 | 66 | 300 | 60 |
| 301 | 66 | 301 | 60 |
| 302 | 66 | 302 | 60 |
| 303 | 67 | 303 | 60 |
| 304 | 67 | 304 | 60 |
| 305 | 67 | 305 | 60 |
| 306 | 67 | 306 | 61 |

| | | | |
|---|---|---|---|
| 307 | 67 | 307 | 61 |
| 308 | 67 | 308 | 61 |
| 309 | 67 | 309 | 61 |
| 310 | 67 | 310 | 61 |
| 311 | 67 | 311 | 61 |
| 312 | 67 | 312 | 61 |
| 313 | 67 | 313 | 61 |
| 314 | 67 | 314 | 61 |
| 315 | 67 | 315 | 61 |
| 316 | 68 | 316 | 61 |
| 317 | 68 | 317 | 61 |
| 318 | 68 | 318 | 61 |
| 319 | 68 | 319 | 61 |
| 320 | 68 | 320 | 61 |
| 321 | 68 | 321 | 61 |
| 322 | 68 | 322 | 61 |
| 323 | 69 | 323 | 61 |
| 324 | 69 | 324 | 61 |
| 325 | 69 | 325 | 61 |
| 326 | 71 | 326 | 61 |
| 327 | 72 | 327 | 61 |
| 328 | 72 | 328 | 61 |
| 329 | 72 | 329 | 61 |
| 330 | 72 | 330 | 61 |
| 331 | 73 | 331 | 61 |
| 332 | 73 | 332 | 61 |
| 333 | 73 | 333 | 61 |
| 334 | 73 | 334 | 61 |
| 335 | 73 | 335 | 61 |
| 336 | 73 | 336 | 61 |
| 337 | 73 | 337 | 61 |

| | | | |
|---|---|---|---|
| 338 | 73 | 338 | 61 |
| 339 | 73 | 339 | 61 |
| 340 | 73 | 340 | 61 |
| 341 | 73 | 341 | 61 |
| 342 | 74 | 342 | 61 |
| 343 | 74 | 343 | 61 |
| 344 | 74 | 344 | 61 |
| 345 | 74 | 345 | 61 |
| 346 | 74 | 346 | 61 |
| 347 | 74 | 347 | 61 |
| 348 | 74 | 348 | 61 |
| 349 | 75 | 349 | 61 |
| 350 | 75 | 350 | 61 |
| 351 | 75 | 351 | 62 |
| 352 | 75 | 352 | 62 |
| 353 | 75 | 353 | 62 |
| 354 | 75 | 354 | 62 |
| 355 | 75 | 355 | 62 |
| 356 | 75 | 356 | 62 |
| 357 | 75 | 357 | 62 |
| 358 | 76 | 358 | 62 |
| 359 | 77 | 359 | 62 |
| 360 | 77 | 360 | 62 |
| 361 | 77 | 361 | 62 |
| 362 | 77 | 362 | 62 |
| 363 | 77 | 363 | 62 |
| 364 | 77 | 364 | 62 |
| 365 | 77 | 365 | 62 |
| 366 | 77 | 366 | 62 |
| 367 | 77 | 367 | 62 |
| 368 | 77 | 368 | 62 |

| | | | |
|---|---|---|---|
| 369 | 77 | 369 | 62 |
| 370 | 77 | 370 | 62 |
| 371 | 77 | 371 | 62 |
| 372 | 77 | 372 | 62 |
| 373 | 77 | 373 | 62 |
| 374 | 77 | 374 | 62 |
| 375 | 77 | 375 | 62 |
| 376 | 77 | 376 | 62 |
| 377 | 78 | 377 | 62 |
| 378 | 79 | 378 | 62 |
| 379 | 79 | 379 | 62 |
| 380 | 79 | 380 | 62 |
| 381 | 79 | 381 | 62 |
| 382 | 80 | 382 | 62 |
| 383 | 80 | 383 | 62 |
| 384 | 80 | 384 | 62 |
| 385 | 80 | 385 | 62 |
| 386 | 80 | 386 | 62 |
| 387 | 80 | 387 | 62 |
| 388 | 80 | 388 | 62 |
| 389 | 80 | 389 | 62 |
| 390 | 80 | 390 | 62 |
| 391 | 80 | 391 | 62 |
| 392 | 80 | 392 | 62 |
| 393 | 80 | 393 | 62 |
| 394 | 80 | 394 | 62 |
| 395 | 80 | 395 | 62 |
| 396 | 80 | 396 | 62 |
| 397 | 81 | 397 | 62 |
| 398 | 81 | 398 | 62 |
| 399 | 81 | 399 | 63 |

| | | | |
|---|---|---|---|
| 400 | 81 | 400 | 63 |

Lampiran 5. Data Jumlah *Node* Alive

| Iterasi | Jumlah *Node Alive* Algoritma *K-Means* | Iterasi | Jumlah *Node Alive* Algoritma *K-Means++* |
|---|---|---|---|
| 1 | 100 | 1 | 100 |
| 2 | 100 | 2 | 100 |
| 3 | 100 | 3 | 100 |
| 4 | 100 | 4 | 100 |
| 5 | 100 | 5 | 100 |
| 6 | 100 | 6 | 100 |
| 7 | 100 | 7 | 100 |
| 8 | 100 | 8 | 100 |
| 9 | 100 | 9 | 100 |
| 10 | 100 | 10 | 100 |
| 11 | 100 | 11 | 100 |
| 12 | 100 | 12 | 100 |
| 13 | 100 | 13 | 100 |
| 14 | 100 | 14 | 100 |
| 15 | 100 | 15 | 100 |
| 16 | 100 | 16 | 100 |
| 17 | 100 | 17 | 100 |
| 18 | 100 | 18 | 100 |
| 19 | 100 | 19 | 100 |
| 20 | 100 | 20 | 100 |
| 21 | 100 | 21 | 100 |
| 22 | 100 | 22 | 100 |
| 23 | 100 | 23 | 100 |
| 24 | 100 | 24 | 100 |
| 25 | 100 | 25 | 100 |
| 26 | 100 | 26 | 100 |
| 27 | 100 | 27 | 100 |

| | | | |
|---|---|---|---|
| 28 | 100 | 28 | 100 |
| 29 | 100 | 29 | 100 |
| 30 | 100 | 30 | 100 |
| 31 | 100 | 31 | 100 |
| 32 | 100 | 32 | 100 |
| 33 | 100 | 33 | 100 |
| 34 | 100 | 34 | 100 |
| 35 | 100 | 35 | 100 |
| 36 | 100 | 36 | 100 |
| 37 | 100 | 37 | 100 |
| 38 | 100 | 38 | 100 |
| 39 | 100 | 39 | 100 |
| 40 | 100 | 40 | 100 |
| 41 | 100 | 41 | 100 |
| 42 | 100 | 42 | 100 |
| 43 | 100 | 43 | 100 |
| 44 | 100 | 44 | 100 |
| 45 | 100 | 45 | 100 |
| 46 | 100 | 46 | 100 |
| 47 | 100 | 47 | 100 |
| 48 | 100 | 48 | 100 |
| 49 | 100 | 49 | 100 |
| 50 | 100 | 50 | 100 |
| 51 | 100 | 51 | 100 |
| 52 | 100 | 52 | 100 |
| 53 | 100 | 53 | 100 |
| 54 | 100 | 54 | 100 |
| 55 | 100 | 55 | 100 |
| 56 | 100 | 56 | 100 |
| 57 | 100 | 57 | 100 |
| 58 | 100 | 58 | 100 |

| | | | |
|---|---|---|---|
| 59 | 100 | 59 | 100 |
| 60 | 100 | 60 | 100 |
| 61 | 100 | 61 | 100 |
| 62 | 100 | 62 | 100 |
| 63 | 100 | 63 | 100 |
| 64 | 100 | 64 | 100 |
| 65 | 100 | 65 | 100 |
| 66 | 100 | 66 | 100 |
| 67 | 100 | 67 | 100 |
| 68 | 100 | 68 | 100 |
| 69 | 100 | 69 | 100 |
| 70 | 100 | 70 | 100 |
| 71 | 100 | 71 | 100 |
| 72 | 100 | 72 | 100 |
| 73 | 100 | 73 | 100 |
| 74 | 100 | 74 | 100 |
| 75 | 100 | 75 | 100 |
| 76 | 100 | 76 | 100 |
| 77 | 100 | 77 | 100 |
| 78 | 100 | 78 | 100 |
| 79 | 100 | 79 | 100 |
| 80 | 100 | 80 | 100 |
| 81 | 100 | 81 | 100 |
| 82 | 100 | 82 | 100 |
| 83 | 100 | 83 | 100 |
| 84 | 100 | 84 | 100 |
| 85 | 100 | 85 | 100 |
| 86 | 100 | 86 | 100 |
| 87 | 100 | 87 | 100 |
| 88 | 100 | 88 | 100 |
| 89 | 100 | 89 | 100 |

| | | | |
|---|---|---|---|
| 90 | 100 | 90 | 100 |
| 91 | 100 | 91 | 100 |
| 92 | 100 | 92 | 100 |
| 93 | 100 | 93 | 100 |
| 94 | 100 | 94 | 100 |
| 95 | 100 | 95 | 100 |
| 96 | 100 | 96 | 100 |
| 97 | 100 | 97 | 100 |
| 98 | 100 | 98 | 100 |
| 99 | 100 | 99 | 100 |
| 100 | 100 | 100 | 100 |
| 101 | 100 | 101 | 100 |
| 102 | 100 | 102 | 100 |
| 103 | 100 | 103 | 100 |
| 104 | 100 | 104 | 100 |
| 105 | 100 | 105 | 100 |
| 106 | 100 | 106 | 100 |
| 107 | 100 | 107 | 100 |
| 108 | 100 | 108 | 100 |
| 109 | 100 | 109 | 100 |
| 110 | 100 | 110 | 100 |
| 111 | 100 | 111 | 100 |
| 112 | 100 | 112 | 100 |
| 113 | 100 | 113 | 100 |
| 114 | 100 | 114 | 100 |
| 115 | 100 | 115 | 100 |
| 116 | 100 | 116 | 100 |
| 117 | 100 | 117 | 100 |
| 118 | 100 | 118 | 100 |
| 119 | 100 | 119 | 100 |
| 120 | 100 | 120 | 100 |

| | | | |
|---|---|---|---|
| 121 | 100 | 121 | 100 |
| 122 | 100 | 122 | 100 |
| 123 | 100 | 123 | 99 |
| 124 | 100 | 124 | 99 |
| 125 | 100 | 125 | 99 |
| 126 | 100 | 126 | 99 |
| 127 | 100 | 127 | 99 |
| 128 | 100 | 128 | 99 |
| 129 | 100 | 129 | 99 |
| 130 | 100 | 130 | 99 |
| 131 | 100 | 131 | 99 |
| 132 | 100 | 132 | 99 |
| 133 | 100 | 133 | 99 |
| 134 | 100 | 134 | 99 |
| 135 | 100 | 135 | 99 |
| 136 | 100 | 136 | 99 |
| 137 | 100 | 137 | 99 |
| 138 | 100 | 138 | 99 |
| 139 | 100 | 139 | 99 |
| 140 | 100 | 140 | 99 |
| 141 | 100 | 141 | 99 |
| 142 | 100 | 142 | 98 |
| 143 | 100 | 143 | 98 |
| 144 | 100 | 144 | 98 |
| 145 | 100 | 145 | 98 |
| 146 | 99 | 146 | 98 |
| 147 | 99 | 147 | 98 |
| 148 | 98 | 148 | 98 |
| 149 | 98 | 149 | 97 |
| 150 | 98 | 150 | 97 |
| 151 | 98 | 151 | 96 |

| | | | |
|---|---|---|---|
| 152 | 98 | 152 | 96 |
| 153 | 98 | 153 | 95 |
| 154 | 98 | 154 | 95 |
| 155 | 98 | 155 | 94 |
| 156 | 98 | 156 | 92 |
| 157 | 98 | 157 | 92 |
| 158 | 98 | 158 | 92 |
| 159 | 98 | 159 | 92 |
| 160 | 98 | 160 | 90 |
| 161 | 98 | 161 | 89 |
| 162 | 98 | 162 | 88 |
| 163 | 98 | 163 | 88 |
| 164 | 98 | 164 | 87 |
| 165 | 96 | 165 | 86 |
| 166 | 96 | 166 | 85 |
| 167 | 95 | 167 | 85 |
| 168 | 95 | 168 | 85 |
| 169 | 94 | 169 | 84 |
| 170 | 94 | 170 | 81 |
| 171 | 94 | 171 | 81 |
| 172 | 94 | 172 | 81 |
| 173 | 94 | 173 | 81 |
| 174 | 93 | 174 | 80 |
| 175 | 93 | 175 | 78 |
| 176 | 93 | 176 | 78 |
| 177 | 93 | 177 | 77 |
| 178 | 93 | 178 | 77 |
| 179 | 93 | 179 | 77 |
| 180 | 92 | 180 | 77 |
| 181 | 92 | 181 | 76 |
| 182 | 92 | 182 | 76 |

| | | | |
|---|---|---|---|
| 183 | 92 | 183 | 76 |
| 184 | 91 | 184 | 76 |
| 185 | 91 | 185 | 76 |
| 186 | 91 | 186 | 75 |
| 187 | 89 | 187 | 74 |
| 188 | 89 | 188 | 74 |
| 189 | 89 | 189 | 73 |
| 190 | 88 | 190 | 71 |
| 191 | 88 | 191 | 71 |
| 192 | 87 | 192 | 71 |
| 193 | 86 | 193 | 71 |
| 194 | 85 | 194 | 70 |
| 195 | 84 | 195 | 70 |
| 196 | 84 | 196 | 70 |
| 197 | 82 | 197 | 69 |
| 198 | 82 | 198 | 68 |
| 199 | 82 | 199 | 67 |
| 200 | 82 | 200 | 66 |
| 201 | 82 | 201 | 66 |
| 202 | 82 | 202 | 66 |
| 203 | 80 | 203 | 66 |
| 204 | 80 | 204 | 66 |
| 205 | 80 | 205 | 66 |
| 206 | 80 | 206 | 66 |
| 207 | 80 | 207 | 65 |
| 208 | 80 | 208 | 65 |
| 209 | 80 | 209 | 64 |
| 210 | 79 | 210 | 64 |
| 211 | 79 | 211 | 64 |
| 212 | 78 | 212 | 64 |
| 213 | 78 | 213 | 62 |

| | | | |
|---|---|---|---|
| 214 | 75 | 214 | 61 |
| 215 | 75 | 215 | 61 |
| 216 | 74 | 216 | 59 |
| 217 | 74 | 217 | 58 |
| 218 | 74 | 218 | 58 |
| 219 | 72 | 219 | 58 |
| 220 | 72 | 220 | 58 |
| 221 | 70 | 221 | 58 |
| 222 | 70 | 222 | 57 |
| 223 | 70 | 223 | 57 |
| 224 | 69 | 224 | 56 |
| 225 | 69 | 225 | 55 |
| 226 | 68 | 226 | 55 |
| 227 | 67 | 227 | 54 |
| 228 | 67 | 228 | 54 |
| 229 | 67 | 229 | 53 |
| 230 | 67 | 230 | 53 |
| 231 | 63 | 231 | 52 |
| 232 | 61 | 232 | 52 |
| 233 | 60 | 233 | 52 |
| 234 | 58 | 234 | 52 |
| 235 | 58 | 235 | 52 |
| 236 | 57 | 236 | 50 |
| 237 | 57 | 237 | 50 |
| 238 | 56 | 238 | 50 |
| 239 | 53 | 239 | 50 |
| 240 | 53 | 240 | 49 |
| 241 | 53 | 241 | 48 |
| 242 | 52 | 242 | 47 |
| 243 | 52 | 243 | 47 |
| 244 | 52 | 244 | 47 |

| | | | |
|---|---|---|---|
| 245 | 52 | 245 | 47 |
| 246 | 51 | 246 | 47 |
| 247 | 51 | 247 | 47 |
| 248 | 51 | 248 | 47 |
| 249 | 51 | 249 | 47 |
| 250 | 50 | 250 | 47 |
| 251 | 50 | 251 | 47 |
| 252 | 49 | 252 | 47 |
| 253 | 49 | 253 | 47 |
| 254 | 49 | 254 | 47 |
| 255 | 49 | 255 | 47 |
| 256 | 48 | 256 | 47 |
| 257 | 48 | 257 | 45 |
| 258 | 48 | 258 | 45 |
| 259 | 48 | 259 | 45 |
| 260 | 48 | 260 | 45 |
| 261 | 47 | 261 | 45 |
| 262 | 46 | 262 | 45 |
| 263 | 45 | 263 | 45 |
| 264 | 45 | 264 | 45 |
| 265 | 43 | 265 | 45 |
| 266 | 42 | 266 | 45 |
| 267 | 42 | 267 | 44 |
| 268 | 42 | 268 | 44 |
| 269 | 42 | 269 | 44 |
| 270 | 42 | 270 | 44 |
| 271 | 42 | 271 | 44 |
| 272 | 42 | 272 | 44 |
| 273 | 41 | 273 | 43 |
| 274 | 41 | 274 | 43 |
| 275 | 41 | 275 | 43 |

| | | | |
|---|---|---|---|
| 276 | 41 | 276 | 42 |
| 277 | 39 | 277 | 42 |
| 278 | 38 | 278 | 42 |
| 279 | 38 | 279 | 42 |
| 280 | 38 | 280 | 42 |
| 281 | 38 | 281 | 42 |
| 282 | 38 | 282 | 42 |
| 283 | 38 | 283 | 42 |
| 284 | 38 | 284 | 42 |
| 285 | 38 | 285 | 42 |
| 286 | 38 | 286 | 42 |
| 287 | 38 | 287 | 42 |
| 288 | 38 | 288 | 42 |
| 289 | 38 | 289 | 42 |
| 290 | 38 | 290 | 42 |
| 291 | 38 | 291 | 42 |
| 292 | 37 | 292 | 42 |
| 293 | 37 | 293 | 42 |
| 294 | 37 | 294 | 42 |
| 295 | 37 | 295 | 42 |
| 296 | 36 | 296 | 41 |
| 297 | 35 | 297 | 41 |
| 298 | 35 | 298 | 41 |
| 299 | 34 | 299 | 40 |
| 300 | 34 | 300 | 40 |
| 301 | 34 | 301 | 40 |
| 302 | 34 | 302 | 40 |
| 303 | 33 | 303 | 40 |
| 304 | 33 | 304 | 40 |
| 305 | 33 | 305 | 40 |
| 306 | 33 | 306 | 39 |

| | | | |
|---|---|---|---|
| 307 | 33 | 307 | 39 |
| 308 | 33 | 308 | 39 |
| 309 | 33 | 309 | 39 |
| 310 | 33 | 310 | 39 |
| 311 | 33 | 311 | 39 |
| 312 | 33 | 312 | 39 |
| 313 | 33 | 313 | 39 |
| 314 | 33 | 314 | 39 |
| 315 | 33 | 315 | 39 |
| 316 | 32 | 316 | 39 |
| 317 | 32 | 317 | 39 |
| 318 | 32 | 318 | 39 |
| 319 | 32 | 319 | 39 |
| 320 | 32 | 320 | 39 |
| 321 | 32 | 321 | 39 |
| 322 | 32 | 322 | 39 |
| 323 | 31 | 323 | 39 |
| 324 | 31 | 324 | 39 |
| 325 | 31 | 325 | 39 |
| 326 | 29 | 326 | 39 |
| 327 | 28 | 327 | 39 |
| 328 | 28 | 328 | 39 |
| 329 | 28 | 329 | 39 |
| 330 | 28 | 330 | 39 |
| 331 | 27 | 331 | 39 |
| 332 | 27 | 332 | 39 |
| 333 | 27 | 333 | 39 |
| 334 | 27 | 334 | 39 |
| 335 | 27 | 335 | 39 |
| 336 | 27 | 336 | 39 |
| 337 | 27 | 337 | 39 |

| | | | |
|---|---|---|---|
| 338 | 27 | 338 | 39 |
| 339 | 27 | 339 | 39 |
| 340 | 27 | 340 | 39 |
| 341 | 27 | 341 | 39 |
| 342 | 26 | 342 | 39 |
| 343 | 26 | 343 | 39 |
| 344 | 26 | 344 | 39 |
| 345 | 26 | 345 | 39 |
| 346 | 26 | 346 | 39 |
| 347 | 26 | 347 | 39 |
| 348 | 26 | 348 | 39 |
| 349 | 25 | 349 | 39 |
| 350 | 25 | 350 | 39 |
| 351 | 25 | 351 | 38 |
| 352 | 25 | 352 | 38 |
| 353 | 25 | 353 | 38 |
| 354 | 25 | 354 | 38 |
| 355 | 25 | 355 | 38 |
| 356 | 25 | 356 | 38 |
| 357 | 25 | 357 | 38 |
| 358 | 24 | 358 | 38 |
| 359 | 23 | 359 | 38 |
| 360 | 23 | 360 | 38 |
| 361 | 23 | 361 | 38 |
| 362 | 23 | 362 | 38 |
| 363 | 23 | 363 | 38 |
| 364 | 23 | 364 | 38 |
| 365 | 23 | 365 | 38 |
| 366 | 23 | 366 | 38 |
| 367 | 23 | 367 | 38 |
| 368 | 23 | 368 | 38 |

| | | | |
|---|---|---|---|
| 369 | 23 | 369 | 38 |
| 370 | 23 | 370 | 38 |
| 371 | 23 | 371 | 38 |
| 372 | 23 | 372 | 38 |
| 373 | 23 | 373 | 38 |
| 374 | 23 | 374 | 38 |
| 375 | 23 | 375 | 38 |
| 376 | 23 | 376 | 38 |
| 377 | 22 | 377 | 38 |
| 378 | 21 | 378 | 38 |
| 379 | 21 | 379 | 38 |
| 380 | 21 | 380 | 38 |
| 381 | 21 | 381 | 38 |
| 382 | 20 | 382 | 38 |
| 383 | 20 | 383 | 38 |
| 384 | 20 | 384 | 38 |
| 385 | 20 | 385 | 38 |
| 386 | 20 | 386 | 38 |
| 387 | 20 | 387 | 38 |
| 388 | 20 | 388 | 38 |
| 389 | 20 | 389 | 38 |
| 390 | 20 | 390 | 38 |
| 391 | 20 | 391 | 38 |
| 392 | 20 | 392 | 38 |
| 393 | 20 | 393 | 38 |
| 394 | 20 | 394 | 38 |
| 395 | 20 | 395 | 38 |
| 396 | 20 | 396 | 38 |
| 397 | 19 | 397 | 38 |
| 398 | 19 | 398 | 38 |
| 399 | 19 | 399 | 37 |

| | | | |
|---|---|---|---|
| 400 | 19 | 400 | 37 |