TUGAS AKHIR - KI141502

SINKRONISASI WAKTU KOMUNIKASI MENGGUNAKAN WEIGHTED ROUND-ROBIN PADA SENSOR PLATFORM TERDISTRIBUSI DAN SMARTPHONE GATEWAY UNTUK MONITORING LINGKUNGAN

RAYHAN GEMARUZMAN
NRP 05111440000174

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

Dosen Pembimbing II
Ir. F.X. Arunanto, M.Sc.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – KI141502

SINKRONISASI WAKTU KOMUNIKASI MENGGUNAKAN WEIGHTED ROUND-ROBIN PADA SENSOR PLATFORM TERDISTRIBUSI DAN SMARTPHONE GATEWAY UNTUK MONITORING LINGKUNGAN

RAYHAN GEMARUZMAN
NRP 05111440000174

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

Dosen Pembimbing II
Ir. F.X. Arunanto, M.Sc.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – KI141502

SYNCHRONIZATION OF COMMUNICATION TIME USING WEIGHTED ROUND-ROBIN ON DISTRIBUTED SENSOR PLATFORM AND SMARTPHONE GATEWAY FOR ENVIRONMENTAL MONITORING

RAYHAN GEMARUZMAN
NRP 05111440000174

First Advisor
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Second Advisor
Ir. F.X. Arunanto, M.Sc.

Department of Informatics
Faculty of Information and Communication Technology
Sepuluh Nopember Institute of Technology
Surabaya 2018

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

SINKRONISASI WAKTU KOMUNIKASI MENGGUNAKAN *WEIGHTED ROUND-ROBIN* PADA SENSOR PLATFORM TERDISTRIBUSI DAN SMARTPHONE *GATEWAY* UNTUK MONITORING LINGKUNGAN

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

RAYHAN GEMARUZMAN **NRP: 05111440000174**

Disetujui oleh Pembimbing Tugas Akhir:

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

2. Ir. F.X. Arunanto, M.Sc.
   (NIP. 195701011983031004) ..... (Pembimbing 2)

SURABAYA
JUNI, 2018

(Halaman ini sengaja dikosongkan)

SINKRONISASI WAKTU KOMUNIKASI MENGGUNAKAN *WEIGHTED ROUND-ROBIN* PADA SENSOR PLATFORM TERDISTRIBUSI DAN SMARTPHONE *GATEWAY* UNTUK MONITORING LINGKUNGAN

Nama Mahasiswa	**: Rayhan Gemaruzman**
NRP	**: 05111440000174**
Departemen	**: Informatika FTIK-ITS**
Dosen Pembimbing 1	**: Waskitho Wibisono, S. Kom., M.Eng., Ph.D.**
Dosen Pembimbing 2	**: Ir. F.X. Arunanto, M.Sc.**

Abstrak

Weighted Round Robin (WRR) merupakan pengembangan algoritma penjadwalan Round Robin dimana setiap node memiliki bobot dan mempengaruhi jadwal dari masing masing node yang ada di dalam arsitektur sistem. Pada tugas akhir ini, dibuat dua metode pendekatan WRR yang diimplementasikan ke dalam Wireless Sensor Network dan menggunakan protokol Bluetooth untuk komunikasi. Metode pendekatan yang dibuat adalah pendekatan waktu dan pendekatan jumlah giliran. Metode pendekatan waktu mengolah bobot node yang sudah didapatkan dan membuat waktu masing masing node untuk melakukan kerja sesuai dengan bobot yang didapatkan. Metode pendekatan jumlah giliran mempengaruhi jumlah giliran dalam satu waktu sebuah node untuk melakukan kerjanya.

Pada tugas akhir ini, WRR lebih efektif dalam mengurangi delay pengiriman dari node sensor ke Android gateway dibandingkan dengan Round Robin biasa. Hal ini

dibuktikan dengan delay WRR Pendekatan waktu sebesar 331,86 ms dan WRR pendekatan giliran sebesar 82,86 ms dibandingkan dengan metode Round Robin biasa dengan delay sebesar 478,95 ms.

Packet Delivery Ratio dari metode WRR di jarak 3m, 5m, dan 7m secara berturut turut adalah 99% – 100%; 98,87% – 99,64%; 95,95% – 99,1%. Round robin biasa sendiri secara berturut turut adalah 97,15%; 98,85%; dan 98,24%. Dari sini bisa dilihat bahwa WRR memiliki titik atas PDR lebih tinggi dibandingkan metode Round Robin biasa sehingga jarak efektif dari penggunaan Weighted Round Robin dalam arsitektur WSN adalah 3-5 m. Hal ini dibuktikan dengan PDR metode WRR sebesar 98,87% – 100%.

Yang terakhir adalah rata-rata delay pengiriman data kritikal pada metode WRR Waktu sebesar 15.719,5 ms atau 15,7 detik. Kemudian, metode WRR Giliran memiliki rata-rata delay 3497,8 ms atau 3,5 detik. Sedangkan, metode Round Robin memiliki rata rata delay sebesar 17.022,3 atau 17,02 detik.

Kata kunci: *Weighted Round-Robin*, WSN, Bluetooth

SYNCHRONIZATION OF COMMUNICATION TIME USING WEIGHTED ROUND-ROBIN ON DISTRIBUTED SENSOR PLATFORM AND SMARTPHONE GATEWAY FOR ENVIRONMENTAL MONITORING

Student's Name	**: Rayhan Gemaruzman**
Student's ID	**: 05111440000174**
Department	**: Informatika FTIK-ITS**
First Advisor	**: Waskitho Wibisono, S. Kom., M.Eng., Ph.D.**
Second Advisor	**: Ir. F.X. Arunanto, M.Sc.**

Abstract

Weighted Round Robin (WRR) is the development of the Round Robin scheduling algorithm where each node has a weight and affects the schedules of each node in the system architecture. In this final project, two methods of WRR approach are implemented into Wireless Sensor Network and use Bluetooth protocol for communication. Approach method that made is approach of time and approach of turn number. Approach time method process the weight of nodes that have been obtained and make the time of each node to do the work in accordance with the weight obtained. The turn number approximation method affects the number of turns at a time a node to do its work.

In this final project, WRR is more effective in reducing delivery delay from sensor node to Android gateway compared to regular Round Robin. This is evidenced by the WRR Approach time method delay is 331.86 ms and WRR

turn approach is 82.86 ms compared with the usual Round Robin method is 478.95 ms.

Packet Delivery Ratio of WRR method at 3m, 5m, and 7m respectively is 99% - 100%; 98.87% - 99.64%; 95.95% - 99.1%. The usual round robin is 97.15%; 98.85%; and 98.24%. From this it can be seen that WRR has a height point of PDR higher than the usual Round Robin method so that the effective distance from the use of Weighted Round Robin in the WSN architecture is 3-5 m. This is evidenced by PDR WRR method of 98.87% - 100%.

Finnaly, average delay of sending critical data on the WRR Time method is 15719.5 ms or 15.7 seconds. Then, the WRR Turn method has an average delay of 3497.8 ms or 3.5 seconds. Meanwhile, Round Robin method has an average delay of 17022.3 or 17.02 seconds.

***Keyword: Weighted Round-Robin*, WSN, Bluetooth**

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul:

"Sinkronisasi Waktu Komunikasi Menggunakan *Weighted Round Robin* pada Sensor Platform Terdistribusi dan Smartphone *Gateway* untuk Monitoring Lingkungan"

Harapan dari penulis semoga apa yang tertulis di dalam buku tugas akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini dan ke depannya, serta dapat memberikan kontribusi yang nyata.

Dalam pelaksanaan dan pembuatan tugas akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT.
2. Keluarga penulis (Mama, Papa, Chiqa, Alm. Nenek Lanang, Nenek Tino, Ibu Wowo, Om Andre, Uak Tem, Uak Nur, Goza, Mami Dewi, Om Dodi, Tante Lydia, Deva, Neo, Kimi, Aunty Inge, Om Kiki, Keang, Keyla, Om Andi, Tante Desy, Naura dan Keluarga penulis yang lain)yang selalu memberi dukungan berupa doa, moral ,dan materil yang tak terhingga kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.

3. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. dan Bapak Ir. F.X. Arunanto, M.Sc.. selaku dosen pembimbing penulis yang telah membimbing, memberikan nasihat, dan memotivasi penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
4. Bapak Dr.Eng. Darlis Herumurti, S.Kom., M.Kom. selaku kepala Departemen Informatika ITS.
5. Teman-teman dari Warkop x Jojoran (Afif, Faris, Aldo, Nezar, Gllenn, Sani, Botak, Lian, Buyung, Paul, Pentol, Oing, Dyo, Hilman, Fikry, Hamka, Faiq, Tras, Cimeng, Hari, Hakim, Upil, Fito, Anandi, Amik, Bajikas, Sekbay, Penyok, Kevin, Riefqy, Akhyar, Nanda, Dito, Rian, Nanda, Fathur, dan Petrus) yang selalu memberikan semangat, selalu memberikan hiburan kepada penulis, teman-teman yang sering diajak nongkrong, teman-teman yang bisa diajak untuk bertukar pikiran dan pendapat, dan juga menjadi keluarga kedua penulis saat berkuliah di Departemen Informatika ITS.
6. Teman teman THEREDSTC (Adi, Romi, Adib, Ichsan, Otong, Pur, Teja, Isye, Hanif, Haura) yang selalu memberikan semangat dan sarana berkumpul untuk menonton klub kebanggaan penulis Liverpool FC.
7. Sahabat-sahabat penulis di angkatan 2014 (Vivi, Kania, Bebet, Nay, Raras, Tiara, Pina, Anindita, Nafia ,Mila dan sahabat 2014 yang lain) yang telah menemani dan berjuang bersama penulis selama berkuliah di Departemen Informatika ITS.
8. Kabinet BEM FTIf Presisi Bermanfaat (Ovan, Irma, Fian, Sita, Septy, Faiz, Elva, Putra, Delia, Nana,

Hanif, Rika, DJ, Suhud, Nody, Nia, Afiif, Fata, Guntur, dan Dhevina) yang sudah memberikan penulis pembelajaran di bidang organisasi dan juga sebagai keluarga baru penulis.

9. Sahabat penulis Eci, Ibad, Galih, Dimas, Fachri, Wisnu, Pancek, Fara, Chang, Nikita dan yang tidak bisa disebutkan satu persatu yang selalu membantu, menghibur, menjadi tempat bertukar ilmu, serta berjuang bersama-sama dengan penulis.
10. Angkatan 2011, 2012, 2013, 2015, 2016 yang selalu memberikan pelajaran pelajaran berharga untuk hidup dan juga membuat penulis menjadi pribadi yang lebih baik.
11. Yang terakhir untuk orang-orang yang tidak dapat disebutkan oleh penulis, dan yang sudah membaca buku Tugas Akhir ini.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya. Semoga untuk semua orang dan terutama yang sudah membaca buku ini, dimanapun, kapanpun, dan bagaimanapun kondisinya penulis doakan selalu bahagia. Aamiin.

Surabaya, 6 Juni 2018

Rayhan Gemaruzman

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Informasi merupakan hal yang sangat krusial pada keadaan saat ini. Banyak hal-hal yang bergantung pada informasi. Contohnya adalah informasi perubahan keadaan sekitar seperti informasi tentang adanya kebakaran pada suatu hutan, informasi tentang adanya perubahan cuaca yang ekstrem dan sebagainya. Tentunya semakin cepat informasi didapatkan maka semakin cepat pula bagaimana penanganan terhadap informasi yang telah diperoleh. Tentunya untuk mendapatkan suatu informasi maka diperlukan sebuah beberapa data agar informasi yang kita peroleh menjadi valid. Contohnya adalah ketika ada kebakaran maka data yang diperlukan untuk mengidentifikasi kalau yang terjadi adalah kebakaran adalah intensitas api, suhu, dan adanya intensitas asap yang besar. Salah satu cara untuk mendapatkan berbagai data tersebut adalah dengan menggunakan sensor yang terkait dengan data tersebut.

Sensor adalah sesuatu yang digunakan untuk mendeteksi adanya perubahan lingkungan fisik atau kimia. Variabel keluaran dari sensor yang diubah menjadi besaran listrik disebut Transduser [1]. Tentunya hasil dari variable keluaran sensor tersebut akan di proses sehingga adanya klasifikasi dari hasil variable tersebut. Untuk membuat keluaran sensor dapat diproses dengan baik ,pemrosesan dibantu dengan mikrokontroller Arduino.

Mikrokontroller Arduino merupakan sebuah platform elektrokika open source dimana papan dari Arduino tersebut dapat membaca input dari sensor yang sudah dipasang dalam

papan Arduino [2]. Dari sensor yang sudah dipasang ke Arduino (*Node* Sensor) akan melakukan *sensing* sesuai dengan hasil programan dari Arduino. Jika ada *node* sensor lebih dari satu melakukan *sensing* di area yang sudah ditentukan kemudian, data yang didapat dari *node* sensor dikirimkan secara terus menerus, maka hal tersebut akan menguras tenaga dan beban jaringan.

Salah satu cara untuk mengurangi tenaga yang digunakan dan juga beban jaringan adalah melakukan sinkronisasi pengiriman data dan juga dengan menerapkan penjadwalan seperti *Round Robin*. Namun jika menggunakan *Round Robin*, proses monitoring dari *node* sensor akan menjadi kurang efektif dikarenakan jika ada *node* yang miliki urgensitas lebih, *node* tersebut tetap mendapatkan waktu *sensing* yang sama dengan *node* lainnya [3]. Namun ada pengembangan dari *Round Robin* untuk mengatasi masalah itu yaitu dengan *Weighted Round Robin*.

Weighted Round Robin adalah algoritma penjadwalan pengembangan dari *Round Robin* dengan memberikan jadwal kerja dari masing-masing *node* akan tetapi masing masing *node* memiliki penjadwalan yang sama seperti *Round Robin*.. Namun masing-masing *node* mempunyai bobot atau *weight* berdasarkan hasil data yang diperoleh. Algoritma ini berjalan dengan adanya bobot atau *weight* dimana *node* berjalan sesuai dengan pembobotan dari setiap *node* yang ada [4] [5].

Untuk itu dalam tugas akhir ini akan dilakukan sinkronisasi waktu untuk *node* sensor agar dapat melakukan kerjanya secara bergantian, mensinkronkan pengiriman data dari masing-masing *node*, dan juga mengetahui *node* sensor mana yang memiliki urgensitas lebih dengan menggunakan algoritma *Weighted Round Robin*.

1.2 Rumusan Masalah

Berikut beberapa hal yang menjadi rumusan masalah pada Tugas Akhir ini:

1. Bagaimana efektifitas *Weighted Round Robin* dalam *node* sensor?
2. Bagaimana efektifitas pengiriman data dari *Weighted Round Robin* dalam *node* sensor*?*
3. Bagaimana efektifitas penggunaan Bluetooth dalam *node* sensor?
4. Bagaimana penggunaan *Weighted Round Robin* dalam mengatasi data kritikal atau data yang urgen dari *node* sensor?

1.3 Batasan Permasalahan

Batasan masalah yang terdapat pada Tugas Akhir ini adalah sebagai berikut:

1. Modul sensor mikrokontroller menggunakan modul Arduino.
2. Bahasa yang digunakan adalah Bahasa C dengan library Arduino.
3. Menggunakan Arduino IDE sebagai workspace untuk pemrograman mikrokontroller arduino.
4. Komunikasi untuk mengiriman data melalui bluetooth.
5. Koordinator dari jaringan sensor tersebut adalah smartphone.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah mensinkronisasikan waktu komunikasi antar *node* sensor menggunakan *Weighted Round Robin* sehingga data yang didapat dari masing masing *node* sensor dapat diterima lebih cepat dan mengurangi beban jaringan dari arsitektur jaringan sensor.

1.5 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.5.1 Penyusunan Proposal Tugas Akhir

Tahapan awal dari Tugas Akhir ini adalah penyusunan Proposal Tugas Akhir. Proposal Tugas Akhir berisi pendahuluan, deskripsi, dan gagasan metode-metode yang dibuat dalam Tugas Akhir ini. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya Tugas Akhir, rumusan masalah yang diangkat, batasan masalah untuk Tugas Akhir, manfaat dan tujuan dari hasil pembuatan Tugas Akhir ini. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan Tugas Akhir.

1.5.2 Studi Literatur

Pada Studi literature ini akan dipelajari sejumlah refrensi yang diperlukan dalam pengerjaan Tugas Akhir ini yaitu mengenai *Wireless Sensor Network*, Bluetooth,

Android, Mikrokontroler Arduino, dan *Weighted Round-Robin.*

1.5.3 Implementasi Sistem

Implementasi merupakan tahap untuk mengimplementasikan metode-metode yang sudah diajukan pada proposal Tugas Akhir dengan menggunakan Bahasa pemrograman C/C++ untuk implementasi di Mikrokontroler Arduino, Bahasa Java untuk aplikasi Android sebagai koordinator dari pengiriman data, Arduino IDE sebagai *tools* pemrograman Arduino, dan Android Studio sebagai *tools* pemrograman aplikasi Android.

1.5.4 Pengujian dan Evaluasi

Pengujian akan dilakukan dua tipe pengujian yang pertama adalah yaitu uji performa untuk menguji performa pengiriman data dari *node* ke koordinator dan juga pengujian akurasi metode terhadap data yang dikirim. Kemudian yang kedua adalah uji fungsionalitas. Uji fungsionalitas meliputi uji fungsi dari aplikasi Android dan juga fungsionalitas Arduino terhadap fungsi yang dilakukan.

1.5.5 Penyusunan Buku

Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan Tugas Akhir yang mencakup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.6 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori atau penjelasan dari metode, algoritma, *library*, dan *tools* yang digunakan dalam penyusunan Tugas Akhir ini.
3. Bab III. Perancangan
   Bab ini berisi tentang perancangan metode yang nantinya akan diimplementasikan dan dilakukan uji coba.
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi yang berbentuk kode sumber dari proses pembuatan program untuk *node* mikrokontroler Arduino, kemudian penjelasannya beserta kode sumber untuk aplikasi Android.
5. Bab V. Uji Coba dan Evaluasi
   Bab ini berisikan hasil uji coba dan evaluasi hasil dari uji coba fungsionalitas dan performa.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.
7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam Tugas Akhir.
8. Lampiran

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar atau penjelasan dari metode dan *tools* yang digunakan dalam Tugas Akhir. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan riset yang berkaitan.

2.1 Jaringan Nirkabel

Jaringan Nirkabel (*Wireless Network*) merupakan bidang disiplin ilmu yang berkaitan dengan komunikasi antar system komputer tanpa menggunakan kabel. Jaringan nirkabel dapat diterapkan untuk komunikasi antar perangkat komputer baik jarak dekat maupun jarak jauh.

Jaringan nirkabel pertama yang dibuat yaitu jaringan nirkabel yang dibangun di Universitas Hawaii pada tahun 1971. Jaringan tersebut dibangun di empat pulau untuk menghubungkan perangkat komputer di pulau-pulau tersebut. Ide merambah ke ranah komputasi personal pada jaringan muncul sekitar tahun 1980-an sejak kegiatan pertukaran data menjadi cukup popular [6].

Jaringan nirkabel secara umum diatur dalam dokumen IEEE 802.11 dan termasuk di dalamnya Wi-Fi (*Wireless Fidelity*) yang diatur dalam IEEE 802.11b. Protokol IEE 802.11g adalah versi lainnya yang memiliki performa lebih cepat [7]. Pada perkembangannya diatur juga teknologi lain dalam IEEE 802.15 untuk WPAN (*Wireless Personal Area Network*). Beberapa diantarannya adalah Bluetooth (diatur dalam IEEE 802.15.1), Zigbee (802.15.4) dan sebagainya.

2.2 Jaringan Sensor Nirkabel

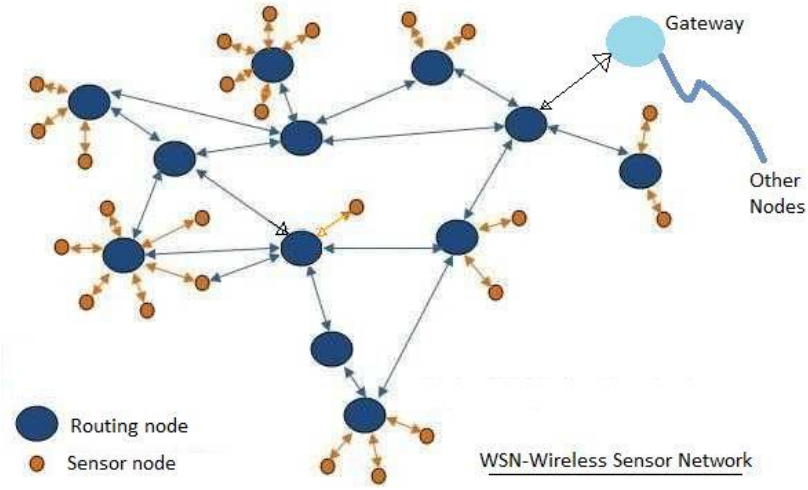
Jaringan sensor nirkabel yang dikenal dengan nama *Wireless Sensor Network* adalah salah satu jenis jaringan nirkabel yang berkerja dengan memanfaatkan *node-node* sensor dengan tujuan untuk melakukan proses sensor, monitoring, pengiriman data dan memberikan informasi kepada pengguna.

Wireless Sensor Network (jaringan sensor nirkabel) terdiri dari susunan dan kumpulan *node* - *node* sensor yang tersebar di suatu wilayah tertentu (*sensor field*) yang bekerja untuk mengambil data-data analog (*sensing*) dan mentranformasikan kedalam bentuk data digital agar data tersebut dapat di kelola dan menghasilkan suatu informasi mengenai lingkungan tersebut. *Node-node* pada jaringan sensor nirkabel selalu dilengkapi dengan dua komponen utama yang berfungsi untuk mendapatkan data dan informasi. Terdapat dua komponen utama dalam *node* jaringan sensor nirkabel yaitu sensor dan actuator.

1. Sensor dapat didefinisikan sebagai perangkat keras pada perangkat *node* yang berfungsi untuk melakukan pemindaian terhadap area *sensing*. Sensor bekerja menerima inputan dari rangsangan lingkungan tersebut. Sensor meliputi banyak jenis, antara lain kelembaban, radiasi, temperatur, tekanan, mekanik, gerakan, getaran, posisi, dan lain-lain. Setiap jenis sensor memiliki perangkat lunak (aplikasi, sistem operasi) dan perangkat keras masing-masing, yang kemudian akan digabungkan dan dijalankan ke dalam sistem *Wireless Sensor Network* (WSN) sedangkan actuator.
2. *Actuator* dapat di definisikan sebagai perangkat yang berfungsi untuk mengkonversi nilai inputan hasil penginderaan perangkat sensor hingga merubahnya

menjadi sebuah nilai variabel pada perangkat *node* WSN. Actuator juga melakukan pengiriman data antar *node* melalui perangkat nirkabel.

Masing–masing *node* dalam jaringan sensor nirkabel biasanya dilengkapi dengan radio transciever atau alat komunikasi nirkabel lainnya, mikrokontroler kecil, dan sumber energi ataupun baterai.



Gambar 2.1 Ilustrasi Arsitektur Jaringan Sensor Nirkabel (*Wireless Sensor Network*) [8].

Sebuah system pada Jaringan Sensor Nirkable terdapat 3 *node* utama diantaranya :

1. *Node* Sensor Berfungsi sebagai *node* yang melakukan proses sensor (memindai) terhadap lingkungan dimana WSN diimplementasikan untuk memperoleh sejumlah data yang kemudian dikirimkan ke server secara *online* melalui Internet.

2. *Node* Router Berfungsi untuk menentukan rute pengiriman paket data dari alamat pengirim ke alamat tujuan.
3. *Node Gateway* (*Sink Node*) Bertindak sebagai pintu gerbang, keluar masuknya paket data yang dikirimkan oleh *node* sensor dan diterima oleh komputer server(pusat).

Menurut Arati Manjeshwar dan Dharma P. Agrawal (2001) klasifikasi jaringan sensor nirkabel terbagi atas 2 yaitu:
a. Jaringan Proaktif adalah *node-node* pada jaringan ini secara berkala mengaktifkan sensor dan *transmitter*-nya untuk mendeteksi lingkungan dan mengirimkan data hasil *sensing*. Dengan demikian, jaringan proaktif dapat memberikan gambaran parameter yang bersangkutan secara berkala. Dan juga sangat cocok diaplikasikan pada kebutuhan monitoring data secara periode. Contohnya Algoritma LEACH yang bekerja secara periodik untuk mengirim data hasil *sensing* dan diwaktu lain menonaktifkan sensor dan *transmitter* untuk menyimpan energi.
b. Jaringan Reaktif adalah *node-node* pada jaringan akan bereaksi secara langsung apabila terjadi perubahan drastis ataupun tiba-tiba pada nilai atribut penginderaan. contohnya adalah algoritma TEEN yang bekerja ketika ada perubahan data secara drastis dan tidak melakukan pengiriman ketika data yang di peroleh masih sama [9].

2.3 Bluetooth

Bluetooth merupakan teknologi jaringan nirkabel dengan *low cost energy* yang digunakan untuk *streaming* Audio,

transfer data, dan broadcast informasi antar perangkat ada dua jenis dalam teknologi Bluetooth [10]:

1. *Basic Rate/Enhanced Data Rate* (BR/EDR)
   Untuk teknologi BR/EDR dapat membuat koneksi jaringan secara kontinu dan menggunakan *point-to-point* (P2P) sebagai topologi jaringannya yang membuat *one-to-one* (1:1) *device communication*. Untuk jenis teknologi ini biasanya di terapkan di speaker nirkabel , headset dan sebagainya.

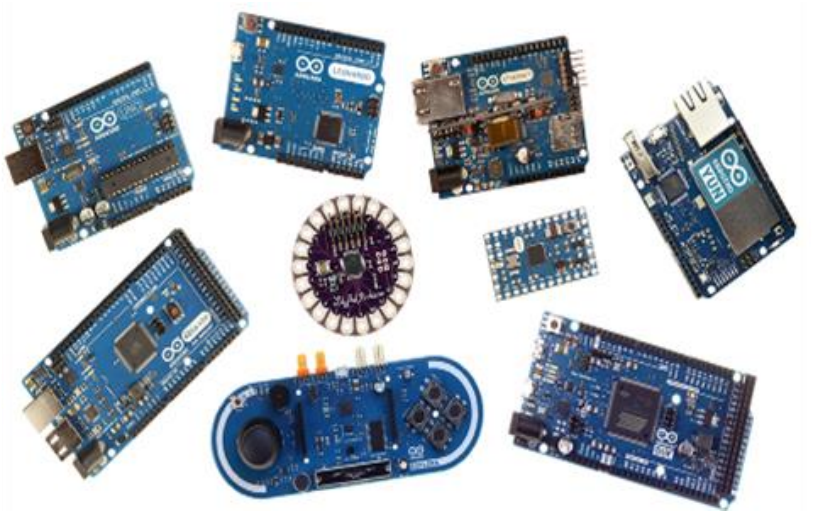
2. *Low Energy* (LE)
   Bluetooth LE dapat melakukan *short-burst* jaringan nirkabel dan menggunakan banyak topologi jaringan seperti, *Point-to-point* (*one-to-one*), *broadcast(one-to-many*), dan *mesh(many-to-many)* untuk Bluetooth Low Energy juga kompatibel dengan Arduino lewat komponen seperti HC-05, HC-06 dan juga seri HM.

Pada tugas akhir kali ini penulis akan menggunakan teknologi Bluetooth sebagai protokol komunikasi antar *node*. Spesifiknya adalah komunikasi antar *node* mikrokontroler Arduino terhadap *smartphone* Android sebagai koordinator antar *node*.

2.4 Arduino

Arduino adalah *kit* elektronik atau papan rangkaian elektronik *open source* yang di dalamnya terdapat komponen utama yaitu sebuah *chip* mikrokontroler dengan jenis AVR dari perusahaan Atmel. Mikrokontroler itu sendiri adalah chip atau IC (*integrated circuit*) yang bisa diprogram menggunakan komputer. Tujuan menanamkan program pada mikrokontroler

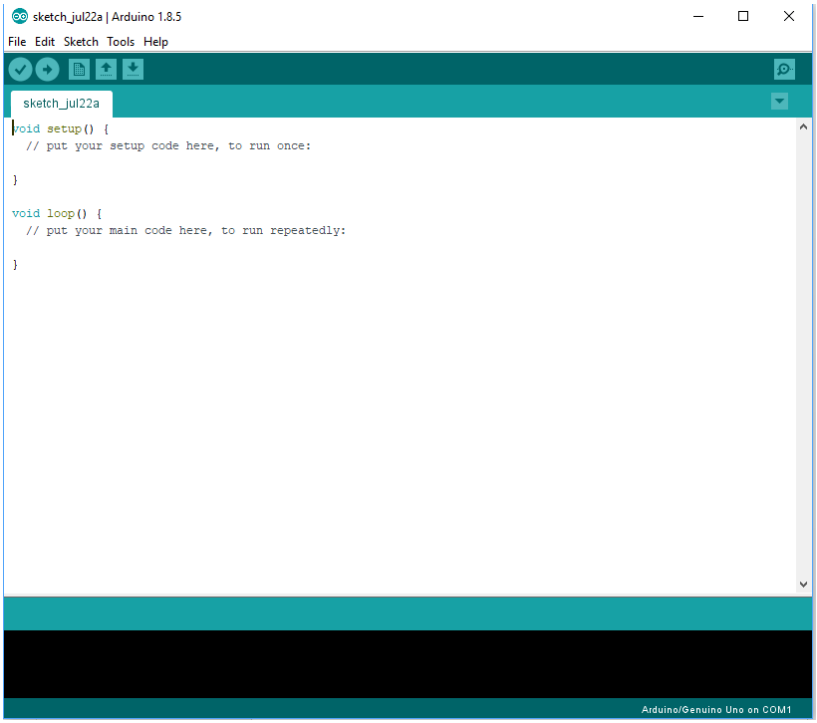
adalah agar rangkaian elektronik dapat membaca input, memproses input tersebut dan kemudian menghasilkan output sesuai yang diinginkan. Jadi mikrokontroler bertugas sebagai ‘otak’ yang mengendalikan *input*, proses dan *output* sebuah rangkaian elektronik. Mikrokontroler ada pada perangkat elektronik di sekitar. Misalnya *handphone*, MP3 player, DVD, televisi, AC, dll. Mikrokontroler juga dipakai untuk keperluan mengendalikan robot. Baik robot mainan, maupun robot industri. Karena komponen utama Arduino adalah mikrokontroler, maka Arduino pun dapat diprogram menggunakan komputer sesuai kebutuhan kita. [11]



Gambar 2.2 Jenis dan Bentuk Mikrokontroler Arduino [12].

Arduino IDE (*Integrated Development Environtment*) merupakan software yang digunakan untuk melakukan

pemrograman Arduino dan juga mengkoneksikan Arduino dan perangkat keras Genuino yang nantinya kode hasil program akan di upload ke perangkat keras Arduino. Dalam IDE ini terdapat workspace menulis kode, text console, dan juga beberapa fitur lainnya. Arduino IDE dibuat dari Bahasa pemrograman Java. Arduino IDE juga dilengkapi dengan pustaka (*library*) C dan C++. Perangkat lunak ini dikembangkan menggunakan proyek open source lain yaitu processing yang kemudian dirombak menjadi Arduino IDE itu sendiri [2] [13].

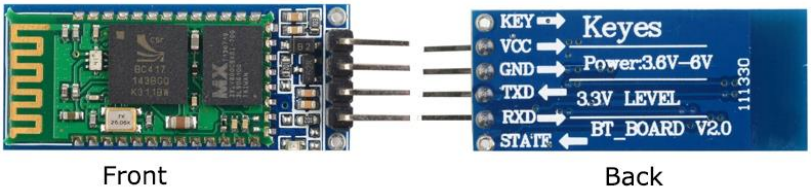


Gambar 2.3 Bentuk Tampilan Arduino IDE.

2.5 Modul Bluetooth HC-05

Modul HC-05 merupakan *Bluetooth SPP* (*Serial Port Protocol*) yang mudah digunakan. Modul SPP Bluetooth sudah terkualifikasi sepenuhnya dengan Bluetooth V2.0+EDR (*Enhanced Data Rate*) Modulasi 2.4 GHz radio *transceiver*. Menggunakan *CSR Bluecore 04-External single chip Bluetooth System* dengan teknologi CMOS dan AFH(*Adaptive Frequency Hopping Feature*). [14]

Modul Bluetooth *low-cost* ini kompatibel dengan Arduino dan mikrokomputer lain. HC-05 merupakan modul yang bisa di program menjadi *slave* ataupun *master*. Kemudian modul ini juga memiliki dua tipe operasi. Yang pertama adalah *Command Mode* dimana kita dapat mengirimkan *AT Command* dan yang kedua adalah *Data Mode* dimana modul tersebut dapat mentransmisikan dan menerima data ke modul Bluetooth yang lain. [15]



Gambar 2.4 Bentuk dari Modul HC-05 [16].

2.6 Modul Sensor Asap MQ-2

Modul sensor asap MO-2 merupakan modul sensor yang tebuat dari bahan yang memiliki konduktifitas rendah ketika berada di udara bersih. Bahan tersebut adalah *SnO2*. Ketika Sensor tersebut terkena gas yang mudah terbakar, maka

konduktifitas dari bahan tersebut akan tinggi. Sensor ini juga sangat sensitive terhadap beberapa zat seperti *LPG*, *Propana* dan *Hidrogen*, juga dapat digunakan untuk *Metana* dan uap yang mudah terbakar lainnya [16].

Gambar 2.5 Bentuk dari Modul MQ-02 [17]

2.7 Sistem Operasi Android

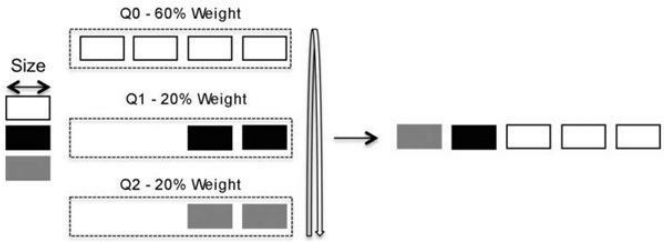
Android adalah sistem operasi berbasis Linux yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet. Android awalnya dikembangkan oleh Android. Inc dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya Open Handset Alliance, konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler [17].

Pada Tugas Akhir ini penulis akan membuat Smartphone Android menjadi koordinator komunikasi antar *node*.

2.8 Weighted Round-Robin

Round Robin merupakan salah satu algoritma untuk penjadwalan suatu kerja dari system. *Round Robin* menggunakan sebuah *Time Slice* (Sering disebut juga sebagai *Scheduling Quantum*) yaitu sebuah list urutan kerja dari suatu *node* dalam satuan waktu dan di lakukan secara loop atau berulang [19].

Weighted Round-Robin (WRR) merupakan algoritma penjadwalan jaringan berdasarkan pengembangan algoritma Round-Robin. Dalam disiplin ilmu jaringan, modifikasi dari WRR ditinjau dari penambahan request atau beban kerja terhadap *node* yang memiliki *weight* atau bobot yang lebih besar dibandingkan dengan bobot *node* lain [5].



Gambar 2.6 Ilustrasi dari *Weighted Round-Robin* [19]

Dalam Tugas Akhir ini penulis menggunakan ide dari algortima penjadwalan *Weighted Round-Robin* untuk di implementasikan kedalam arsitektur WSN. Dimana biasanya algoritma ini digunakan dalam *load balancing* server, ataupun manajemen paket dalam jaringan.

BAB III
PERANCANGAN

Bab ini membahas mengenai perancangan implementasi sistem yang dibuat pada Tugas Akhir. Bagian yang akan dijelaskan pada bab ini berawal dari deskripsi umum, Arsitektur Umum Sistem, Perancangan *node* sensor dan perancangan Android *gateway*

3.1 Deskripsi Umum

Pada Tugas Akhir ini penulis akan mengimplementasikan algoritma penjadwakan *Weighted Round-Robin* dalam sistem dimana nantinya yang akan menjadi koordinator dari Arsitektur sistem tersebut adalah *smartphone* Android. Untuk penerapannya akan ada 3 *node* sensor pada Arduino yang sudah di pasang dengan modul Bluetooth, kemudian masing-masing *node* sudah di *Pairing* dengan *Smartphone* Android. *Node* sensor akan mulai melakukan *sensing* ketika mendapat perintah dari *smartphone* Android. Perintah tersebut juga berisi waktu *sensing* dari *Node* sensor tersebut. kemudian setelah waktu *sensing* sesuai dari perintah smartphone tersebut sudah terpenuhi, *Node* sensor akan mengirimkan data berupa keterangan *node* pengirim, *level* atau bobot hasil data *sensing* yang sudah dihitung dan estimasi waktu *node* tersebut melakukan *sensing*.

Pada *node* sensor akan dihitung sebuah bobot atau tingkat urgensitas dari *node* tersebut yang nantinya akan di proses oleh Android *gateaway*. Berikut adalah perhitungan bobot dari masing masing *node*:

$$\boldsymbol{L = \left|\frac{n - \bar{x}}{n}\right|} \qquad \textbf{3.1}$$

Keterangan:

L = Level atau bobot dari setiap *node* yang telah melakukan *sensing.*
n = Titik normal dari data yang di deteksi.
$\bar{x}$ = Rata – rata dari hasil data yang telah di dapat sesuai dengan waktu *sensing* suatu *node*.

Untuk Android *gateway* nantinya akan ada 2 pendekatan *Weighted Round-Robin* yang akan di implementasikan. Pendekatan tersebut adalah pendekatan jumlah waktu dan pendekatan jumlah giliran. Pendekatan jumlah waktu adalah penentuan jumlah waktu *sensing* dari masing masing *node* berdasarkan bobot dengan perhitungan sebagai berikut:

$$\boldsymbol{T_x = \frac{L_x}{\Delta L} * Nt} \qquad \textbf{3.2}$$

Keterangan:

T_x = Waktu *node* x untuk melakukan *Sensing.*
L_x = *Level* atau bobot *node* x.
ΔL = Jumlah total semua bobot *node.*
Nt = Estimasi waktu total dalam satu putaran dimana terhitung satu putaran jika semua *node* sudah selesai melakukan *sensing*.

Kemudian untuk pendekatan jumlah giliran adalah penentuan jumlah giliran dalam satu ukuran waktu tertentu. Untuk pendekatan ini setiap *node* akan di berikan jatah untuk melakukan *sensing* sejumlah x dan untuk waktu *sensing* yang di

tentukan untuk setiap *node* sama. Jadi masing masing *node* memiliki waktu *Sensing* yang sama namun yang berbeda adalah jumlah giliran *node* tersebut untuk melakukan *sensing*. Untuk perhitungan jumlah giliran dari *node* ditentukan dengan perhitungan berikut:

$$\boldsymbol{S_x = \frac{L_x}{\Delta L} * Ns \ , S_x = \{0, 1, 2 \ldots\}} \qquad \textbf{3.3}$$

Keterangan:

S_x = jumlah giliran *node* x dalam satu putaran untuk melakukan *sensing* dengan catatan hasil perhitungan merupakan bilangan asli. Jika hasilnya bukan bilangan asli maka hasil perhitungan akan di bulatkan.

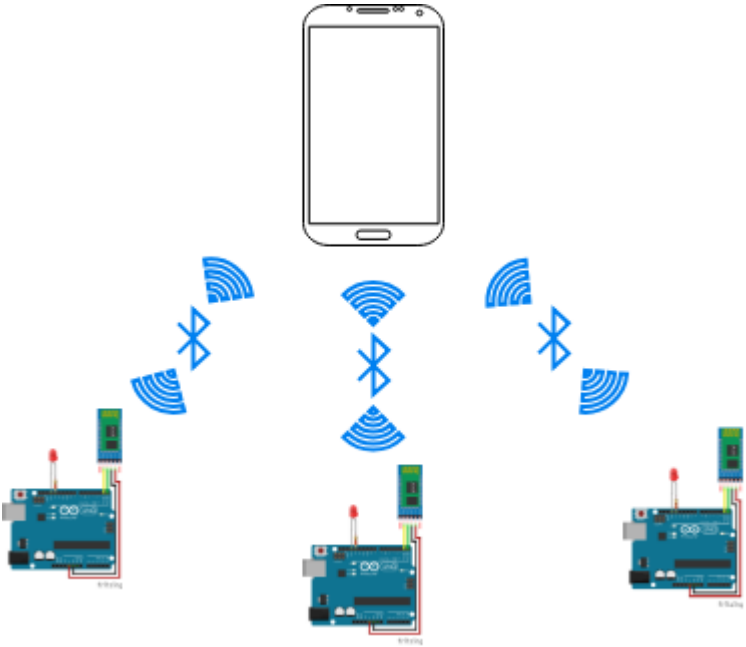
L_x = *Level* atau bobot *node* x.

ΔL = Jumlah total semua bobot *node*.

Ns = Total giliran dalam satu lap dimana terhitung satu putaran jika jumlah giliran setiap *node* mencapai total giliran.

3.2 Arsitektur Umum Sistem

Sistem yang akan di buat pada tugas akhir ini terdiri dari satu *smartphone* Android yang nantinya sebagai koordinator atau *master*, dan *node* sensor lebih dari satu sebagai *slave* yang akan mengirimkan data kepada Android *gateway* seperti pada Gambar 3.1.



Gambar 3.1 Arsitektur Umum Sistem.

Node sensor akan mengambil data berdasarkan lingkungan sekitarnya. Setelah data didapatkan, sensor menghitung level atau bobot dari *node* tersebut dan dikirimkan ke *smartphone* melalui Bluetooth. Dari sini *smartphone* akan meninjau dari data yang didapatkan. Kemudian *smartphone* akan mengirimkan perintah ke *node* selanjutnya untuk melakukan *sensing*, menghitung bobot, lalu datanya dikirim sama seperti *node* sebelumnya. Ketika semua *node* sudah mengirimkan semua bobotnya dan hasil data *sensing*, Android akan melakukan perbandingan bobot dari masing masing *node* yang nantinya akan menjadi acuan untuk menentukan prioritas dari masing masing *node* sensor.

3.3 Perancangan *Node* Sensor

Pada bagian ini akan dijelaskan mengenai rancangan dari *node* sensor. *Node* sensor akan dibangun menggunakan mikrokontroler Arduino UNO. Untuk komponen yang akan di gunakan adalah sensor asap MQ-2, LED sebagai penanda, kemudian HC-06 sebagai modul komunikasi nirkabel Bluetooth.

3.3.1 Perancangan Rangkaian Utama

Rangkaian dari *node* sensor membutuhkan beberapa komponen yang harus dipersiapkan. Berikut ini Tabel 3.1 menjabarkan komponen yang dibutuhkan beserta penjelasannya:

Tabel 3.1 Komponen Rangkaian

No	Nama Komponen	Deskripsi	Jumlah
1.	*Arduino UNO*	Merupakan sebuah mikrokontroler yang akan menjadi otak sekaligus mengatur berjalannya komponen lain. Tipe Arduino yang digunakan ini adalah tipe *UNO*	1 buah
2.	*HC-06*	Merupakan modul komunikasi nirkabel menggunakan Bluetooth. Modul ini bertindak sebagai *slave*.	1 buah
3.	*MQ-2*	Modul sensor asap sebagai sensor yang akan digunakan	1 buah
6.	*Breadboard*	Sebuah papan yang digunakan untuk membuat *prototype* rangkaian elektronik	1 buah

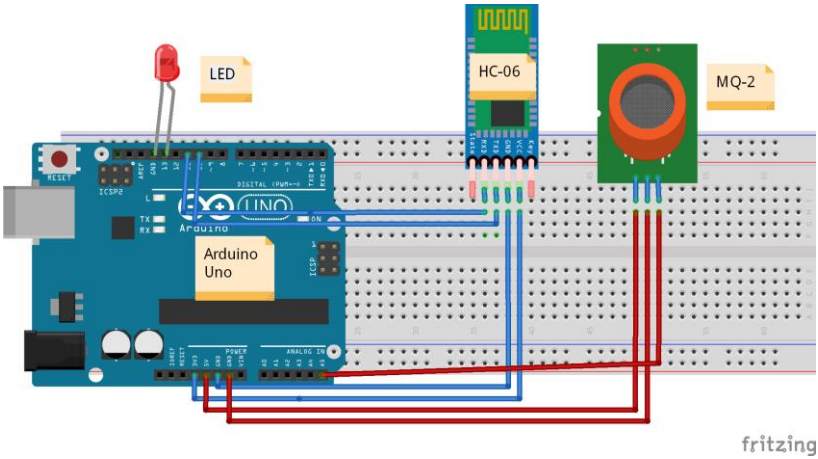
No	Nama Komponen	Deskripsi	Jumlah
7.	*Kabel Jumper*	Menghubungkan antara satu komponen dengan komponen lainnya	10 – 12 buah

Semua komponen yang ada, kemudian dirangkai menjadi satu. Gambar 3.2 merupakan gambaran rangkaian yang akan dibuat. Pada gambar tersebut, setiap komponen mulai dari mikrokontroler Arduino UNO, HC-06, MQ-2 dan LED dirangkai menjadi satu menggunakan kabel *jumper* dalam satu *breadboard*. Penjelasan mengenai *PIN* yang terhubung dengan Arduino akan dijelaskan pada Tabel 3.2.

Tabel 3.2 Koneksi PIN Arduino dengan Komponen Lain

PIN	Modul / Komponen	Deskripsi
POWER		
3.3V	*Breadboard (VCC)*	Sebagai *VCC*
GND	*Breadboard (GND)*	Sebagai *GROUND*
ANALOG IN		
A5	*MQ-02*	Terhubung dengan *PIN* pada sensor asap MQ-2
DIGITAL (PWM~)		
D10	*HC-06*	Berhubungan dengan *PIN* TX pada modul HC-06
D11	*Breadboard (HC-06)*	Berhubungan dengan *breadboard* untuk kemudian diteruskan agar tersambung dengan *PIN* RX pada modul HC-06
D13	*LED*	Terhubung dengan LED Sensor

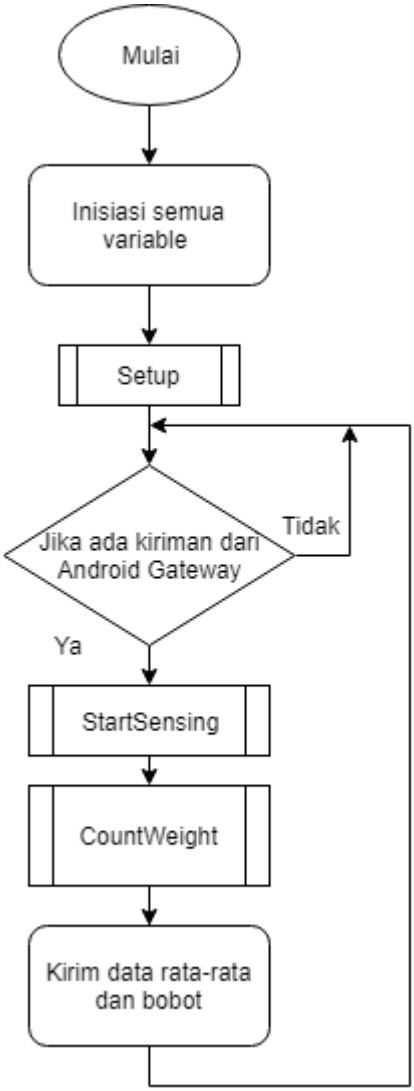
Rancangan rangakaian utama *node* sensor pada Gambar 3.2 adalah gambaran dari rangkaian yang akan diimplementasikan selanjutnya. Sensor MQ-2 akan digunakan untuk deteksi asap, yang kemudian akan dikirimkan ke Android *gateway* melalui modul Bluetooth HC-06.



Gambar 3.2 Rangkaian *Node* Sensor

3.3.2 Diagram Alur (*Node* Sensor)

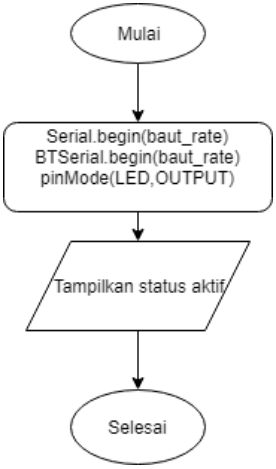
Pada subbab ini akan membahas tentang perancangan diagram alir dari *Node* sensor secara keseluruhan. Perancangan diagram alir ini juga memperlihatkan bagaimana aliran data dan fungsi fungsi yang nantinya akan di implementasikan dalam *node* sensor.



Gambar 3.3 Diagram alir *Node* Sensor

Gambar 3.3 merupakan diagram alir dari setiap *node* yang nantinya akan terhubung ke Android *gateway*. Dari diagram alir tersebut ada beberapa sub proses yaitu subproses *Setup*, *StartSensing* dan *CountWeight*. Sebelum memulai proses yang lainnya, variable akan diinisiasikan. Kemudian, *node* akan menunggu sampai ada kiriman dari Android *gateway*. Setelah mendapatkan kiriman dari Android *gateway* berupa estimasi waktu dari *node* tersebut melakukan giliran untuk *sensing, Node* sensor akan melakukan *sensing* sampai waktu yang sudah di tentukan oleh Android *gateway*. Setelah itu *node* akan melakukan perhitungan bobot berdasarkan data yang didapatkan dari hasil *sensing* dengan rumus 3.1. Selanjutnya *node* akan kembali mengirimkan rata-rata dari data hasil *sensing* dan juga bobot dari *node* tersebut yang nantinya akan di proses oleh Android *gateway*.

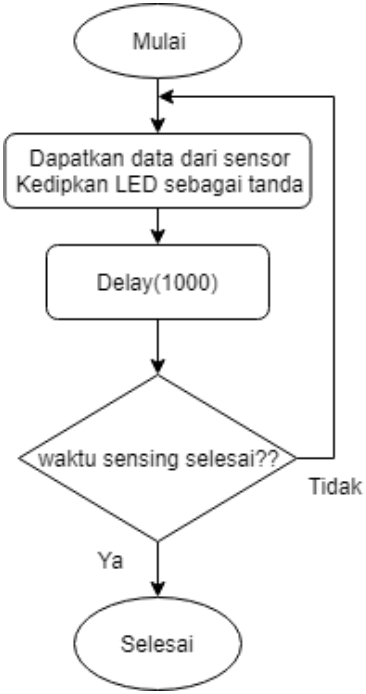
3.3.2.1 Diagram Alir Subproses Setup



Gambar 3.4 Diagram alir subproses Setup

Subproses setup adalah proses inisiasi untuk memulai jalannya *node* sensor. Pada subproses ini akan dijalankan sirkuit *Serial* dan Bluetooth. Perintah *Serial.begin(baud_rate)* dan *BTSerial.begin(baud_rate)* akan dipanggil sebagai inisiasi sirkuit *Serial*. kemudian *pinMode(LED,OUTPUT)* juga dipanggil sebagai inisiasi LED sebagai keluaran dari *Node* sensor tersebut. *Baud rate* yang digunakan adalah 9600 untuk semua sirkuit *Serial*. Gambar 3.4 merupakan diagram alir dari subproses *Setup*.

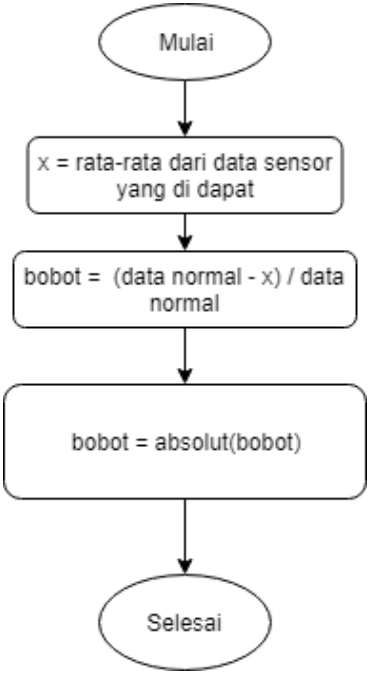
3.3.2.2 Diagram Alir Subproses *StartSensing*



Gambar 3.5 Diagram alir subproses *StartSensing*

Pada subproses ini *node* akan mendapatkan data dari sensor yang nantinya akan di proses pada subproses berikutnya. Kemudian LED akan di kedipkan sebagai tanda bahwa pada waktu tersebut adalah giliran dari *node* yang melakukan *sensing*. Kemudian *node* akan melakukan *delay* sebagai perhitungan waktu sesuai dengan estimasi yang diberikan oleh Android *Gateway*. Subproses ini akan di *Looping* sampai waktu giliran dari *node* tersebut selesai.

3.3.2.3 Diagram Alir Subproses *CountWeight*



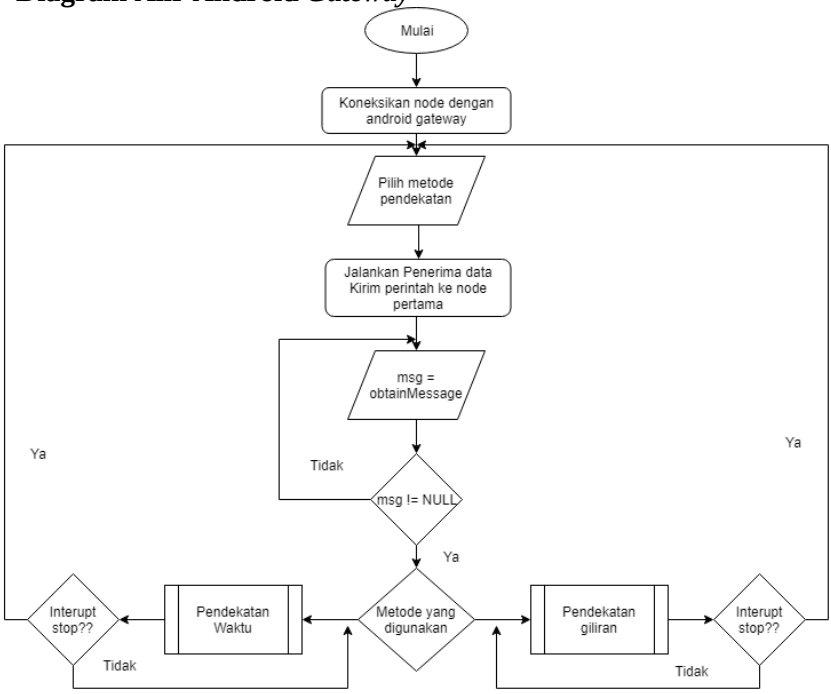
Gambar 3.6 Diagram alir subproses *CountWeight*.

Gambar 3.6 merupakan diagram alir dari subproses *CountWeight* dimana bobot atau level dari *node* sensor akan dihitung. Rata-rata data dari hasil *sensing* akan dihitung, kemudian akan diproses sesuai dengan rumus 3.1.

3.4 Perancangan Android *Gateway*

Pada subbab ini, akan dibahas alur dari sistem yang akan berjalan pada Android *gateway*. Perancangan aliran data sistem pada *gateway* secara keseluruhan dilakukan agar dapat lebih mudah memahami jalannya *gateway* secara menyeluruh.

3.4.1 Diagram Alir Android *Gateway*

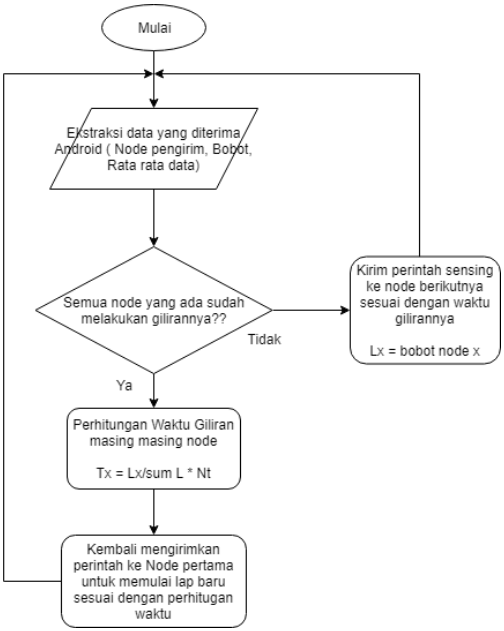


Gambar 3.7 Diagram alir Android *gateway* secara umum.

Android *Gateway* akan menggunakan salah satu metode pendekatan *Weighted Round Robin* yang di pilih. Kemudian Android akan menjalankan penerima data dan mengirimkan perintah sensing untuk kepada *node* pertama. Selanjutnya Android akan mengecek apakah ada kiriman data dari *node* yang di kirim yang nantinya data kiriman tersebut akan diproses sesuai dengan metode pendekatan *Weighted Round Robin* yang dipilih.

Gambar 3.7 merupakan diagram alir dari Android *gateway* secara umum. Untuk setiap metode pendekatan akan diproses melalui *message handling* di Android *gateway*.

3.4.1.1 Diagram Alir Subproses Metode Pendekatan Waktu



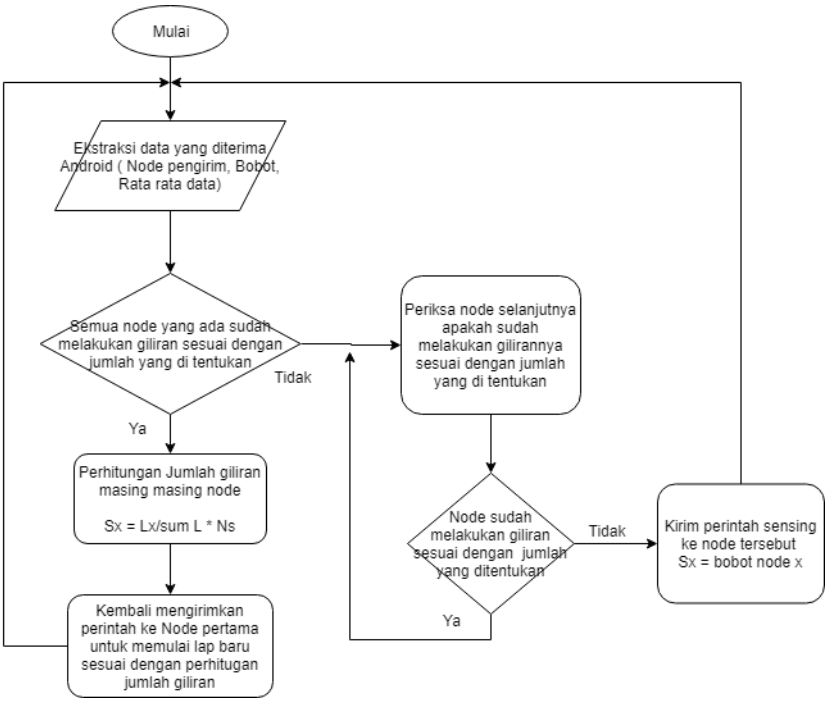
Gambar 3.8 Diagram alir subproses metode pendekatan waktu.

Gambar 3.8 merupakan diagram alir dari sub proses metode pendekatan waktu. Metode ini di mulai dari ekstraksi data yang di kirimkan oleh *node x.* Data ini berisi *node* pengirim, hasil perhitungan bobot dari *node* pengirim, kemudian rata-rata data dari hasil *sensing node* tersebut. Kemudian Android *gateway* akan memeriksa apakah semua *node* yang ada sudah melakukan giliran *sensing* sesuai dengan waktu yang ditentukan atau belum. Jika semua *node* belum melakukan gilirannya maka Android akan memerintahkan *node x+1* atau *node* selanjutnya untuk melakukan gilirannya. Namun jika semua *node* sudah melakukan gilirannya maka Android akan melakukan perhitungan waktu sesuai dengan rumus 3.2. Setelah hasil dari perhitungan didapatkan, Android akan kembali mengirimkan estimasi giliran sesuai dengan perhitungan kepada *node* pertama untuk memulai putaran yang baru.

3.4.1.2 Diagram Alir Subproses Metode Pendekatan Giliran

Untuk alur metode ini hampir menyerupai dari metode pendekatan waktu. Di metode ini memiliki perbedaan di pemeriksaan *node*-nya, Waktu yang di estimasikan, dan perhitungan menggunakan bobot yang diterima. Untuk pemeriksaan nodenya sendiri adalah dengan mengecek *node* tersebut apakah sudah memenuhi jumlah giliran yang ditentukan, sementara untuk metode pendekatan waktu tidak mengecek *node* nya namun langsung mengirim perintah ke *node* selanjutnya untuk melakukan *sensing*. Untuk waktu yang diestimasikan oleh Android *gateway* terhadap masing masing *node* disamakan namun yang membedakan adalah jumlah giliran dari masing masing *node*. Untuk perhitungan jumlah

giliran yang diestimasikan akan di hitung menggunakan rumus 3.3. Gambar 3.9 merupakan diagram alir dari metode pendekatan giliran.



Gambar 3.9 Diagram alir subproses metode pendekatan Giliran.

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Pada bab sebelumnya telah dijelaskan mengenai rancangan dari keseluruhan sistem yang akan dibangun. Setelah dilakukan proses perancangan, proses selanjutnya yang harus ditempuh adalah proses implementasi. Oleh karena itu, pada bab ini akan berisi penjelasan mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi berupa secara umum meliputi *pseudocode*, antarmuka aplikasi, rangkaian utama dan sebagainya.

4.1 Lingkungan Implementasi

Lingkungan implementasi merupakan suatu lingkungan dimana system akan dibangun. Pada bagian lingkungan implementasi ini, untuk mempermudahnya akan dibagi menjadi dua. Pembahasan yang pertama yaitu Lingkungan Implementasi Perangkat Keras dan Lingkungan Implementasi Perangkat Lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Pada bagian ini akan dibahas mengenai perangkat keras apa saja yang dibutuhkan untuk membangun sistem. Lingkungan implementasi perangkat keras dari sistem yang akan dibangun secara lebih lengkap dijelaskan pada Tabel 4.1 di bawah ini.

Tabel 4.1 Lingkungan Implementasi Perangkat Keras

Perangkat	Detail Perangkat
Perangkat Komputer	**Model :** • DELL Inspiron 3668 **Manufaktur :**

Perangkat	Detail Perangkat
	• Dell Inc. **Processor :** • Intel(R) Core(TM) i5-7400 CPU@3.00 GHz (4 CPU) **Memori :** • 8 GB **Sistem Operasi :** • Microsoft Windows 10 Pro 64-bit
Perangkat Mikrokontroler	**Mikrokontroler :** • Atmega328 **Model :** • Arduino UNO *Revision* 3 **Tegangan :** • 5 V **Memori *Flash* :** • 32 KB **SRAM :** • 2 KB **Sensor :** • Modul MQ-2 (Asap) ***Transceiver* :** • Model HC-06 (Bluetooth)
Perangkat Mobile (*Smartphone*)	**Model :** • ASUS Zenfone Max ZC550KL **Manufaktur :** • ASUSTeK Computer Inc. **Processor :** • Qualcomm Snapdragon 400/410 @1.21 GHz **Memori :**

Perangkat	Detail Perangkat
	• 2 GB **Berat :** • 202 gram **Dimensi :** • 156 x 77.5 x 10.6 mm **Lebar Layar :** • 5.5” **Resolusi :** • 720 x 1280 pixels **Bluetooth :** • 4.0 + EDR + A2DP **Baterai :** • 5000 mAh **Sistem Operasi :** • Android 6.0.1 (Marsmallow)

4.1.2 Lingkungan Implementasi Perangkat Lunak

Setelah mengetahui perangkat keras apa saja yang dibutuhkan, sekarang saatnya untuk menjabarkan perangkat lunak yang dibutuhkan. Pada bagian ini akan dibahas mengenai perangkat lunak apa saja yang dibutuhkan untuk membangun sistem. Lingkungan implementasi perangkat lunak dari sistem yang akan dibangun secara lebih lengkap dijelaskan pada Tabel 4.2 di bawah ini.

Tabel 4.2 Lingkungan Implementasi Perangkat Lunak.

Perangkat	Detail Perangkat
Perangkat Komputer	**Sistem Operasi :** • Microsoft Windows 10 Pro 64-bit

Perangkat	Detail Perangkat
	***Software* Arduino :** • Arduino IDE 1.8.5 ***Software* Android :** • Android Studio 2.2

4.2 Implementasi *Node* Sensor

Pada bab ini akan dijelaskan mengenai implementasi yang akan dilakukan untuk membangun *node* sensor. Beberapa hal yang akan di bahas disini adalah Rangkaian utama (perangkat keras) dari *node* sensor dan fungsi – fungsi (perangkat lunak).

4.2.1 Implementasi Rangkaian Utama

Node sensor ini akan diimplementasikan menggunakan mikrokontroler Arduino UNO R3. Mikrokontroler Arduino yang digunakan, akan dirangkai dan digabungkan dengan komponen – komponen yang lainnya. Sehingga dapat terbentuk sebuah alat sesuai apa yang telah dirancang sebelumnya yang dijabarkan pada Tabel 4.3.

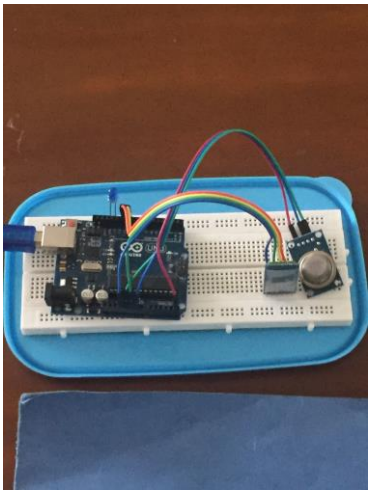
Tabel 4.3 Komponen Rangkaian Implementasi *Node* Sensor.

Perangkat Keras		Jumlah
Mikrokontroler	Arduino UNO Rev 3	Tiga buah
Sensor	MQ-2	Tiga buah
	LED	Tiga buah
Kabel	Kabel USB *Type A to Type B*	Tiga buah
	Kabel *jumper*	Tiga set (setiap set 8 – 10 buah)

Perangkat Keras		Jumlah
Board	*Breadboard*	Tiga buah
Transceiver	Modul Bluetooth HC-06	Tiga buah

Komponen akan di buat dan disusun seperti pada Gambar 4.1. modul HC-06 akan bertindak sebagai modul pertukaran data (komunikasi) yaitu melalui Bluetooth. Kemudian sensor yang yang di gunakan adalah sensor asap atau MQ-2. Sensor lain yang digunakan adalah LED sebagai penanda *node* sensor ketika sedang melakukan giliran *sensing*. LED akan berkedip sesuai dengan lama waktu dari *node* sensor tersebut untuk melakukan *sensing*.

Gambar 4.1 memperlihatkan rangkaian utama yang telah diimplementasikan berdasarkan pada bab 3.3.1



Gambar 4.1 implementasi Rangkaian Utama *Node* Sensor.

Komponen atau perangkat keras yang digunakan tiap *node*, sama seperti yang lainnya. Implementasi terhadap

perakitan komponen menghasilkan rangkaian yang sesuai dengan apa yang telah dirancang sebelumnya.

4.2.2 Implementasi Fungsi (*Node* Sensor)

Pada bagian ini akan dijabarkan mengenai implementasi perangkat lunak *node* sensor. Implementasi yang dimaksud berupa kode program yang dibuat dalam bentuk fungsi – fungsi tertentu. Bahasa pemrograman yang digunakan untuk implementasi *node* sensor adalah Arduino.

Bahasa pemrograman Arduino merupakan bahasa pemrograman yang digunakan untuk mengendalikan komponen – komponen menggunakan mikrokontroler Arduino. Ada beberapa fungsi utama yang akan di implementasikan ke dalam Arduino, yaitu *setup, loop, startSensing,* dan *countWeight.* Kemudian untuk setiap *node* yang ada dalam arsitektur memiliki kode sumber yang sama hanya saja dalam implementasinya nanti yang membedakan adalah variable *node* sensor untuk menandakan masing masing *node* sensor yang ada di dalam arsitektur system. Berikut adalah penjelasannya.

4.2.2.1 Fungsi *setup*

Fungsi *setup* merupakan fungsi yang digunakan untuk inisialisasi awal yang merujuk pada perancangan di bab 3.3.2.1 Fungsi ini dijalankan hanya saat mikrokontroler baru dihidupkan (tersambung dengan sumber listrik). Kode Sumber 4.1 merupakan penjabaran mengenai fungsi *setup* dan inisialisasi variabel global yang nantinya akan dipakai oleh *node* sensor untuk fungsi fungsi yang lainnya dan juga inisiasi pin yang akan dipakai oleh Bluetooth modul.

```
1  initialize SoftwareSerial BT(10, 11)
2  initialize all sensor pin
```

```
3  initialize all sensor variables
4  initialize idSensor
5
6  FUNCTION setup()
7    open Serial.begin(BAUD_RATE_9600)
8    open BTSerial.begin(BAUD_RATE_9600)
9    open Pinmode(LED,OUTPUT)
   ENDFUNCTION
```

Kode Sumber 4.1 *Pseudocode* Fungsi *setup* dan Inisialisasi Variabel Global.

4.2.2.2 Fungsi *loop*

Fungsi *loop* merupakan fungsi yang akan terus dilakukan seiring waktu selama mikrokontroler mendapat pasokan energi listrik. Didalam fungsi ini *node* sensor akan terus melakukan loop sampai ada perintah dari Android *gateway* untuk melakukan perintah. Jika ada perintah atau kiriman data dari Android *gateway* maka *node* sensor akan melakukan pekerjaannya sesuai dengan perintah Android, pekerjaan yang dimaksud merupakan *sensing*. Setelah pekerjaan dari *node* sensor tersebut sudah sesuai dengan perintah Android *gateway* maka *node* akan mengirimkan bobot, dan hasil *sensing* tersebut ke Android *gateway*. Kode Sumber 4.2 merupakan pseudocode implementasi fungsi loop dari *node* sensor.

```
1  initialize all sensor variables
2  FUNCTION loop()
3    if BTSerial.available
4       set timemonitor ← BTSerial.read()
5       call clockStart(timemonitor)
6       call countWeight()
```

```
7-8   set dump ← (String) idSensor + "#" + "OK#" + timemonitor + "#" + weight + "\n"
      call BTSerial.print(dump.c_str())
      ENDFUNCTION
```

Kode Sumber 4.2 *Pseudocode* Fungsi *loop.*

4.2.2.3 Fungsi *startSensing*

Fungsi *startSensing* merupakan fungsi yang digunakan untuk memulai sensing dari *node* sensor. Di dalam fungsi ini akan di isi variable *timemonitor* yang merupakan waktu untuk melakukan *sensing* yang sudah di tentukan oleh Android *gateway*. Ketika fungsi tersebut dipanggil maka akan terjadi *looping* sampai waktu yang di tentukan sudah terpenuhi. Untuk membuat waktu tersebut sesuai dengan ukuran waktu seperti umumnya, akan digunakan *delay* selama satu detik di dalam fungsi ini.

Didalam fungsi ini juga akan dilakukan pembacaan data dari pin input sensor yang sudah dipasang, yaitu sensor asap atau MQ-2. Difungsi ini juga membuat LED yang terpasang dalam *node* sensor berkedip untuk memastikan bahwa *node* tersebut sedang melakukan *sensing*. Kode Sumber 4.3 merupakan penjabaran dari fungsi *startSensing*.

```
1  initialize all sensor variable
2  FUNCTION startSensing(int timemonitor)
3    initialize reading[]
4    for i = 0 to timemonitor , step =1 do
5      delay(500)
6      set reading[i] ← analogread(smokePin)
7      digitalwrite(LED,High)
8      delay(500)
```

```
9     digitalwrite(LED,Low)

    ENDFUNCTION
```

Kode Sumber 4.3 *Pseudocode* Fungsi *startSensing*.

4.2.2.4 Fungsi *countWeight*

Pada fungsi ini akan di hitung bobot atau *weight* dari *node* sensor yang sudah melakukan *sensing*. Fungsi ini dijalan kan setelah *node* sudah melakukan *sensing* sesuai dengan variable *timemonitor* yang sudah di tentukan oleh smartphone. Disini juga akan dihitung rata rata dari data yang telah di dapatkan dan sudah di simpan di dalam variable *array reading* dalam fungsi *startSensing*.

Didalam fungsi ini juga akan dipanggil fungsi *countAverage* untuk mendapatkan rata-rata dari data yang didapatkan. Untuk perhitungan dari fungsi *countWeight* sesuai dengan rumus 3.1. Kode Sumber 4.4 merupakan pseudocode dari fungsi *countWeight.*

```
1   initialize all sensor variables
2   FUNCTION (float) countAverage()
3     set Sum ← 0
4     for k = 0 to timemonitor, step = 1 do
5       Sum += reading[k]
6     set average ← (float) sum/timemonitor
7     return average
8
9   FUNCTION countWeight()
10    set xvalue  ← call countAverage()
11    set weight ← (normal – xvalue)/normal
12    set weight ← call fabs(weight)
```

	ENDFUNCTION

Kode Sumber 4.4 *Pseudocode* Fungsi *countWeight.*

4.3 Implementasi Android *Gateway*

Pada bagian implementasi ini dibahas mengenai bagaimana implementasi yang dilakukan untuk membangun Android *gateway* berdasarkan apa yang telah dirancang pada bab 3.4

4.3.1 Implementasi Fungsi (Android *Gateway*)

Sesuai dengan perancangan diagram alir 3.4.1 aplikasi Android *gateway* ini akan bertindak sebagai koordinator atau yang mengatur jalannya masing masing pendekatan *Weighted Round Robin* yang sudah di rancang dalam bab 3.4. Android *gateway* akan menjalankan metode WRR (*Weighted Round Robin*) sesuai dengan pilihan dari pengguna aplikasi. Pilihan metode pendekatan tersebut adalah metode pendekatan waktu atau metode pendekatan giliran. Masing-masing metode bekerja didalam *message handler*. Selain itu Android *gateway* nantinya akan menampilkan log hasil dari pengiriman data dari *node* sebagai bentuk monitoring dari *node* tersebut.

Namun dalam Android ada beberapa proses yang penting juga agar aplikasi dari Android *gateway* berjalan sesuai dengan yang diharapkan. Ada proses inisiasi awal (*onCreate)*, proses menyambungkan *node* sensor dengan Android *gateway* (*onResume)*, dan proses untuk membuat koneksi dari masing masing *node* terjaga (*ConnectedThread)*.

4.3.1.1 Fungsi *onCreate*

Pada pemrograman Android, fungsi *onCreate* adalah sebuah fungsi yang berguna untuk inisiasi awal. Fungsi ini akan dijalankan di awal ketika aplikasi Android dijalankan. Untuk implementasinya, fungsi *onCreate* melakukan inisiasi variabel-variabel yang diperlukan, serta menjalankan beberapa fungsi lain seperti fungsi inisialisasi Bluetooth, inisialisasi log, dan tombol tombol penunjang aplikasi. Kode Sumber 4.5 merupakan penjabaran dari fungsi *onCreate* dalam bentuk *pseudocode*.

```
1   @Override
2   function onCreate(Bundle savedInstanceState)
3     super.onCreate(savedInstanceState)
4     initialize all variables
5     set btAdapter ← getDefaultAdapter()
6     call checkBTState()
7     set wrrtype1 ← new Handler()
8     set wrrtype2 ← new Handler()
9
10    set listener btnfirstmethod(onClick
11      @Override
12      function onClick()
13        set flagstop ← 0
14        set methodtype ← 1
15        call sendmessage()
16    )
17
18    set listener btnsecondmethodt(onClick
19      @Override
20      function onClick()
```

```
21      set flagstop ← 0
22      set methodtype ← 2
23      call sendmessage()
24    )
```

Kode Sumber 4.5 *Pseudocode* Fungsi *onCreate.*

4.3.1.2 Fungsi *onResume*

Fungsi ini merupakan fungsi yang akan berjalan apabila layar sebuah *smartphone* mati lalu kemudian menyala kembali. Saat itu terjadi fungsi ini akan bekerja. Selain itu fungsi ini akan dijalankan di awal saat aplikasi baru dibuka yaitu setelah menjalankan fungsi *onCreate*.

Saat fungsi ini berjalan, aktivitas yang terjadi adalah Android *gateway* akan mulai menyambungkan dirinya dengan *node* sensor yang ada (di dalam Tugas Akhir ini *node* sensor berjumlah 3) dengan menggunakan Bluetooth. Koneksi dilakukan pada *node* sensor sesuai dengan alamat Bluetooth yang telah disimpan sebelumnya (*hardware address*). Berikut adalah Kode Sumber 4.6 sebagai *pseudocode* dari fungsi *onResume().*

```
1   @Override
2   function onResume()
3     super.onResume()
4     initialize all variables
5     for f = 0 to device, step =1 do
6       set device ← getRemoteDevice(address[f])
7       set btSocket[f] ←
8   createBluetoothSocket(device[f])
9       call btSocket[f].connect()
.
```

10	**set** mConnectedThread[f] ← newConnectedThread(btSocket[f]) **call** mConnectedThread.start()

Kode Sumber 4.6 *Pseudocode* Fungsi *onResume.*

4.3.1.3 *Handling Message* Metode Pendekatan

Dalam bagian subbab ini akan dijelaskan bentuk implementasi dari masing – masing metode pendekatan. Pada implementasinya masing masing metode pendekatan akan diimplementasikan ke dalam *message Handler*. Ketika Android *gateway* sudah menetapkan jenis metode pendekatannya, maka Android *gateway* akan mengolah *message* yang diterima dari *node* sensor sesuai dengan metode pendekatan yang akan digunakan. Implementasi dari metode pendekatan yang akan di bahas adalah metode pendekatan waktu, metode pendekatan jumlah giliran, dan yang terakhir adalah metode round-robin biasa. Untuk *handling message* ini ada didalam fungsi *onCreate*.

4.3.1.3.1 Implementasi Metode Pendekatan Waktu

Sesuai dengan perancangan yang sudah dijelaskan pada bab 3.4.1.1 pada metode ini ketika jenis metode pendekatan sudah ditetapkan pada Android *gateway* maka akan dilakukan pengiriman *message* ke *node* pertama. Kemudian ketika *node* pertama sudah mengirimkan *message* ke Android *gateway*, maka Android *gateway* akan mengolah data yang diterima, sehingga didapatkan *sensorid*, kemudian bobot dari *node* pengirim yang sudah dihitung didalam *node* itu sendiri, kemudian rata-rata dari data yang sudah didapatkan. Android

gateway akan melakukan pengecekan setiap *node* mengirimkan *message*. Ketika semua *node* sudah mengirimkan message makaakan dilakukan perhitungan bobot masing masing *node*. Ketika perhitungan sudah dilakukan, Android *gateway* akan kembali melakukan pengiriman kembali ke *node* pertama dalam urutan, dan *node* akan melakukan *sensing* sesuai dengan perhitungan jumlah waktu. Berikut ini Kode Sumber 4.7 merupakan implementasi dari metode pendekatan waktu.

```
1   Roundrobintype1 ← call new Handler(){
2    function handleMessage(android.os.Message msg)
3       initialize all variables
4       if msg.what == handlerstate
5         set readMessage ← new
6   StringBuilder().append((String)
.   msg.obj).append("\n").toString()
7         set messageparts[] ← readMessage.split("#")
8         set sensorid ← (int) messageparts[0]
9         append.log(readMessage)
10        if sendflag[sensorid -1] == 1
11          set xvalue[sensorid-1] ← (float)
12  messageparts[3]
13          set flag = 0
14          for i=0 to device , step=1
15            if xvalue[i] != 0
16              flag++
17
18          if flag == device
19            set sumvalue ← 0
20            for t=0 to device, step=1
21              sumvalue+=xvalue[t]
22
```

```
23          for k=0 to device, step=1
.             set percentage ← (double)
24  (xvalue[k]/sumvalue)*100
25            set time[t] ← percentage*90/100
26            set xvalue[t] ← 0
27
28        for i=0 to device, step=1
29          if sendflag[i] == 1
30            if i!=device-1
31              set sendflag[i+1] ←1
32              call
33  mConncetedThread[i+1].write(time[i+1]
34              break
35
36            else
37              set sendflag[0] ←1
38              call mConncetedThread[0].write(time[0]
39              break
40
41        scrollview.post(new Runnable()
42          @Override
.           Public void run()
43              Call
    scrollview.fullScroll(ScrollView.FOCUS_DOWN)
          )
```

Kode Sumber 4.7 *Pseudocode handle message* metode pendekatan waktu.

4.3.1.3.2 Implementasi Metode Pendekatan Jumlah giliran

Seperti perancangan pada bab 3.4.1.2 metode pendekatan ini hamper memiliki alur proses yang sama dengan metode

pendekatan waktu. Namun yang membedakan disini adalah perhitungan dari bobotnya menghasilkan jumlah giliran pada masing masing *node*. *Node* 1,2,3 akan bergantian melakukan *sensing* secara urutan. Ketika asumsinya bahwa *node* ke 2 sudah melakukan jumlah giliran sesuai dengan perhitungan Android *gateway* maka di putaran selanjutnya hanya *node* 1, dan 3 yang melakukan *sensing*. Ketika semua *node* sudah melakukan *sensing* sejumlah yang ditentukan Android *gateway* maka Android *gateway* akan kembali melakukan perhitungan jumlah giliran masing masing *node* berdasarkan bobot terakhir yang ditentukan.

```
1   Roundrobintype1 ← call new Handler(){
2    function handleMessage(android.os.Message msg)
3       initialize all variable
4       if msg.what == handlerstate
5         set readMessage ← new
6   StringBuilder().append((String)
.   msg.obj).append("\n").toString()
7         set messageparts[] ← readMessage.split("#")
8         set sensorid ← (int) messageparts[0]
9         append.log(readMessage)
10        if countcompare[sensorid-1] !=
11  counterlap[sensorid-1]
12          countcompare[sensorid-1]++
13        set xvalue[sensorid-1] ← (float)
14  messageparts[3]
15        set flag = 0
16
17          for i=0 to device , step=1
18            if  countcompare[i] == counterlap[i]
19              flag++
```

```
20
21          if flag == device
22            set sumvalue ← 0
23            set eq ← 0
 .            for t=0 to device, step=1
24             sumvalue+=xvalue[t]
25
26            for k=0 to device, step=1
 .             set percentage ← (double)
27  (xvalue[k]/sumvalue)*100
28             set ctrlap ← (float) percentage*9/100
29             set ctrcmpr ← (int) ctrlap
30             if ctrlap – ctrcmpr>0.5
31               set eq ← Math.round(ctrlap)
32             else
33               set eq  ← Math.floor(ctrlap)
34             set counterlap[t] ← (int) eq
35             set xvalue[t] ← 0
36             set countcompare[t] ← 0
37             set sendflag[t] ← 0
38
39            set sendflag[0] ← 1
40            call mConncetedThread[0].write(time[0]
41
42
43          else if sendflag[sensorid-1]== 1
44            set sendflag[sensorid-1] ← 0
45            set i ← sensorid-1
46            if i == device-1
47              set i ← 0
48            else
49              i++
```

```
50          while countcompare[i] == counterlap[i]
51            if i == device-1
52              set i ← 0
53            else
54              i++
55          call mConncetedThread[i].write(time[i])
56          set sendflag[i] ← 1
57
58        scrollview.post(new Runnable()
59          @Override
60          Public void run()
61            Call
62  scrollview.fullScroll(ScrollView.FOCUS_DOWN)
63        )
```

Kode Sumber 4.8 *Pseudocode* handle message implementasi metode pendekatan jumlah giliran.

4.3.1.3.3 Implementasi Metode *Round-Robin*

Metode *Round-Robin* akan diimplementasikan juga didalam Android *gateway*, yang nantinya akan digunakan sebagai pembanding untuk uji coba antara 2 metode pendekatan yang sudah diimplementasikan dengan metode *Round Robin* biasa. Untuk *Round Robin* alurnya hampir sama dengan metode pendekatan waktu, namun yang membedakan disini adalah bobot dan proses perhitungan waktu tidak akan di gunakan sehingga waktu untuk masing masing *node* melakukan *sensing* sama dengan jumlah total waktu setiap *node* metode pendekatan waktu. Kode Sumber 4.9 merupakan *pseudocode* dari metode *Round Robin*.

```
1   Roundrobin ← call new Handler(){
2    function handleMessage(android.os.Message msg)
3       initialize all variables
4       if msg.what == handlerstate
5         set readMessage  ← new
6   StringBuilder().append((String)
.   msg.obj).append("\n").toString()
7         set messageparts[]  ← readMessage.split("#")
8         set sensorid  ← (int)  messageparts[0]
9         append.log(readMessage)
10        if sendflag[sensorid -1] == 1
11          set xvalue[sensorid-1] ← (float)
12  messageparts[3]
13          set flag = 0
14          for i=0 to device , step=1
15            if xvalue[i] != 0
16              flag++
17
18         for i=0 to device, step=1
19           if sendflag[i] == 1
20             if i!=device-1
21               set sendflag[i+1] ←1
22               call
23  mConncetedThread[i+1].write(time[i+1]
24               break
25
26             else
27               set sendflag[0] ←1
28               call mConncetedThread[0].write(time[0]
29               break
30
31        scrollview.post(new Runnable()
```

```
32          @Override
 .          Public void run()
33             Call
34  scrollview.fullScroll(ScrollView.FOCUS_DOWN)
          )
```

Kode Sumber 4.9 *Pseudocode handle message* metode *Round Robin* biasa.

4.3.1.4 Kelas *ConnectedThread*

Pada kelas *ConnectedThread* berfungsi sebagai pengatur masuk dan keluarnya data didalam Android *gateway*. Setelah socket dari Bluetooth sudah dibuat kelas ini akan terus melakukan looping untuk menunggu adanya kiriman data dari *node* sensor. Kemudian di kelas ini juga Android *gateway* melakukan proses *handlilng* data sesuai dengan metode pendekatan yang ditentukan.

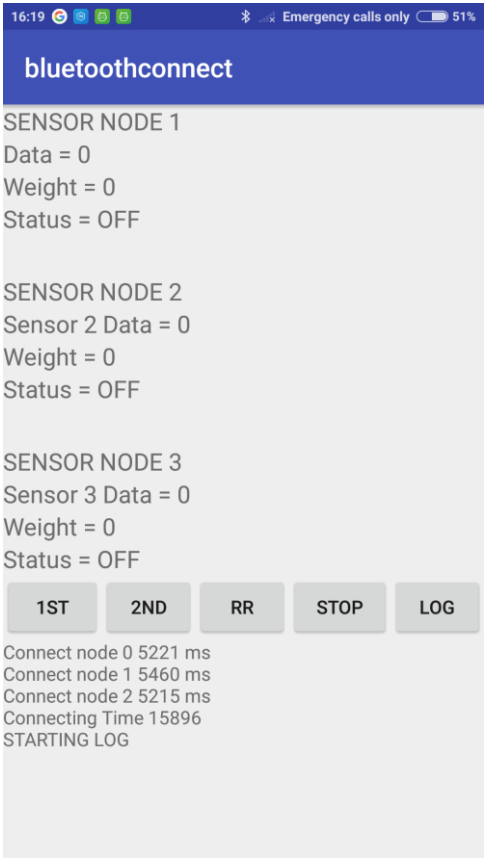
```
1   Private class ConnectedThread extends Thread
2    Initialize all variables
3
4    Public ConnectedThread(BluetoothSocket socket)
5     set tmpIn ←null
6     set tmpOut ←null
7
8     try
9       set tmpIn ← socket.getInputStream()
10      set tmpOut ← socket.getOutputStream()
```

```
11      catch IOException e
12
13      set mmInStream ←tmpIn
14      set  mmOutStream ←tmpOut
15
16    Public void run()
17      set buffer ← (byte[]) new byte[256]
18      set bytes ← 0
19      while true
20        try
21          set buffer ← (byte) mmInStream.read()
22        catch IOException e
23
24
25       if buffer[bytes] == '\n' or buffer[bytes]=='\r'
26         set readMessage ← new String(buffer,0,bytes)
27         if methodtype == 1
28
 .    roundrobintype1.obtainMessage(handlerState,bytes,-
29   1,readMessage).sendToTarget()
30         else if methodtype ==2
 .
31   roundrobintype2.obtainMessage(handlerState,bytes,-
32   1,readMessage).sendToTarget()
 .         else if methodtype ==0
33           roundrobin.obtainMessage(handlerState,bytes,-
34   1,readMessage).sendToTarget()
35
36       else bytes++
37
38    public void write(int input)
        try mmOutStream.write(input)
```

	catch IOException e

Kode Sumber 4.10 *Pseudocode* kelas *ConnectedThread.*

4.3.2 Implementasi Antarmuka (Android *Gateway*)



Gambar 4.2 Implementasi Antarmuka Pengguna (Android *Gateway*).

Pada bagian ini, akan dijelaskan mengenai implementasi antarmuka pengguna dari aplikasi Android *gateway*. Untuk antar muka dari aplikasi Android *gateway* terdiri dari 3 baris list kondisi dari masing masing *node*, kemudian ada 5 tombol penting , yaitu tombol *Sensing* dengan metode WRR pendekatan jumlah waktu, tombol *Sensing* dengan metode WRR pendekatan jumlah giliran, tombol *Sensing* dengan metode *Round Robin*, tombol Stop dan tombol mencatat *log*. Dibawah dari tombol tombol tersebut ada kotak l*og* untuk melihat *log* dari komunikasi dari system.

(Halaman ini sengaja dikosongkan)

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dilakukan tahap uji coba dan evaluasi sesuai dengan rancangan dan implementasi tugas akhir. Dari hasil yang didapatkan setelah melakukan uji coba, akan dilakukan evaluasi sehingga dapat ditarik kesimpulan pada bab selanjutnya.

5.1 Lingkungan Uji Coba

Lingkungan uji coba mencakup perangkat dan peralatan yang nantinya akan digunakan dalam uji coba. Uji coba akan dilakukan dengan spesifikasi seperti tabel dibawah ini. Untuk perangkat mikrokontroler dan sensor yang sudah dirangkai sebanyak 3 buah, kemudian untuk perangkat *smartphone* berjumlah 1 buah.

Tabel 5.1 Spesifikasi Perangkat Uji coba

Perangkat	Detail Perangkat
Perangkat Mikrokontroler	**Mikrokontroler :** • Atmega328 **Model :** • Arduino UNO *Revision* 3 **Tegangan :** • 5 V **Memori *Flash* :** • 32 KB **SRAM :** • 2 KB **Sensor :** • Modul MQ-2 (Asap)

Perangkat	Detail Perangkat
	***Transceiver* :** • Model HC-06 (Bluetooth)
Perangkat Mobile (*Smartphone*)	**Model :** • Xiaomi Redmi Note 3 Pro **Manufaktur :** • Xiaomi **Processor :** • Qualcomm MSM8956 Snapdragon 650 Quad-core 1.4 GHz Cortex-A53 **Memori :** • 2 GB **Berat :** • 164 gram **Dimensi :** • 150 x 76 x 8.7 mm **Lebar Layar :** • 5.5” **Resolusi :** • FHD (1080 x 1920) **Bluetooth :** • 4.0 + EDR + A2DP **Baterai :** • 4000 mAh **Sistem Operasi :** • MIUI 7.0 (based on Android 5.1.1 Lollipop)

5.2 Skenario Uji Coba Fungsionalitas

Uji Coba Fungsionalitas akan dilakukan untuk menguji fungsionalitas dari sistem yang sudah dibuat. Beberapa yang diujikan disini adalah uji coba konektivitas Android *gateway* ke semua *node* sensor, dan uji coba mendapatkan nilai sensor dari pengiriman data ke Android *gateway*

5.2.1 Skenario Uji Coba (UJ-F1) – Uji Konektivitas *Node* Sensor dengan Android *Gateway*

Dalam uji coba fungsionalitas ini akan diuji bagaimana konektifitas *node* terhadap Android *Gateway* ketika aplikasi dalam Android *gateway* dijalankan. Hasil yang diharapkan adalah berupa *log* yang sudah di program didalam Android *Gateway* yang nantinya akan menunjukan bahwa koneksi telah berhasil dilakukan.

Tabel 5.2 Skenario Uji Konektivitas *Node* Sensor dengan Android *Gateway*.

ID	**UJ-F1**
Nama	Uji Coba Konektivitas*Node* Sensor dengan Android *Gateway*
Tujuan Uji Coba	Menguji fungsionalitas dari Android *gateway* dalam membuat koneksi dengan *node* sensor.
Kondisi Awal	Menyalakan *node* sensor dan menjalankan aplikasi di Android *Gateway*
Skenario	1. Menyalakan semua *node* sensor 2. Melakukan pemantauan Keberhasilan koneksi 3. Melakukan pengamatan log di Aplikasi Android Studio
Masukan	Data Sensor
Keluaran	Tampilan Log dalam aplikasi Android Studio
Hasil yang Diharapkan	Android *Gateway* dapat terkoneksi dengan semua *node* sensor.

5.2.2 Skenario Uji Coba (UJ-F2) - Mendapatkan Nilai dari *Node* Sensor dan Mengirimkan Data ke Android *Gateway*

Fungsionalitas selanjutnya yang akan diuji adalah fungsionalitas dari *node* sensor yang akan mengirimkan data ke Android *gateway* melalui koneksi Bluetooth. Kemudian nantinya akan dilihat hasilnya melalui *Serial monitor*

Tabel 5.3 Skenario Uji Coba Mendapatkan Nilai Sensor dan Mengirimkan Data ke Android *Gateway.*

ID	**UJ-F2**
Nama	Uji Coba Mendapatkan Nilai dari *Node* Snesor dan Mengirimkan Data ke Android *Gateway*
Tujuan Uji Coba	Menguji Fungsionalitas dari *node* sensor yang nantinya akan melakukan *sensing* sesuai dengan metode yang sudah di implementasikan.
Kondisi Awal	Menyalakan *node* sensor dan Android *gateway*
Skenario	1. Menyalakan semua *node* sensor 2. Menjalankan aplikasi di Android *Gateway* 3. Melakukan pemantauan nilai yang didapatkan dari *node* sensor di Serial Monitor Arduino 4. Melakukan pemantauan nilai yang disudah dikirimkan oleh *node* sensor di Android *Gateway*
Masukan	Nilai Sensor Asap
Keluaran	Tampilan nilai pada *serial monitor* dan data yang diterima Android *gateway*
Hasil yang Diharapkan	*Node* sensor bisa mendapatkan nilai sensor asap dan kemudian akan dilakukan pengiriman ke Android *Gateway*

5.3 Hasil Uji Coba Fungsionalitas

Pada bagian sebelumnya telah dibahas skenario uji coba dari hasil implementasi tugas akhir ini. Pada bagian ini akan

dibahas hasil uji coba dari masing-masing skenario uji coba. Berikut adalah hasil dari uji coba sesuai dengan skenario.

5.3.1 Hasil Uji Coba (UJ-F1) - Uji Konektivitas *Node* Sensor dengan Android *Gateway*

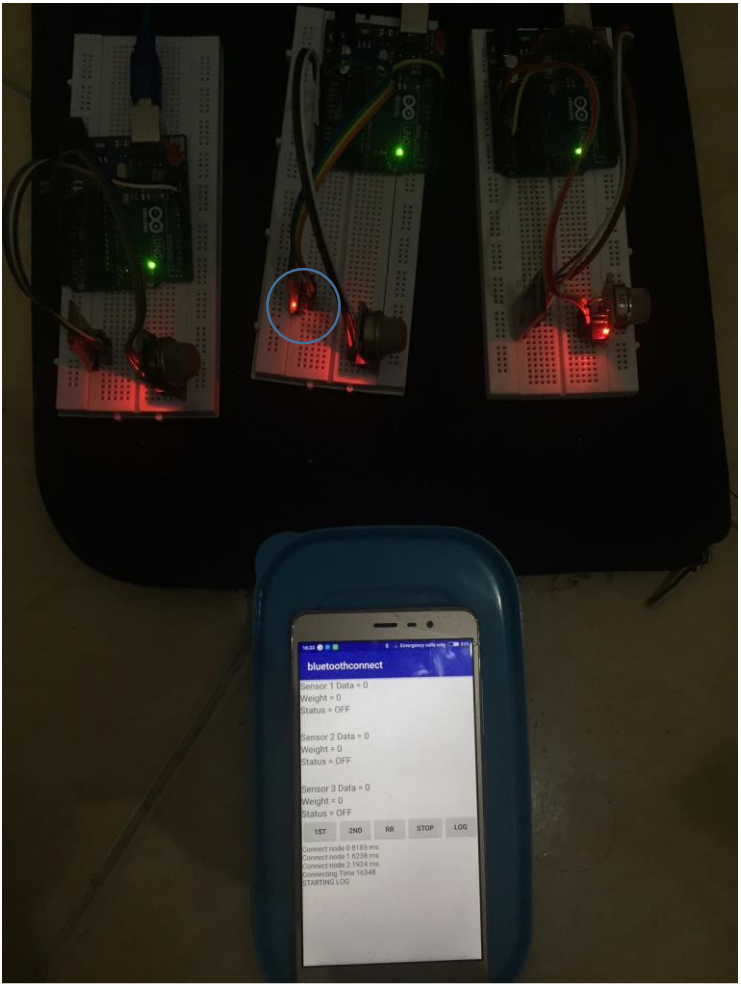
Sesuai dengan Skenario Uji coba pada bab 5.2.1 akan dilakukan pengujian dengan mengkoneksikan 3 *node* sekaligus dengan Android *gateway* dimana nantinya penggunaan metode dapat digunakan dengan baik. Pengujian dimulai dengan menyalakan semua *node* sensor, kemudian menjalankan aplikasi Android *gateway*. Kemudian akan dipantau *log* dari Android studio dan juga nyala dari lampu modul Bluetooth HC. Pada Gambar 5.1 dapat dilihat bahwa pembuatan socket Bluetooth berhasil dan kemudian koneksi dari Android *Gateway* dengan masing masing *node* sensor berhasil.

```
$ adb shell am start -n "com.example.rayha.bluetoothconnect/com.example.rayha.bluetoothconnect.discover"
Client not ready yet..Waiting for process to come online
Connected to process 16796 on device 33be4242
Capturing and displaying logcat messages from application. This behavior can be disabled in the "Logcat
D/Socket Created: ....Connection Socket 0 OK....
D/Sukses:  CreateSocket1
W/BluetoothAdapter: getBluetoothService() called with no BluetoothManagerCallback
D/Socket Created: ....Connection Socket 1 OK....
D/Sukses:  CreateSocket1
W/BluetoothAdapter: getBluetoothService() called with no BluetoothManagerCallback
D/Socket Created: ....Connection Socket 2 OK....
I/ViewRootImpl: CPU Rendering VSync enable = true
D/OpenGLRenderer: Use EGL_SWAP_BEHAVIOR_PRESERVED: true
D/Atlas: Validating map...
D/ActivityThreadInjector: clearCachedDrawables.
```

Gambar 5.1 Hasil dari log Android Studio saat mengkoneksikan aplikasi di Android *Gateway* dengan 3 *Node* sensor.

Kemudian pada Gambar 5.2, Gambar 5.3, dan Gambar 5.4 dapat dilihat bahwa nyala lampu dari masing maisng *node* sensor tidak berkedip cepat menandakan bahwa *node* sensor sudah terkoneksi dengan Android *gateway*

Gambar 5.2 Nyala lampu pada *node* 1 menunjukan bahwa *node*1 sudah terkoneksi dengan Android *gateway*.



Gambar 5.3 Nyala lampu pada *node* 2 menunjukan bahwa *node* 2 sudah terkoneksi dengan Android *gateway*.

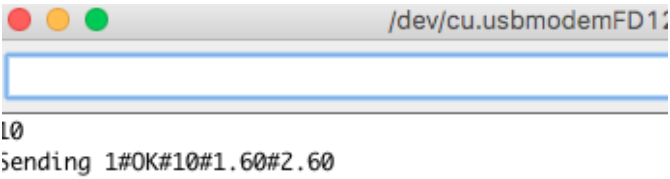
Gambar 5.4 Nyala lampu dari *node* 3 menunjukan bahwa *node* 3 sudah terkoneksi dengan Android *gateway*.

Nyala lampu yang terjadi adalah kedipan yang dari lampu modul HC adalah kedipan yang lambat jika sudah terkoneksi dengan Android *Gateway*. Dan jika kedipannya cepat maka *node* sensor belum terkoneksi dengan Android *gateway*.

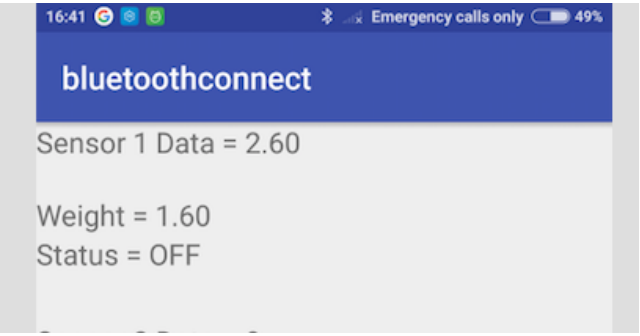
5.3.2 Hasil Uji Coba (UJ-F2) - Mendapatkan Nilai dari *Node* Sensor dan Mengirimkan Data ke Android *Gateway*

Pengujian akan dilakukan sesuai dengan scenario ujicoba pada bab 5.2.2. pertama semua *node* sensor akan dinyalakan dan aplikasi Android *Gateway* dijalankan. Kemudian Android *Gateway* akan mengirimkan perintah dan menunggu masing masing *node* sensor mengirimkan data ke Android *Gateway*. Pemantauan akan dilakukan di serial monitor Arduino IDE dan juga tampilan di aplikasi Android *Gateway* . Jika data yang ditampilakan pada serial monitor sama dengan apa yang ditampilkan pada aplikasi Android *gateway* maka pengiriman data dari *node* sensor ke Android *gateway* berhasil. Untuk format data yang dikirim dari *node* sensor adalah id *node*#OK#waktu *sensing*#bobot*node*#Data *sensing*.

Pada masing masing *node* akan dilakukan uji coba yang sama. Gambar 5.5 dan Gambar 5.6 menunjukan hasil dari uji coba yang dilakukan pada *node* 1.

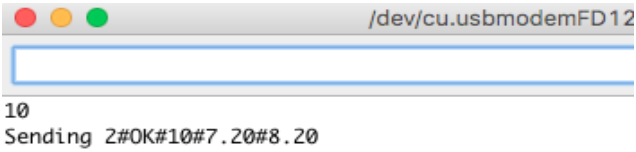


Gambar 5.5 Serial monitor pada *Node* sensor 1.

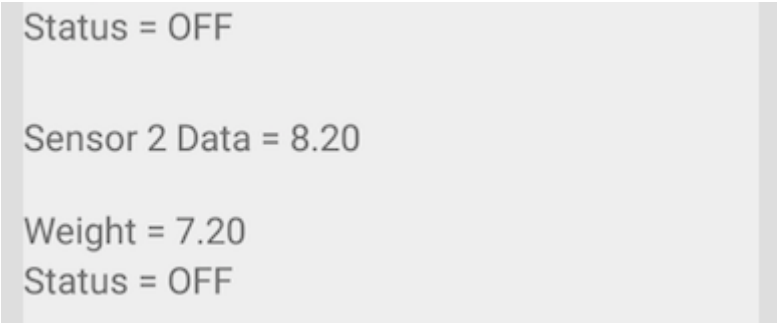


Gambar 5.6 Hasil Uji coba pengiriman data dari *node* sensor 1.

Uji coba dilanjutkan mencoba fungsionalitas dari *node* 2 dengan perlakuan dan scenario yang sama dengan *node* 1 yaitu, Jika data yang dikirimkan dari *node* 2 sama dengan data yang ditampilkan pada aplikasi Android *Gateway* maka uji coba dari *node* 2 berhasil. Gambar 5.7 dan Gambar 5.8 menunjukan hasil uji coba yang dilakukan kepada *node* 2.

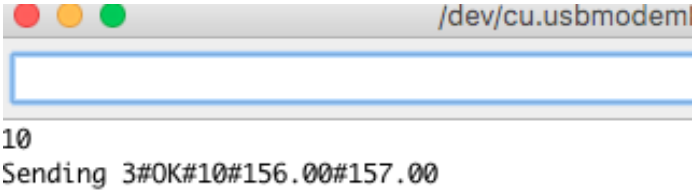


Gambar 5.7 Serial monitor pada *node* sensor 2.

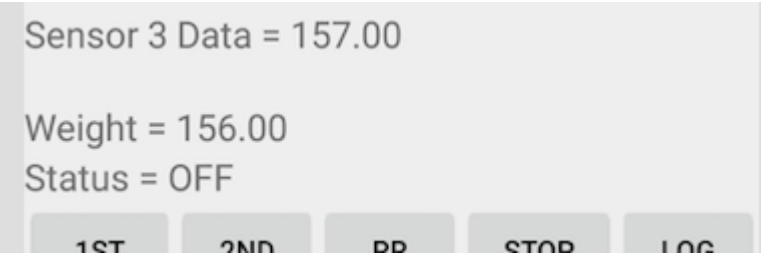


Gambar 5.8 Hasil Uji coba pengiriman data dari *node* sensor 2.

Kemudian *node* 3 juga dilakukan uji coba dengan scenario yang sama seperti *node* 1, dan 2. Pada Gambar 5.9 dan Gambar 5.10 menunjukan hasil uji coba yang dilakukan terhadap *node* sensor 3.



Gambar 5.9 Serial monitor pada *node* sensor 3.



Gambar 5.10 Hasil Uji Coba Pengiriman data dari *node* sensor 3**.**

5.4 Skenario Uji Coba Performa

Uji coba performa akan dilakukan untuk menguji performa kerja dari beberapa metode yang sudah diimplementasikan. Beberapa hal masalah performa yang akan diujikan ke masing masing metode adalah *Delay* pengiriman data dari msaing masing *node* dan *Packet Delivery Ratio* dari setiap Metode pendekatan. Berikut di bawah ini merupakan scenario pengujian yang dilakukan.

5.4.1 Skenario Uji Coba (U-P1) – *Delay* Pengiriman Data dari *Node* Sensor ke Android *Gateway*

Uji coba ini dilakukan untuk mendapatkan *delay* rata rata komunikasi antar *node* sensor dan Android dengan menggunakan metode yang sudah di implementasikan. Metode yang dibandingkan adalah metode WRR (Pendekatan waktu dan pendekatan jumlah giliran), dan metode *Round Robin* biasa.

Tabel 5.4 Skenario Uji Coba *Delay* Pengiriman data dari *node* sensor ke Android *Gateway.*

ID	**UJ-P01**
Nama	Uji Coba *Delay* Pengiriman Data dari *Node* Sensor ke Android *Gateway*

Tujuan Uji Coba	Menguji performa yang berupa *delay* rata-rata yang dibutuhkan dalam proses pengiriman data dari masing masing *node* ke Android *gateway* pada masing masing metode pendekatan melalui Bluetooth
Kondisi Awal	Mengirimkan data dari android ke *node* sensor , kemudian *node* sensor akan mengirimkan data ke Android *gateway*
Skenario	1. Menjalankan *node* sensor dengan mengirimkan perintah dari Android *Gateway* menggunakan WRR pendekatan waktu 2. Menunggu jalannya system sampai 2 menit 3. Memberhentikan pengiriman ketika sudah mencapai 2 menit 4. Melihat hasil yang didapat berupa log di Android *gateway* yang nantinya akan dihitung dengan mengurangi waktu perhitungan slot waktu masing masing *node* dengan waktu yang di dapat dari hasil uji coba 5. Menghitung *delay* waktu pengiriman rata-rata pada metode WRR Pendekatan waktu 6. Menjalankan *node* sensor dengan mengirimkan perintah dari Android *Gateway* menggunakan WRR pendekatan jumlah giliran 7. Menunggu jalannya system sampai 2 menit 8. Memberhentikan pengiriman ketika sudah mencapai 2 menit 9. Melihat hasil yang didapat berupa log di Android *gateway* yang nantinya akan dihitung dengan mengurangi waktu perhitungan slot waktu masing masing *node* dengan waktu yang di dapat dari hasil uji coba 10. Menghitung *delay* waktu pengiriman rata-rata pada metode WRR Pendekatan jumlah giliran 11. Menjalankan *node* sensor dengan mengirimkan perintah dari Android *Gateway* menggunakan *Round Robin* biasa

	12. Menunggu jalannya system sampai 2 menit 13. Memberhentikan pengiriman ketika sudah mencapai 2 menit 14. Melihat hasil yang didapat berupa log di Android *gateway* yang nantinya akan dihitung dengan mengurangi waktu perhitungan slot waktu masing masing *node* dengan waktu yang di dapat dari hasil uji coba 15. Menghitung *delay* waktu pengiriman rata-rata pada metode *Round robin* biasa
Masukan	Data *node* sensor
Keluaran	Waktu dalam detik
Hasil yang Diharapkan	Mendapatkan perbandingan *delay* waktu rata – rata pada masing masing metode pendekatan Weighted *Round Robin* pada *node* sensor.

5.4.2 Skenario Uji Coba (U-P2) – *Packet Delivery Ratio* Dari Masing-Masing Metode Pendekatan

Pengujian ini dilakukan untuk menghitung *Packet Delivery Ratio* dari masing masing metode yang diimplementasikan. Uji coba akan dilakukan dengan menjalankan masing masing metode dengan ketentuan jarak antara *node* sensor dan Android *gateway* 3m, 5m. Dan 7m. Pengujian akan dilakukan selama 2 menit dan 5 kali percobaan Tabel 5.5 merupakan skenario uji coba secara lebih rinci.

Tabel 5.5 Skenario Uji Coba *Packet Delivery Rasio* dari masing masing metode pendekatan.

ID	**UJ-P02**
Nama	Uji Coba *Packet Delivery Rasio* dari masing masing metode pendekatan
Tujuan Uji Coba	Menguji performa yang berupa *Packet Delivery Ratio* dari masing masing metode pada jarak 3m, 5m, dan 7m.

Kondisi Awal	Mengirimkan data dari android ke *node* sensor , kemudian *node* sensor akan mengirimkan data ke Android *gateway*
Skenario	1. Menjalankan *node* sensor dengan mengirimkan perintah dari Android *Gateway* menggunakan WRR pendekatan waktu dengan jarak 3m 2. Menunggu jalannya system sampai 2 menit 3. Memberhentikan pengiriman ketika sudah mencapai 2 menit 4. Melihat hasil dari *log* yang dicatat kemudian melakukan step ke 1-3 selama 5 kali untuk jarak 5 dan 7 meter 5. Menjalankan *node* sensor dengan mengirimkan perintah dari Android *Gateway* menggunakan WRR pendekatan jumlah giliran dengan jarak 3m 6. Menunggu jalannya sistem sampai 2 menit 7. Memberhentikan pengiriman ketika sudah mencapai 2 menit 8. Melihat hasil dari *log* yang dicatat kemudian melakukan step ke 5-7 selama 5 kali untuk jarak 5 dan 7 meter 9. Menjalankan *node* sensor dengan mengirimkan perintah dari Android *Gateway* menggunakan *Round robin* waktu dengan jarak 3m 10. Menunggu jalannya sistem sampai 2 menit 11. Memberhentikan pengiriman ketika sudah mencapai 2 menit 12. Melihat hasil dari *log* yang dicatat kemudian melakukan step ke 1-3 selama 5 kali untuk jarak 5 dan 7 meter
Masukan	Data sensor
Keluaran	Jumlah paket

Hasil yang Diharapkan	Mendapatkan *Packet Delivery Ratio* dari masing masing metode dengan jarak yang berbeda beda.

5.4.3 Skenario Uji Coba (U-P3) – *Delay* Pengiriman Data Kritikal Dari Masing-Masing Metode Pendekatan

Pengujian ini dilakukan untuk mendapatkan *delay* data kritikal dari masing masing metode akan dilakukan scenario ujicoba dengan pada detik ke 45 akan dikirimkan data yang penting dimana akan didapatkan *delay* penerimaan data dari masing masing metode. Untuk detik yang dihitung sudah termaksud dengan detik perhitungan masing masing metode. Akan dilakukan 10 kali percobaan pengiriman data kritikal. Data kritikal yang diambil adalah data yang terakhir dikirim setelah detik ke 45. Tabel 5.6 merupakan skenario uji coba secara lebih rinci.

Tabel 5.6 Skenario Uji Coba *Delay* Pengiriman Data Kritikal Dari Masing-Masing Metode Pendekatan.

ID	**UJ-P03**
Nama	Uji Coba *Delay* Pengiriman Data Kritikal dari Masing- Masing Metode Pendekatan
Tujuan Uji Coba	Menguji performa yang berupa *Delay* data kritikal yang dikirimkan dari masing masing metode pendekatan.
Kondisi Awal	Mengirimkan data dari android ke *node* sensor , kemudian *node* sensor akan mengirimkan data ke Android *gateway*
Skenario	1. Menjalankan *node* sensor dengan mengirimkan perintah dari Android *Gateway* menggunakan WRR Waktu 2. Menunggu jalannya system sampai data kritikal didapatkan, kemudian melihat hasil log berupa *delay* pengiriman data

	3. Mengulangi proses 1 dan 2 sebanyak 10 kali untuk dicatat hasil nya 4. Menjalankan *node* sensor dengan mengirimkan perintah dari Android *Gateway* menggunakan WRR giliran 5. Menunggu jalannya system sampai data kritikal didapatkan, kemudian melihat hasil log berupa *delay* pengiriman data 6. Mengulangi proses 4 dan 5 sebanyak 10 kali untuk dicatat hasil nya 7. Menjalankan *node* sensor dengan mengirimkan perintah dari Android *Gateway* menggunakan *Round Robin* 8. Menunggu jalannya system sampai data kritikal didapatkan, kemudian melihat hasil log berupa *delay* pengiriman data 9. Mengulangi proses 7 dan 8 sebanyak 10 kali untuk dicatat hasil nya
Masukan	Data sensor
Keluaran	*Delay* pengiriman data
Hasil yang Diharapkan	Mendapatkan Delay pengiriman data pada situasi kritikal pada masing masing metode..

5.5 Hasil Uji Coba Performa

Pada bagian sebelumnya telah dibahas skenario uji coba dari hasil implementasi tugas akhir ini. Pada bagian ini akan dibahas hasil uji coba dari masing-masing skenario uji coba. Berikut adalah hasil dari uji coba sesuai dengan skenario.

5.5.1 Hasil Uji Coba (U-P1) - Delay Pengiriman data dari *node* sensor ke Android *Gateway*

Mengacu pada skenario U-P1 pada bab 5.4.1 telah dilakukan pengujian sesuai dengan skenario. Pengujian dilakukan selama 2 kali masing masing selama 2 menit. Hasil uji didapatkan dengan melihat log waktu dari setiap paket yang diterima Android *gateway* pada masing-masing metode. Berikut adalah tabel hasil uji coba percobaan pertama dan percobaan kedua.

Tabel 5.7 Hasil uji coba *Delay* Pengiriman data dari *node* sensor ke Android *Gateway* percobaan pertama.

Percobaan ke1	Rata-rata *delay* (ms)		
Node	Weighted Round-Robin		Round-Robin
	Waktu	Giliran	
Node 1	309.75	61.10810811	419.0833333
Node 2	380.3333333	67.48717949	473.3333333
Node 3	515.0833333	70.10810811	618.5
Rata-Rata *Delay*	401.7222222	66.23446523	503.6388889

Tabel 5.8 Hasil uji coba *Delay* Pengiriman data dari *node* sensor ke Android *Gateway* percobaan kedua.

Percobaan ke2	Rata-rata *delay* (ms)		
	Weighted Round-Robin		Round-Robin
Node	Waktu	Giliran	
Node 1	421.9166667	132	377.8333333
Node 2	266.9166667	84.23684211	451.9166667
Node 3	306.75	82.21428571	533.0833333
Rata-Rata *Delay*	331.8611111	99.48370927	454.2777778

Berdasarkan percobaan pertama didapatkan bahwa rata-rata *delay* pengiriman data dari Metode WRR Pendekatan waktu sebesar 401.72 ms, sementara rata-rata *delay* pengiriman data Metode WRR Pendekatan giliran sebesar 66.23 ms, dan untuk metode *Round Robin* biasa memiliki *delay* 503.63.

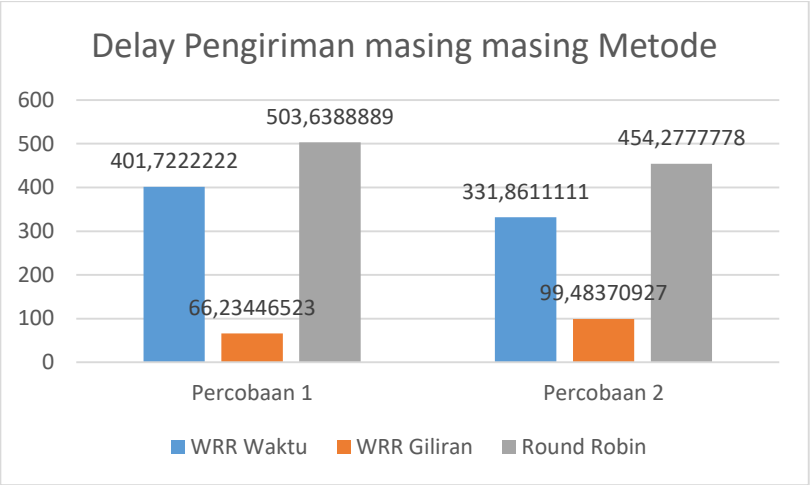
Untuk percobaan kedua didapatkan hasil yang kurang lebih sama untuk masing masing metode. Metode WRR pendekatan waktu memiliki *delay* sebesar 331.86 ms. Kemudian untuk metode WRR pendekatan giliran memiliki *delay* sebesar 99.48 ms. Dan untuk metode *Round Robin* biasa memiliki *delay* 454.27 ms.

Berdasarkan dua percobaan diatas bisa dihitung rata rata *delay* dari masing-masing metode.

Tabel 5.9 Rata rata Hasil uji coba *Delay* Pengiriman data dari *node* sensor ke Android *Gateway* dari dua percobaan.

Node	Weighted Round-Robin		Round-Robin
	Waktu	Giliran	
Percobaan 1	401.7222222	66.23446523	503.6388889
Percobaan 2	331.8611111	99.48370927	454.2777778
Rata-Rata *Delay*	331.8611111	82.85908725	478.9583334

Dari tabel tersebut didapatkan rata-rata dari 2 eksperimen didapatkan rata-rata *delay* metode WRR Pendekatan waktu sebesar 331.86 ms, untuk *delay* dari metode WRR pendekatan giliran sebesar 82.85 ms, dan yang terakhir untuk *Round Robin* memiliki *delay* sebesar 478.95 ms. Jika dibandingan antar metode Metode WRR dengan pendekatan jumlah giliran memiliki *delay* pengiriman terkecil dibandingkan metode yang lain. Kemudian Metode *Round Robin* biasa memiliki *delay* pengiriman lebih besar dibandingkan 2 metode WRR. Gambar 5.11 merupakan grafik perbandingan antara 3 metode berdasarkan *delay* pengiriman data dan dari 2 percobaan



Gambar 5.11 Grafik perbandingan masing masing metode.

5.5.2 Hasil Uji Coba (U-P2) - *Packet Delivery Ratio* Dari Masing-Masing Metode Pendekatan

Mengacu pada skenario U-P2 pada bab 5.4.2 telah dilakukan pengujian sesuai dengan skenario. Pengujian dilakukan selama 5 kali pada jarak 3m ,5m, dan 7m masing masing selama 2 menit. Hasil uji didapatkan dengan melihat log dan menghitung jumlah packet yang diterima beserta packet yang tidak diterima. Berikut adalah tabel hasil uji coba metode WRR dengan pendekatan waktu pada masing masing jarak.

Tabel 5.10 Hasil Uji coba *Packet Delivery Ratio* Metode WRR pendekatan waktu dengan jarak 3m.

WRR Waktu Jarak 3m		Percobaan ke					Akurasi
		1	2	3	4	5	
	Sesuai	38	37	38	36	37	99%

Jumlah data masuk	Total	40	37	38	36	37	

Tabel 5.11 Hasil Uji coba *Packet Delivery Ratio* Metode WRR pendekatan waktu dengan jarak 5m.

WRR Waktu Jarak 5m		Percobaan ke					Akurasi
		1	2	3	4	5	
Jumlah data masuk	Sesuai	37	35	36	34	36	98.87%
	Total	37	36	36	35	36	

Tabel 5.12 Hasil Uji coba *Packet Delivery Ratio* Metode WRR pendekatan waktu dengan jarak 7m.

WRR Waktu Jarak 7m		Percobaan ke					Akurasi
		1	2	3	4	5	
Jumlah data masuk	Sesuai	30	32	34	32	24	95.95%
	tidak	31	34	34	32	27	

Untuk metode WRR Waktu memiliki *Packet Delivery Ratio*(PDR) sebesar 99% untuk jarak kurang lebih 3m. kemudian untuk jarak 5m metode ini memiliki akurasi sebesar 98.87%. sementara untuk jarak 7m WRR waktu memiliki akurasi sebesar 95.95%.

Untuk metode WRR jumlah giliran , berikut adalah table hasil uji coba metode WRR dengan pendekatan giliran:

Tabel 5.13 Hasil Uji coba *Packet Delivery Ratio* Metode WRR pendekatan giliran dengan jarak 3m.

WRR Giliran Jarak 3m		Percobaan ke					Akurasi
		1	2	3	4	5	
Jumlah data masuk	Sesuai	111	112	110	111	101	100%
	Total	111	112	110	111	101	

Tabel 5.14 Hasil Uji coba *Packet Delivery Ratio* Metode WRR pendekatan giliran dengan jarak 5m.

WRR Giliran Jarak 5m		Percobaan ke					Akurasi
		1	2	3	4	5	
Jumlah data masuk	Sesuai	110	110	110	109	108	99.64%
	Total	110	110	111	109	109	

Tabel 5.15 Hasil Uji coba *Packet Delivery Ratio* Metode WRR pendekatan giliran dengan jarak 7m.

WRR Giliran Jarak 7m		Percobaan ke					Akurasi
		1	2	3	4	5	
Jumlah data masuk	Sesuai	104	104	103	88	77	99.108%
	tidak	105	104	103	90	78	

Untuk metode WRR giliran lebih banyak mengirimkan paket dalam 2 menit dikarenakan estimasinya bukan perwaktu namun per giliran. Untuk metode WRR giliran memiliki akurasi pengiriman data sebesar 100% untuk jarak 3m. kemudian untuk jarak 5m metode ini memiliki akurasi sebesar 99.64%. sementara untuk jarak 7m WRR waktu memiliki akurasi sebesar 99.108%

Untuk metode *Round Robin* biasa , berikut adalah table hasil uji coba metode *Round Robin* biasa:

Tabel 5.16 Hasil Uji coba *Packet Delivery Ratio* Metode *Round Robin* dengan jarak 3m.

Round Robin Jarak 3m		Percobaan ke					Akurasi
		1	2	3	4	5	
Jumlah data masuk	Sesuai	35	32	33	34	34	97.15%
	Total	35	34	36	34	34	

Tabel 5.17 Hasil Uji coba *Packet Delivery Ratio* Metode *Round Robin* dengan jarak 5m

Round Robin Jarak 5m		Percobaan ke					Akurasi
		1	2	3	4	5	
Jumlah data masuk	Sesuai	36	33	36	36	35	98.85%
	Total	36	35	36	36	35	

Tabel 5.18 Hasil Uji coba *Packet Delivery Ratio* Metode *Round Robin* dengan jarak 7m.

Round Robin Jarak 7m		Percobaan ke					Akurasi
		1	2	3	4	5	
Jumlah data masuk	Sesuai	35	33	31	35	34	98.24%
	tidak	35	33	32	36	35	

Untuk metode *Round Robin* biasa memiliki akurasi pengiriman data sebesar 97.15% untuk jarak 3m. kemudian untuk jarak 5m metode ini memiliki akurasi sebesar 98.85%.

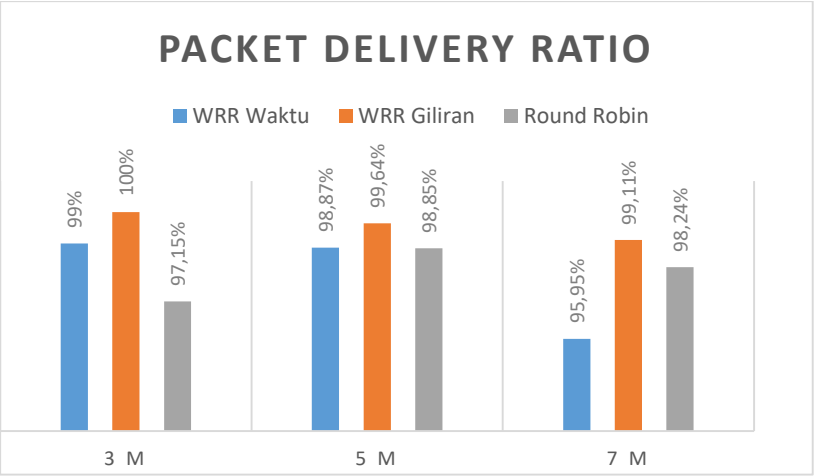
sementara untuk jarak 7m metode ini memiliki akurasi sebesar 98.24%.

Berdasarkan dari hasil semua uji coba packet delivery dapat dirangkum dengan Tabel 5.19 dimana semakin jauh jarak yang digunakan untuk masing masing metode semakin berkurang PDR nya kemudian bisa dilihat juga dari table bahwa metode WRR dengan pendekatan giliran memiliki PDR yang lebih besar dibandingkan metode metode lainnya.

Tabel 5.19 Hasil Uji Coba PDR dari masing masing metode.

Akurasi	*Packet Delivery Ratio* (%)		
Jarak	Weighted Round-Robin		Round-Robin
	Waktu	Giliran	
3 m	99%	100%	97.15%
5 m	98.87%	99.64%	98.85%
7 m	95.95%	99.108%	98.24%

Kemudian dari tabel diatas bisa dibuat grafik PDR sesuai dengan Gambar 5.12 dimana WRR memiliki grafik yang menurun jika jarak pengiriman data ditambahkan, namun untuk PDR dari *Round Robin* tidak mengalami penurunan namun naik ketika jarak 5m dan kembali menurun ketika jarak 7m.



Gambar 5.12 Grafik perbandingan PDR dari masing masing metode.

5.5.1 Hasil Uji Coba (U-P3) - *Delay* Pengiriman Data Kritikal Dari Masing-Masing Metode Pendekatan

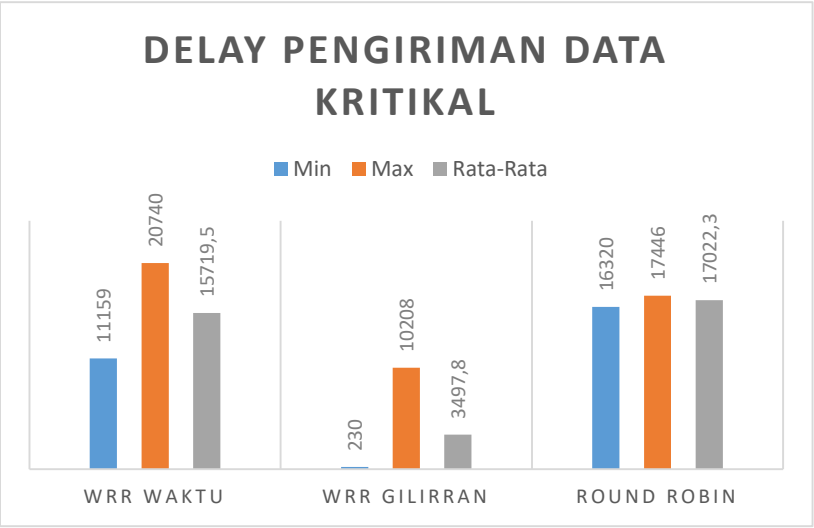
Mengacu pada skenario U-P3 pada bab 5.4.3 telah dilakukan pengujian sesuai dengan skenario. Pengujian dilakukan selama 10 kali dari masing masing metode dan dilakukan sampai data kritikal didapatkan Hasil uji didapatkan dengan melihat log. Berikut adalah tabel hasil uji coba masing masing metode.

Tabel 5.20 Hasil Uji Coba *Delay* Pengiriman Data Kritikal Dari Masing-Masing Metode Pendekatan.

Percobaan	*Delay*(ms)		
	WRR Waktu	WRR Giliran	*Round Robin*
1	11159	1093	16769
2	12838	10208	17446
3	17811	998	17173

4	19141	291	17065
5	12737	9375	16320
6	13683	868	16933
7	20740	1434	17094
8	17257	583	16800
9	18745	9898	17423
10	13084	230	17200
Min	11159	230	16320
Max	20740	10208	17446
Rata-Rata	15719.5	3497.8	17022.3

Mengacu pada tabel Tabel 5.20 bisa dilihat bahwa rata-rata *delay* pengiriman data kritikal pada metode WRR Waktu sebesar 15719.5 ms atau 15.7 Detik. Kemudian untuk metode WRR Giliran memiliki rata-rata *delay* 3497.8 ms atau 3.5 detik. Dan untuk metode Roun Robin memiliki rata rata *delay* sebesar 17022.3 atau 17.02 detik. Dari sini bisa disimpulkan bahwa metode WRR masih memiliki rata-rata *delay* pengiriman data yang lebih kecil jika data tersebut adalah data yang kritikal dibandingkan dengan metode *Round Robin* biasa namun disini bisa dilihat bahwa data *delay* pada 10 percobaan untuk WRR cenderung tidak stabil atau range minimal dan maksimumnya sangat jauh. Sementara untuk *Round Robin* biasa range minimal dan maksimumnya tidak terlalu jauh. Berikut adalah rangkuman dari hasil uji coba *delay* pengiriman data kritikal dari masing-masing metode pendekatan dalam bentuk grafik.



Gambar 5.13 Grafik perbandingan *delay* pengiriman data kritikal dari masing masing metode.

BAB VI
KESIMPULAN DAN SARAN

Pada Bab ini akan diberikan kesimpulan yang diperoleh dari Tugas Akhir yang telah dikerjakan dan saran tentang pengembangan dari Tugas Akhir ini yang dapat dilakukan di masa yang akan datang.

6.1 Kesimpulan

Berdasatkan hasil uji coba dari Tugas akhir ini bisa di simpulkan beberapa hal sebagai berikut:

1. *Weighted Round Robin* (WRR) lebih efektif dalam mengurangi *delay* pengiriman dari *node* sensor ke Android *gateway* dibandingkan dengan *Round Robin* biasa hal ini dibuktikan dengan *delay* WRR Pendekatan waktu sebesar 331,86 ms dan WRR pendekatan giliran sebesar 82,86 ms dibandingkan dengan metode *Round robin* biasa dengan *delay* sebesar 478.95 ms.
2. *Packet Delivery Ratio* dari metode WRR di jarak 3m ,5m, dan7m secara berturut turut adalah 99% – 100%; 98,87% – 99,64%; 95,95%–99,1% dan untuk *Round Robin* biasa sendiri secara berturut turut adalah 97,15%; 98,85% dan 98,24%. dari sini bisa di lihat bahwa WRR memiliki titik atas PDR lebih tinggi dibandingkan metode *Round Robin* biasa.
3. Jarak efektif dari penggunaan *Weighted Round robin* dalam arsitektur WSN adalah 3-5 m hal ini dibuktikan dengan PDR metode WRR sebesar 98,87 – 100%.
4. Range *delay* pengiriman data yang penting ketika menggunakan metode WRR Pendekatan waktu

adalah 11,1–20,7 detik dengan rata-rata *delay* sebesar 15,7 detik. Untuk range *delay* pengiriman data dari metode WRR pendekatan giliran adalah 0,2 – 10,2 detik dengan rata rata *delay* sebesar 3,4 detik. Dan untuk metode *Round Robin* biasa memiliki range *delay* pengiriman data kritikal sebesar 16,3 – 17,4 detik dengan rata-rata *delay* sebesar 17,02 detik. Hal ini membuktikan bahwa untuk pengiriman data kritikal metode WRR pendeketan giliran memiliki performa lebih baik dibandingkan dengan metode lainnya.

6.2 Saran

Saran yang diberikan untuk pengembangan penelitian kedepannya adalah sebagai berikut :

1. Bisa diterapkan ke dalam *node* sensor dengan mikrokontroller *Arduino 101* dimana Arduino tersebut memilki modul Bluetooth yang sudah tertanam di board dari Arduino tersebut. Sehingga tidak perlu menggunakan modul HC tambahan.
2. Menambahkan beberapa metode pendekatan lain seperti shuffle urutan giliran atau mungkin interrupt dari *node* sensor jika memang pada saat itu kondisi bobot sangat jauh dengan bobot *node* lainnya sehingga bisa melakukan pengecekan saat itu juga.

DAFTAR PUSTAKA

[1] R. Margaret, "whatis Tech Terget," [Online]. Available: https://whatis.techtarget.com/definition/sensor. [Diakses 13 10 2017].

[2] Arduino, "Arduino Software(IDE)," [Online]. Available: https://www.arduino.cc/en/Guide/Environment. [Diakses 19 Desember 2017].

[3] Nginx, "Using Round Robin for Simple Load Balancing," [Online]. Available: https://www.nginx.com/resources/glossary/round-robin-load-balancing/. [Diakses 21 7 2018].

[4] IBM, "IBM Knowledge Center - Weighted Round Robin," [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSCVKV_9.1.1/Campaign/Listeners/WeightedRoundRobin.html. [Diakses 13 4 2018].

[5] Wikipedia, "Weighted Round Robin," [Online]. Available: https://en.wikipedia.org/wiki/Weighted_round_robin. [Diakses 20 Maret 2018].

[6] A. E. d. G. Fleishman, The Wireless Networking Starter Kit, Second Edition, Berkeley: Peachpit Press, 2004.

[7] G. Matthew, Wireless Networks: The Definitive Guide, Second Edition, Sebastopol,CA: O,Reilly Media Inc, 2015.

[8] T. Agarwal, "Difference Type of Arduino Board Used by Engineering Student," [Online]. Available:

https://www.elprocus.com/different-types-of-arduino-boards/. [Diakses 19 7 2018].

[9] A. Wibowo, “Apa itu Jaringan Sensor Nirkabel (Wireless Sensor Network)?,” 1 Juni 2016. [Online]. Available: http://jaringansensornirkabel.blogspot.co.id/2016/06/apa-itu-jaringan-sensor-nirkabel.html. [Diakses 19 Desember 2017].

[10] Bluetooth, “What is Bluetooth Technology,” [Online]. Available: https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works . [Diakses 19 Desember 2017].

[11] I. Efendy, “Pengertian dan Kelebihan Arduino,” [Online]. Available: https://www.it-jurnal.com/pengertian-dan-kelebihan-arduino/ [. [Diakses 19 Desember 2017].

[12] RFWireless World, “WSN Wireless Sensor Network,” [Online]. Available: http://www.rfwireless-world.com/Articles/WSN-Wireless-Sensor-Network.html. [Diakses 19 7 2018].

[13] Sinau Arduino, “Mengenal Arduino Software(IDE),” 2016. [Online]. Available: http://www.sinauarduino.com/artikel/mengenal-arduino-software-ide/. [Diakses 20 Desember 2017].

[14] ITEAD, “Serial Port Bluetooth Module (Master/Slave) :HC-05,” [Online]. Available: https://www.itead.cc/wiki/Serial_Port_Bluetooth_Module_(Master/Slave)_:_HC-05. [Diakses 14 3 2018].

[15] Arduino Wikispace, “Bluetooth HC-05 , HC-06 Modules How To,” [Online]. Available: https://arduino-

info.wikispaces.com/BlueTooth-HC05-HC06-Modules-How-To. [Diakses 14 3 2018].

[16] Sunfounder, “Bluetooth Transceiver Module HC-06,” [Online]. Available: http://wiki.sunfounder.cc/index.php?title=Bluetooth_Transceiver_Module_HC-06. [Diakses 22 7 2018].

[17] Pololu, “https://www.pololu.com/file/0J309/MQ2.pdf,” [Online]. Available: https://www.pololu.com/file/0J309/MQ2.pdf. [Diakses 24 Mei 2018].

[18] S. Studio, “Groove - Gas Sensor MQ-2,” [Online]. Available: http://wiki.seeedstudio.com/Grove-Gas_Sensor-MQ2/. [Diakses 22 7 2018].

[19] Wikipedia, “Android (Sistem Operasi),” [Online]. Available: https://id.wikipedia.org/wiki/Android_(sistem_operasi). [Diakses Maret 3 2018].

[20] Arpaci-Dusseau, R. H dan A. C, Operating Systems, Three Easy Pieces Version 0.80, Madison, WI: Arpaci-Dusseau Books, 2014.

[21] What-When-How, “Queuing and Scheduling (QOS-Enabled Networks) Part 2,” [Online]. Available: http://what-when-how.com/qos-enabled-networks/queuing-and-scheduling-qos-enabled-networks-part-2/. [Diakses 22 7 2018].

(Halaman ini sengaja dikosongkan)

LAMPIRAN

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Rayhan Gemaruzman lahir di Jakarta pada 26 Februari 1996. Penulis menempuh pendidikan formal TK Islam Al-Azhar 19 Pamulang (2001-2002), SDI Al-Azhar 15 Pamulang(2002-2008), SMP Al-Azhar BSD(2008-2011), SMAN 2 Tangerang Selatan (2011-2014), dan melanjutkan studi S1 di Departemen Informatika ITS (2014-2018). Bidang studi yang diambil oleh penulis pada saat berkuliah di Departemen Informatika ITS adalah Komputasi Berbasis Jaringan (KBJ). Penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer-Informatika (2015-2016), Badan Eksekutif Mahasiswa Fakultas Teknologi Informasi (2015-2017). Penulis juga aktif dalam kegiatan kepanitian seperti SCHEMATICS 2015 dan SCHEMATICS 2016 divisi *Revolutionary Entertainment and Expo with Various Arts* (REEVA). Penulis pernah kerja praktik di Kantor pusat Bank Indonesia Jakarta periode Januari – Februari 2017. Penulis dapat dihubungi melalui nomor *handphone* 085854048381 atau di email Rayhan_gee26@hotmail.com