TUGAS AKHIR – IF184802

RANCANG BANGUN SISTEM PENGAMATAN LINGKUNGAN MENGGUNAKAN *WIRELESS SENSOR NETWORK* BERBASIS nRF24L01 TERDISTRIBUSI DENGAN LAYANAN *DASHBOARD* UNTUK VISUALISASI DATA PENGAMATAN SECARA *REAL TIME*

UBUT EKA PUTRA
NRP 05111640000008

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

Dosen Pembimbing II
Dr. Eng, Radityo Anggoro, S. Kom, M. Sc

TUGAS AKHIR – IF184802

RANCANG BANGUN SISTEM PENGAMATAN LINGKUNGAN MENGGUNAKAN *WIRELESS SENSOR NETWORK* BERBASIS nRF24L01 TERDISTRIBUSI DENGAN LAYANAN *DASHBOARD* UNTUK VISUALISASI DATA PENGAMATAN SECARA *REAL TIME*

UBUT EKA PUTRA
NRP 05111640000008

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

Dosen Pembimbing II
Dr. Eng, Radityo Anggoro, S. Kom, M. Sc

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS – IF184802

DESIGN AND DEVELOPMENT OF ENVIRONMENTAL OBSERVATION SYSTEM USING nRF24L01 BASED WIRELESS SENSOR NETWORK BASED ON DISTRIBUTION WITH DASHBOARD SERVICE FOR VISUALIZATION OF REAL TIME SAVING DATA

UBUT EKA PUTRA
NRP 05111640000008

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

Second Advisor
Dr. Eng, Radityo Anggoro, S. Kom, M. Sc

INFORMATICS DEPARTMENT
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM PENGAMATAN LINGKUNGAN MENGGUNAKAN *WIRELESS SENSOR NETWORK* BERBASIS nRF24L01 TERDISTRIBUSI DENGAN LAYANAN *DASHBOARD* UNTUK VISUALISASI DATA PENGAMATAN SECARA *REAL TIME*

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Elektro Dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
UBUT EKA PUTRA
NRP: 05111640000008

Disetujui oleh Pembimbing tugas akhir:

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

2. Dr. Eng, Radityo Anggoro, S. Kom, M. Sc .....
   (NIP. 198410162008121002) (Pembimbing 2)

SURABAYA
JULI, 2020

(Halaman ini sengaja dikosongkan)

RANCANG BANGUN SISTEM PENGAMATAN LINGKUNGAN MENGGUNAKAN *WIRELESS SENSOR NETWORK* BERBASIS nRF24L01 TERDISTRIBUSI DENGAN LAYANAN *DASHBOARD* UNTUK VISUALISASI DATA PENGAMATAN SECARA *REAL TIME*

Nama Mahasiswa	**:**	**Ubut Eka Putra**
NRP	**:**	**05111640000008**
Departemen	**:**	**Teknik Informatika FTEIC ITS**
Dosen Pembimbing 1	**:**	**Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Dosen Pembimbing 2	**:**	**Dr. Eng, Radityo Anggoro, S. Kom, M. Sc**

Abstrak

Wireless Sensor Network (WSN) adalah jaringan nirkabel yang tersebar secara terdistribusi yang digunakan dalam jumlah besar untuk memonitor suatu kondisi lingkungan atau sistem oleh pengukuran parameter fisik seperti suhu, tekanan, atau kelembapan. Dimana sensor-sensor node secara tersebar dalam mengumpulkan data-data yang dapat dikirimkan ataupun diolah untuk memonitor suatu lingkungan fisik.

Dengan adanya mekanisme WSN dapat digunakan dalam berbagai pengaplikasian terhadap suatu keadaan ataupun kondisi dimana lingkungan tersebut membutuhkan suatu mekanisme untuk dimonitoring, mengawasi dan mengevaluasi suatu kondisi lingkungan. Dari kondisi tersebut maka dapat dibuat sebuah sistem monitoring untuk memantau suatu kondisi lingkungan.

Sistem pengamatan atau monitoring lingkungan yang di implementasikan yaitu dengan menggunakan mekanisme *Wireless Sensor Network* (WSN) atau komunikasi via jaringan nirkabel untuk mendapatkan data sensor-sensor, data sensor tersebut

didapatkan melalui mikrokontroller yaitu arduino dan terdapat *base station* yaitu raspberry pi sebagai media komunikasi dan pengiriman data ke *database*. Untuk mengirim data dari node sensor ke *base station* menggunakan modul nRF24L01 dan data tersebut disimpan di *database* kemudian data yang telah dikumpulkan divisualisasikan melalui sistem monitoring sensor yaitu layanan *dashboard* untuk menampilkan data secara *realtime*.

Berdasarkan hasil uji coba sistem monitoring sensor yaitu layanan *dashboard* dapat menampilkan data-data yang disimpan di *database* secara *realtime*, serta node sensor dapat mengirimkan data-data sensor ke *database* melalui *base station* yaitu raspberry pi ke *database* melalui jaringan internet. Uji coba telah dilakukan dengan studi kasus kualitas udara dijalan raya dan bekerja dengan baik dalam mengumpulkan dan mengirimkan data. Berdasarkan uji coba performa hasil akurasi pengiriman dengan jarak pengiriman node sensor dengan *base station* yaitu sebagai berikut 99,03% (≤1 m) , 87,22% **(**±5 m), 85,22% (±10 m). Untuk performa *delay realtime* pengiriman data, berdasarkan jumlah node sensor dapat disimpulkan semakin banyak node sensor maka semakin lama *delay realtime* pengiriman data. Sistem pemantauan lingkungan dapat diimplementasikan dalam lingkungan yang secara nyata melalui mekanisme *wireless sensor network*.

.

Kata kunci: Jaringan Sensor Nirkabel, nRF24L01, *Base Station*, Visualisasi Data, Sistem Monitoring Sensor.

DESIGN AND DEVELOPMENT OF ENVIRONMENTAL OBSERVATION SYSTEM USING nRF24L01 BASED WIRELESS SENSOR NETWORK BASED ON DISTRIBUTION WITH DASHBOARD SERVICE FOR VISUALIZATION OF REAL TIME SAVING DATA

Student's Name : Ubut Eka Putra
Student's ID : 05111540000008
Department : Informatics Engineering FTEIC-ITS
First Advisor : Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
Second Advisor : Dr. Eng, Radityo Anggoro, S. Kom, M. Sc

Abstract

Wireless Sensor Network (WSN) is a distributed wireless network that is used in large numbers for monitoring an environment or system by physical measurement parameters such as temperature, pressure, or humidity. Where sensor nodes are distributed as a whole in the data that can be sent or processed for monitoring through the physical environment.

With the WSN mechanism can be used in a variety of applications to a situation or condition where the environment requires a mechanism for monitoring, monitoring and evaluating an environmental condition. From these conditions it can be made a monitoring system to monitor environmental conditions.

The environmental monitoring or monitoring system implemented is by using the Wireless Sensor Network (WSN) mechanism or communication via wireless networks to obtain sensor data, the sensor data is obtained through a microcontroller that is Arduino and there is a base station namely Raspberry Pi as a communication and delivery media. data to the database. To send data from the sensor node to the base station using the nRF24L01 module and the data is stored in a database then the data that has

been collected is visualized through a sensor monitoring system that is a dashboard service to display data in realtime..

Based on the test results of the sensor monitoring system that is the dashboard service can display data stored in the database in real time, and the sensor node can send sensor data to the database through the base station, namely raspberry pi, to the database via the internet network. Trials have been carried out with data quality studios on the streets and work well in collecting and sending data. Based on the test performance of the results of the delivery test with a remote sending sensor node with the following base stations 99.03% (≤1 m), 87.22% (± 5 m), 85.22% (± 10 m). For the performance of late delivery of realtime data based on node sensors, it can be concluded that the more sensor nodes, the longer the delay in sending realtime data. An environmental monitoring system can be implemented in environments that use wireless sensor networks.

Keyword: Wireless Sensor Network, nRF24L01, Base Station, Data Visualization, Sensor Monitoring System.

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul
"RANCANG BANGUN SISTEM PENGAMATAN LINGKUNGAN MENGGUNAKAN *WIRELESS SENSOR NETWORK* BERBASIS nRF24L01 TERDISTRIBUSI DENGAN LAYANAN *DASHBOARD* UNTUK VISUALISASI DATA PENGAMATAN SECARA *REAL TIME*".

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. Dan Nabi Muhammad SAW. yang telah membimbing dan menjadi suri tauladan kehidupan penulis di dunia dan akhirat.
2. Keluarga penulis (Ayah, Ibu, Septi Anugrah Putri, Ammar Asyraf dan keluarga penulis yang lain) yang selalu memberikan dukungan baik berupa doa, moral, dan material yang tak terhingga kepada penulis, sehingga penulis dapat menyelesaikan tugas akhir ini.
3. Bapak Waskitho Wibisono, S.Kom., M.Eng.,Ph.D. dan Bapak Dr. Eng, Radityo Anggoro, S. Kom, M. Sc. selaku Dosen Pembimbing penulis yang telah membimbing, memberikan nasihat, dan memotivasi penulis sehingga penulis dapat menyelesaikan tugas akhir ini.
4. Ibu Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku kepala Departemen Teknik Informatika ITS.

5. Bapak dan Ibu Dosen yang telah memberikan ilmunya selama penulis berkuliah di Informatika ITS.
6. Sahabat baik penulis yaitu Yogik dan Fadli yang selalu menghibur dan mendukung penulis selama bertahun-tahun berkuliah sampai mengerjakan tugas akhir ini dengan sangat baik.
7. Teman-teman dari keluarga besar Laboratorium NCC (Zayn, Ical, Azki, Nuzha, Akmal, Siraj, Adin, Wasil, Ardy, Rapuy, Ivan, Rohman) yang telah menemani, memberi semangat, doa, serta hiburan dikala penulis sedang jenuh saat pengerjaan tugas akhir ini.
8. Teman-teman para penghuni lantai 4 (Mas Fuad, Mas Alvin, Mas Yayan, Mas Djohan, Mas Yoza) yang sudah menemani mengerjakan proyek-proyek selama penulis berkuliah di departemen teknik informatika.
9. Keluarga besar kontrakan barokah, yang selalu memberi support dan dukungan ketika penulis sedang membutuhkan bantuan maupun memberikan dukungan moral dan hiburan.
10. Sahabat dari KMI(Keluarga Muslim Informatika) yang selalui memberi support dan pengaruh yang sangat baik, ketika penulis berkuliah di departemen teknik tnformatika.
11. Teman-teman angkatan 2016 yang sudah menjadi saksi hidup perjalanan karir penulis selama berkuliah di Teknik Informatika ITS.
12. Teman-teman satu bimbingan (Ryanda, Affan, Agung, Yasinta, Chendra) yang selalu memberikan dukungan dan informasi selama mengerjakan tugas akhir ini.
13. Dan juga untuk orang-orang yang tidak dapat disebutkan satu persatu oleh penulis dan pembaca buku tugas akhir ini.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini. Namun, penulis memohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya. Tetaplah berjuang karena

akan ada selalu jalan untuk orang yang tidak pernah menyerah. Semoga kita semua selalu diberi kebahagiaan lahir dan batin dan kesuksesan dunia akhirat. Aamiin.

Surabaya, Juni 2020

Ubut Eka Putra

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Wireless Sensor Network (WSN) adalah jaringan nirkabel yang tersebar secara terdistribusi yang digunakan dalam jumlah besar untuk memonitor suatu kondisi lingkungan atau sistem oleh pengukuran parameter fisik seperti suhu, tekanan, atau kelembapan. Dimana sensor-sensor node secara tersebar dalam mengumpulkan data-data yang dapat dikirimkan ataupun diolah untuk memonitor suatu lingkungan fisik[1]. Dengan adanya teknologi WSN dapat digunakan dalam berbagai pengaplikasian terhadap suatu keadaan ataupun kondisi dimana lingkungan tersebut membutuhkan suatu mekanisme untuk dimonitoring, mengawasi dan mengevaluasi suatu kondisi lingkungan. Dari kondisi tersebut maka dapat dibuat sebuah sistem monitoring untuk memantau suatu kondisi lingkungan.

Sistem pengamatan atau monitoring lingkungan yang akan di implementasikan yaitu dengan menggunakan mekanisme *Wireless Sensor Network* (WSN) atau komunikasi via jaringan nirkabel untuk mendapatkan data sensor-sensor, data sensor tersebut didapatkan melalui mikrokontroller yaitu arduino dimana untuk komunikasi datanya akan menggunakan modul nRF24L01 ,setelah dikumpulkan disebuah *base station yang* di implementasikan menggunakan raspberry pi, kemudian data yang dikumpulkan oleh *base station* akan dikirimkan ke server yang memiliki database sehingga data-data dari database tersebut dapat ditampilkan melalui aplikasi berbasis web di aplikasi berbentuk website tersebut ditampilkan juga informasi kondisi lingkungan berdasarkan data-data sensor yang dikumpulkan.

Dalam tugas akhir ini, diharapkan dapat menghasilkan sebuah rangkaian wireless sensor network di suatu lingkungan dan data yang disimpan dapat ditampilkan dengan layanan dashboard berbasis web secara *real time*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana mendapatkan data-data sensor untuk melakukan monitoring ?
2. Bagaimana rancangan komunikasi nirkabel berbasis nRF24L01 (koneksi antara node sensor dengan *base station*) ?
3. Bagaimana cara mingirimkan data-data yang telah diterima oleh *base station* ke *database* ?
4. Bagaimana membuat rancangan sistem monitoring untuk menyediakan layanan visualisasi sensor secara *real time*?
5. Performa dari keseluruhan sistem yang meliputi:
    a. Berapa akurasi pengiriman data berdasarkan jarak node sensor ke *base station*?
    b. Bagaimana pengaruh *delay realtime* pengiriman terhadap jumlah node sensor?

1.3 Batasan Permasalahan

Berdasarkan masalah yang diuraikan oleh penulis, maka batasan masalah pada tugas akhir ini adalah:

1. Sistem monitoring pemantauan lingkungan yaitu untuk memantau kondisi kualitas udara suatu lingkungan.
2. Menggunakan platform Arduino sebagai mikrokontroller.
3. Menggunakan modul nRF24L01 sebagai media transmisi data.
4. Menggunakan Sensor MQ-2 sebagai sensor untuk mengukur konsentrasi asap atau gas.
5. Menggunakan Sensor MQ-7 sebagai sensor untuk mengukur konsentrasi gas karbon monoksida.
6. Menggunakan Sensor DHT11 sebagai sensor untuk mengukur temperature/suhu udara dan kelembapan.
7. Menggunakan Raspberry Pi 3 b+ sebagai *base station* dan media transmisi untuk mengirimkan data ke database.
8. Menggunakan database Mysql sebagai database pengumpulan data-data sensor.

9. Menggunakan kerangka kerja berbasis php yaitu laravel sebagai kerangka kerja sistem monitoring berbasis web.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut:

1. Membangun sistem rangkaian node *wireless sensor network* yang dapat mengirimkan dan menerima data.
2. Membangun sebuah sistem monitoring pengamatan lingkungan untuk visualisasi data pengamatan secara *realtime*.

1.5 Manfaat

Hasil dari pengerjaan Tugas Akhir ini memiliki manfaat untuk menghasilkan sebuah sistem monitoring lingkungan yang dapat menampilkan data-data sensor yang dikumpulkan dan tersebar dengan menggunakan mekanisme *wireless sensor network.*

Pengerjaan tugas akhir ini juga bermanfaat bagi penulis sebagai sarana untuk mengimplementasikan ilmu-ilmu yang dipelajari penulis selama berkuliah.

1.6 Metodologi

Pembuatan tugas akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Tugas Akhir

Tahap awal pengerjaan tugas akhir ini dimulai dengan penyusunan Proposal tugas Akhir. Proposal tugas akhir ini berisi gambaran tentang gambaran arsitektur sistem *wireless sensor network* yang menggunakan modul nRF24L01 dan juga sensor-sensor maupun mikrokontroller yang digunakan untuk diimplementasikan.

Pendahuluan proposal tugas akhir meliputi hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah yang menjadi konstrain dari tugas akhir, tujuan pembuatan tugas akhir, dan manfaat dari hasil tugas

akhir. Di dalam proposal tugas akhir juga dijabarkan mengenai tinjauan pustaka yang menjadi referensi pendukung dalam pembuatan tugas akhir ini. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

1.6.2 Implementasi Sistem

Implementasi merupakan tahap untuk mengimplementasikan metode-metode yang sudah diajukan pada proposal tugas akhir. implementasi dilakukan dengan menggunakan Arduino sebagai mikrokontroller, nRF24L01 sebagai media komunikasi antara node sensor dengan *base station* yaitu raspberry pi, bahasa C/C++ sebagai bahasa pemrograman Arduino, Bahasa python sebagai bahasa pemrograman di raspberry pi untuk mendapatkan data-data sensor dari node sensor arduino. Dan juga akan menggunakan framework web berbasis php yaitu laravel untuk membuat sistem monitoring lingkungan.

1.6.3 Pengujian dan Evaluasi

Pengujian dan evaluasi dari hasil Tugas Akhir ini akan diujicobakan dengan menempatkan node-node yang disebar untuk mendapatkan data-data agar bisa dimonitoring dan ditampilkan di aplikasi monitoring berbasis web yang berlokasi di jalan raya sekitar tempat tinggal penulis.

1.6.4 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku sebagai dokumentasi dari pelaksanaan tugas akhir yang mencakup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisi penjelasan mengenai latar belakang, rumusan masalah, batasan permasalahan, 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. Kajian teori yang dimaksud berisi tentang penjelasan singkat mengenai *wireless sensor network*, *hardware* yang digunakan serta *software* dan bahasa pemrograman yang digunakan.
3. Bab III. Perancangan Sistem
   Bab ini berisi pembahasan mengenai desain dari *wireless sensor network* yang akan dibuat, meliputi arsitektur dan proses perangkat lunak juga desain perangkat keras.
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi dari desain dari jaringan yang akan dilakukan pada tahap desain atau perancangan, meliputi *pseudocode* program yang terdapat dalam perangkat lunak dan perangkat keras yang digunakan.
5. Bab V. Pengujian dan Evaluasi
   Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.
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 tugas akhir ke depannya.
7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.
8. Lampiran
   Dalam lampiran terdapat dokumentasi saat uji coba dan kode sumber program secara keseluruhan.

(Halaman ini sengaja dikosongkan)

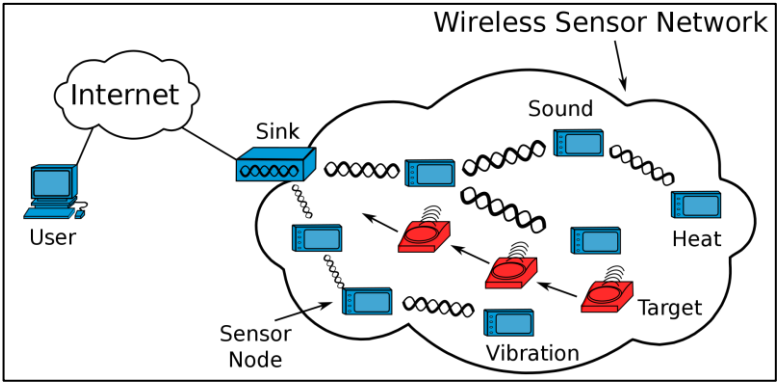
BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar atau penjelasan dari metode dan alat 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 *Wireless Sensor Network*

Wireless Sensor Network (WSN) adalah jaringan nirkabel yang tersebar secara terdistribusi yang digunakan dalam jumlah besar untuk memonitor suatu kondisi lingkungan atau sistem oleh pengukuran parameter fisik seperti suhu, tekanan, atau kelembapan[1].

Dalam implementasinya telah banyak dilakukan menggunakan jaringan sensor nirkabel dalam komunikasi antar node sensor seperti melalui *bluetooth*, Zigbee, frekuensi gelombang ataupun langsung melalui jaringan internet. Penulis akan mengimplementasikan sebuah jaringan sensor nirkabel melalui frekuensi gelombang radio. Gambar 2.1 merupakan contoh ilustrasi dari *wireless sensor network*.



Gambar 2.1 Ilustrasi *Wireless Sensor Network* [16]

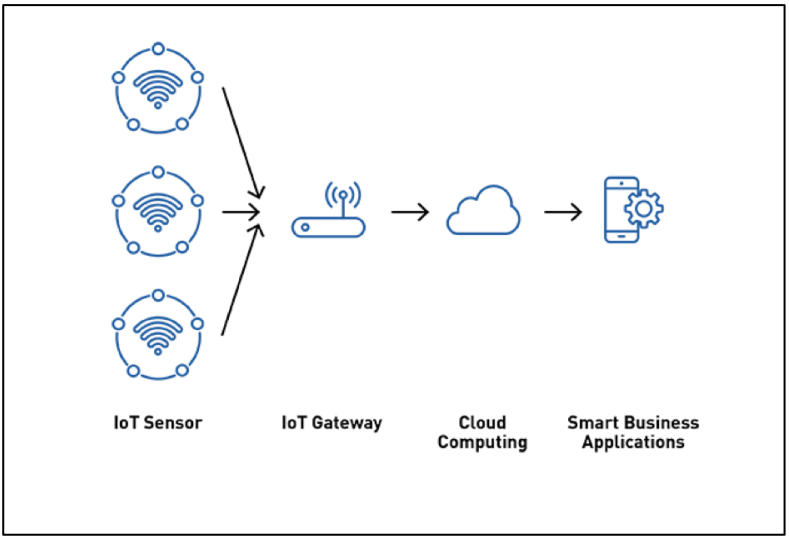
2.2 *Base Station*

Base station dikenal sebagai node gateway atau sink. *Base station* mengoordinasikan jaringan, menerima data dari node sensor, dan memproses dan menyimpannya ke dalam sistem manajemen informasi. Untuk tujuan tersebut, *base station* harus menyertakan beberapa elemen yang bertanggung jawab untuk melakukan tugas-tugas utama. Pertama-tama, diperlukan sirkuit nirkabel yang efisien untuk mengimplementasikan komunikasi dengan node sensor. Kedua, sistem komunikasi secara redundan yang mendukung komunikasi luar harus disertakan. *Base station* juga harus menyediakan antarmuka yang sederhana untuk memungkinkan melakukan operasi dasar[2].

2.3 Sensor *Cloud*

Sensor *Cloud* yaitu mengumpulkan dan memproses informasi dari beberapa jaringan sensor, memungkinkan berbagi informasi dalam skala besar, dan berkolaborasi dengan aplikasi di cloud di antara pengguna. Sensor *Cloud* mengintegrasikan beberapa jaringan dengan sejumlah aplikasi *sensing* dan platform komputasi awan dengan memungkinkan aplikasi menjadi lintas penghubung yang dapat menjangkau beberapa organisasi. Sensor-Cloud memungkinkan pengguna untuk dengan mudah mengumpulkan, mengakses, memproses, memvisualisasikan, menganalisis, menyimpan, berbagi, dan mencari sejumlah besar data sensor dari beberapa jenis aplikasi dan dengan menggunakan IT komputasi dan sumber daya penyimpanan *cloud*[3].

Berdasarkan informasi mengenai sensor cloud, penulis akan mengimplementasikan sistem monitoring sensor dimana data-data sensor tersebut disimpan ke sebuah *cloud* server yang dapat menyimpan data secara langsung sehingga data tersebut dapat digunakan dan divisualisasikan di sistem monitoring sensor yaitu layanan *dashboard* untuk visualisasi data. Gambar 2.2 merupakan ilustrasi umum sensor *cloud*.



Gambar 2.2 Ilustrasi Sensor *Cloud* [17]

2.4 *Mobile Crowdsensing*

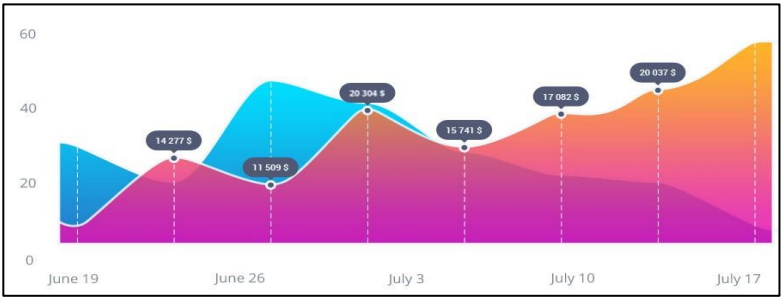
Mobile crowdsensing mengacu pada berbagai macam model penginderaan/*sensing* di mana individu secara kolektif berbagi data dan ekstrak informasi untuk mengukur dan memetakan fenomena kepentingan bersama. Dalam pengaplikasian MCS di lingkungan, contohnya adalah dari lingkungan alami. Yaitu termasuk mengukur tingkat polusi di kota, tingkat air di anak sungai, dan pemantauan habitat satwa liar. Pengaplikasian semacam itu memungkinkan pemetaan berbagai fenomena lingkungan berskala besar dengan melibatkan individu[4].

Dalam tugas akhir ini penulis mengimplementasikan sistem monitoring dengan batasan yaitu memantau kualitas udara, dimana data-data sensor dikumpulkan melalui node-node sensor yang diletakkan secara tersebar.

2.5 Visualisasi Data

Visualisasi data dilihat oleh banyak bidang ilmu sebagai komunikasi visual modern. Visualisasi data tidak berada di bawah bidang manapun, melainkan interpretasi di antara banyak bidang (misalnya, terkadang dilihat sebagai cabang modern dari statistik deskriptif oleh beberapa orang, tetapi juga sebagai dasar alat pengembangan oleh yang lain). Visualisasi data mengikutkan pembuatan dan kajian dari representasi visual dari data, artinya "informasi yang telah diabstraksikan dalam bentuk skematis, termasuk atribut atau variabel dari unit informasi"[5].

Tujuan utama dari visualisasi data adalah untuk mengkomunikasikan informasi secara jelas dan efisien kepada pengguna lewat grafik informasi yang dipilih, seperti tabel dan grafik. Visualisasi yang efektif membantu pengguna dalam menganalisis dan penalaran tentang data dan bukti. Ia membuat data yang kompleks bisa diakses, dipahami dan berguna. Pengguna bisa melakukan pekerjaan analisis tertentu, seperti melakukan pembandingan atau memahami kausalitas, dan prinsip perancangan dari grafik (contohnya, memperlihatkan perbandingan atau kausalitas) mengikuti pekerjaan tersebut. Tabel pada umumnya digunakan saat pengguna akan melihat ukuran tertentu dari sebuah variabel, sementara grafik dari berbagai tipe digunakan untuk melihat pola atau keterkaitan dalam data untuk satu atau lebih variabel. Gambar 2.3 merupakan contoh dari visualisasi data dalam bentuk grafik.



Gambar 2.3 Contoh Visualisasi Data Grafik [18]

2.6 Raspberry Pi

Raspberry Pi, sering disingkat dengan nama Raspi, adalah komputer papan tunggal (*single-board circuit*; SBC) yang seukuran dengan kartu kredit yang dapat digunakan untuk menjalankan program perkantoran, permainan komputer, dan sebagai pemutar media hingga video beresolusi tinggi. Raspberry Pi dikembangkan oleh yayasan nirlaba, Rasberry Pi *Foundation*, yang digawangi sejumlah pengembang dan ahli komputer dari Universitas Cambridge, Inggris.Raspberry Pi memiliki dua model: model A dan model B. Secara umum Raspberry Pi Model B memiliki kapasitas penyimpanan RAM sebesar 512 MB. Perbedaan model A dan B terletak pada modul penyimpanan yang digunakan. Model A menggunakan penyimpanan sebesar 256 MB dan penyimpanan model B sebesar 512 MB. Selain itu, model B sudah dilengkapi dengan port Eternet (untuk LAN) yang tidak terdapat di model A. Desain Raspberry Pi didasarkan pada SoC (*sistem-on-a-chip*) Broadcom BCM2835, yang telah menanamkan prosesor ARM1176JZF-S dengan 700 MHz, GPU VideoCore IV, dan RAM sebesar 256 MB (model B). Penyimpanan data tidak didesain untuk menggunakan cakram keras atau *solid-state drive*, melainkan mengandalkan kartu penyimpanan tipe SD untuk menjalankan sistem dan sebagai media penyimpanan jangka Panjang[6]. Gambar 2.4 merupakan contoh dari raspberry pi 3 model b.

Gambar 2.4 Raspberry Pi 3 Model B

2.7 Arduino

Arduino merupakan sebuah mikrokontroler single-board yang bersifat open-source. Arduino dirancang sedemikian rupa sehingga memudahkan para penggunanya dibidang elektronika. *Board* Arduino didesain menggunakan processor Atmel AVR dan mendukung masukan dan keluaran pada *board*-nya. Bahasa pemrograman yang digunakan adalah C/C++. Dalam sebuah mikrokontroler Arduino dapat pula ditanamkan berbagai macam *library* maupun metode selama kapasitas memorinya mencukupi[7].

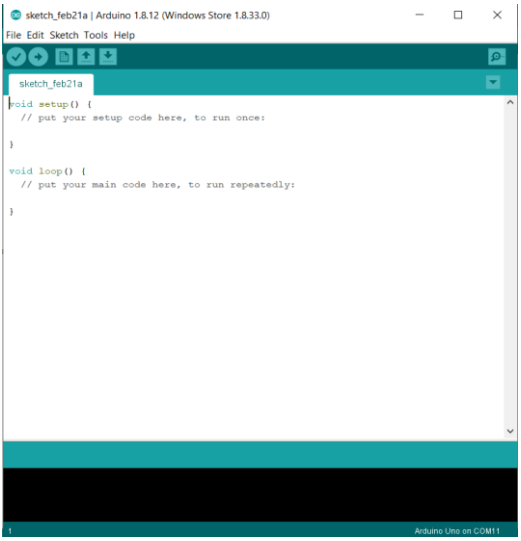
Arduino uno di tugas akhir ini akan digunakan sebagai mikrokontroller dari tiap-tiap node sensor. Dimana arduino dikenal sebagai mikrokontroller yang mudah untuk digunakan dan disambungkan ke berbagai sensor untuk mengambil data-data sensor di lingkungan.

Gambar 2.5 Arduino Uno

2.8 Arduino IDE

Arduino *Integrated Development Environment* (IDE) merupakan editor teks untuk menulis kode, area pesan, konsol teks dan memiliki toolbar dengan tombol untuk fungsi umum dan serangkaian menu. Arduino IDE terhubung ke perangkat keras Arduino dan Genuino untuk mengunggah program dan dapat berkomunikasi dengan mereka. Program yang ditulis menggunakan Arduino Software (IDE) disebut sketsa. Sketsa ini

ditulis dalam editor teks dan disimpan dengan ekstensi file .ino. Editor memiliki fitur untuk memotong / menempel dan mencari / mengganti teks. Area pesan memberi umpan balik saat menyimpan dan mengekspor dan menampilkan kesalahan. Konsol menampilkan output teks oleh Arduino Software (IDE), termasuk pesan kesalahan dan informasi lainnya yang lengkap[8].



Gambar 2.6 Arduino IDE

2.9 nRF24L01

nRF24L01 adalah modul komunikasi serial nirkabel yang menggunakan frekuensi 2,4 GHz. Dilengkapi dengan sirkuit *Low Noise Amplifier* (LNA) dan *Power Amplifier* (PA). nRF24l01 dapat mentransmisikan data hingga jarak 1100 meter. Kecepatan transmisi data dari modul NRF24L01 ini dapat mencapai 2 Mbps, dengan 125 pilihan *multiple frequency* yang dimodulasi dengan algoritma GFSK[9].

nRF24L01 digunakan sebagai media komunikasi antar node agar dapat mengirimkan dan menerima data, selain modul ini

sangat mudah didapatkan, modul ini juga dapat terhubung dengan mikrokontroller arduino uno dan raspberry pi yang nantinya digunakan sebagai node sensor dan *base station*. Dengan *listening address* yang dideklarasikan terlebih dahuulu maka sudah dapat mengirimkan dan menerima data sehingga nantinya dapat diimplementasikan ke *base station* yang harus menerima langsung sekaligus data-data dari node sensor, dan node sensor juga terhubung modul nRF24L01 untuk mengirimkan data-data sensor.

Gambar 2.7 nRF24L01

2.10 Python

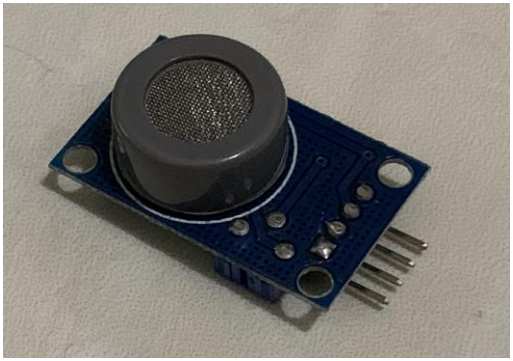
Python adalah bahasa pemrograman *interpretative* multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Python juga didukung oleh komunitas yang besar. Python mendukung multi paradigma pemrograman, utamanya; namun tidak dibatasi; pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang

dilengkapi dengan manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa skrip. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi[10].

Bahasa pemrograman python digunakan oleh *base station* yaitu raspberry pi dalam menerima data dari node sensor dan mengirimkan data-data tersebut ke *database*, dikarenakan untuk menggunakan modul nRF24L01 telah terdapat library untuk menghubungkan raspberry pi dengan modul nRF24L01.

2.11 Sensor MQ-7

Sensor MQ-7 merupakan sensor gas karbon monoksida yang berfungsi untuk mengetahui konsentrasi gas karbon monoksida (CO), sensor MQ7 memiliki sensitivitas tinggi dan respon cepat terhadap gas karbon monoksida dan keluaran dari sensor MQ7 berupa sinyal analog dan membutuhkan tegangan DC sebesar 5Volt[11].



Gambar 2.8 Sensor MQ-7

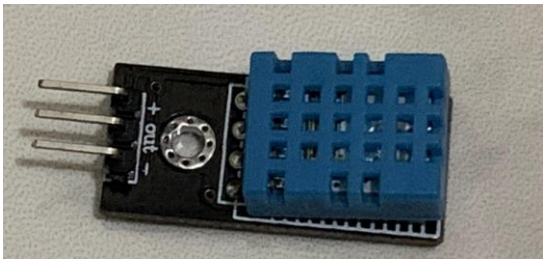
2.12 Sensor MQ-135

Sensor MQ-135 adalah sensor yang memonitor kualitas udara untuk mendeteksi gas amonia, natrium dioksida, alkohol/ethanol, benzena, karbondioksida, gas belerang/sulfur-hidroksida dan asap atau gas-gas lainnya di udara. Sensor ini melaporkan hasil deteksi kualitas udara berupa perubahan nilai resistensi analog di pin keluarannya. Pin keuaran ini bisa disambungkan dengan pin adc (analog-to-digital converter) di mikrokontroller / pin analoh input arduino dengan menambahkan satu buah resistor saja (befungsi sebagai pembagi tegamgan atau voltage divider)[12].

Gambar 2.9 Sensor MQ-135

2.13 Sensor DHT11

DHT11 adalah salah satu sensor yang dapat mengukur dua parameter lingkungan sekaligus, yakni suhu dan kelembaban udara (*humidity*). Dalam sensor ini terdapat sebuah thermistor tipe NTC (*Negative Temperature Coefficient*) untuk mengukur suhu, sebuah sensor kelembaban tipe resisitif dan sebuah mikrokontroller 8-bit yang mengolah kedua sensor tersebut dan mengirim hasilnya ke pin output dengan format single-wire bi-directional (kabel tunggal dua arah)[13].



Gambar 2.10 Sensor DHT11

2.14 Laravel

Laravel adalah kerangka kerja berbasis PHP untuk membangun aplikasi web kelas atas menggunakan sintaksis yang signifikan dan elegan. Muncul dengan kumpulan tools yang beragam dan menyediakan arsitektur aplikasi. Selain itu, laravel mencakup berbagai karakteristik teknologi seperti ASP.NET MVC, CodeIgniter, Ruby on Rails, dan banyak lagi. Kerangka kerja ini adalah kerangka kerja open-source. laravel memfasilitasi pengembang dengan menghemat waktu dan membantu mengurangi pemikiran dan perencanaan untuk mengembangkan seluruh situs web dari awal. Bersamaan dengan itu, keamanan aplikasi juga diperhatikan dengan laravel. Oleh karena itu semua fitur-fiturnya dapat meningkatkan kecepatan pengembangan web. Jika terbiasa dengan dasar-dasar PHP bersama dengan beberapa skrip PHP tingkat menengah, maka laravel dapat membuat pekerjaan lebih efisien[14].

2.15 Bootstrap

Bootstrap adalah library (pustaka / kumpulan fungsi-fungsi) dari framework CSS yang dibuat khusus untuk bagian pengembangan fontend dari suatu website. Didalam library tersebut terdapat berbagai jenis file yang diantaranya HTML, CSS, dan Javascript. Hampir semua *developer* website menggunakan *framework* bootstrap agar memudahkan dan mempercepat pembuatan website. Karena semuanya sudah ada dalam *framework* nya sehingga para *develop* / pengembang hanya tinggal membuat /

menyisipkan *class* nya yang ingin dipakai seperti membuat tombol, grid navigasi dan lain sebagainya.

Bootstrap telah menyediakan kompulan aturan dan komponen *class interface* dasar sebagai modal dalam pembuatan web yang telah dirancang sangat baik untuk memberikan tampilan yang sangat menarik, bersih, ringan dan memudahkan bagi penggunanya. Dan penggunaan bootstrap ini kita juga diberikan keleluasan salama pengembangan website, anda bisa merubah dan menambah *class* sesuai dengan keinginan[15].

(***Halaman ini sengaja di kosongkan***)

BAB III
PERANCANGAN SISTEM

Perancangan sistem merupakan bagian penting dalam pembuatan perangkat lunak dan perangkat keras yang berupa perencanaan secara teknis dari sistem yang dibuat. Pada bab ini akan dibahas mengenai beberapa hal secara umum yaitu deskripsi umum sistem, arsitektur umum sistem, diagram kasus penggunaan, perancangan basis data, diagram alur dan desain antarmuka perangkat lunak sistem monitoring sensor.

3.1 Deskripsi Umum

Pada tugas akhir ini akan dibuat sebuah Sistem pengamatan lingkungan untuk monitor suatu lingkungan dengan menerapkan *wireless sensor network* melalui komunikasi nRF24L01 dengan node-node sensor secara tersebar. Untuk penerapannya, ada 3 bagian utama yang akan dibangun yaitu node sensor dengan menggunakan mikrokontroller Arduino, *base station* dengan menggunakan raspberry pi, dan monitoring sensor berbentuk website untuk monitoring sensor-sensor yang tersebar.

Node sensor yang akan dibangun menggunakan mikrokontroller Arduino UNO, untuk dapat berkomunikasi dengan *base station*, node sensor akan dihubungkan dengan modul nRF24L01. Komunikasi yang akan dibangun merupakan komunikasi secara nirkabel dengan memanfaatkan gelombang RF 2,4GHz.

Untuk mendapatkan data-data yang dikirimkan oleh node-sensor tersebar maka diperlukan adanya media untuk menerima data-data tersebut, maka juga harus membangun *base station* sebagai media menerima data dari node-sensor melalui modul nRF24L01 dan data-data yang diterima akan dikirim ke database melalui jaringan internet, untuk penerapannya *base station* akan menggunakan raspberry pi 3 model B+, dimana raspberry pi versi ini telah tertanam modul *wifi* sehingga dapat terkoneksi secara langsung ke jaringan internet secara nirkabel.

Bagian terakhir yang dibangun adalah monitoring sensor yaitu layanan *dashboard* yang berfungsi untuk memvisualisasikan data-data sensor yang tersebar. Fungsi-fungsi yang ada pada website meliputi, mendaftarkan sensor, menampilkan list sensor dan lokasi *base station*,menambahkan lokasi *base station* menampilkan informasi detail data-data setiap sensor, dan mendaftarkan *rule based* informasi kondisi lingkungan.

3.2 Arsitektur Umum Sistem

Pada sub bab ini akan dijelas mengenai arsitektur umum sistem yang akan bangun. Gambaran sistem secara umum yang dibuat dapat dilihat melalui gambar 3.1, berdasarkan gambar tersebut akan dibuat node sensor secara tersebar berjumlah 3 node, dan terdapat media penerima dan mengirimkan data melalui node sensor ke database via jaringan internet dengan menggunakan raspberry pi, setelah itu data yang tersimpan di *database* akan ditampilkan melalui layanan *dashboard* untuk divisualisasikan.

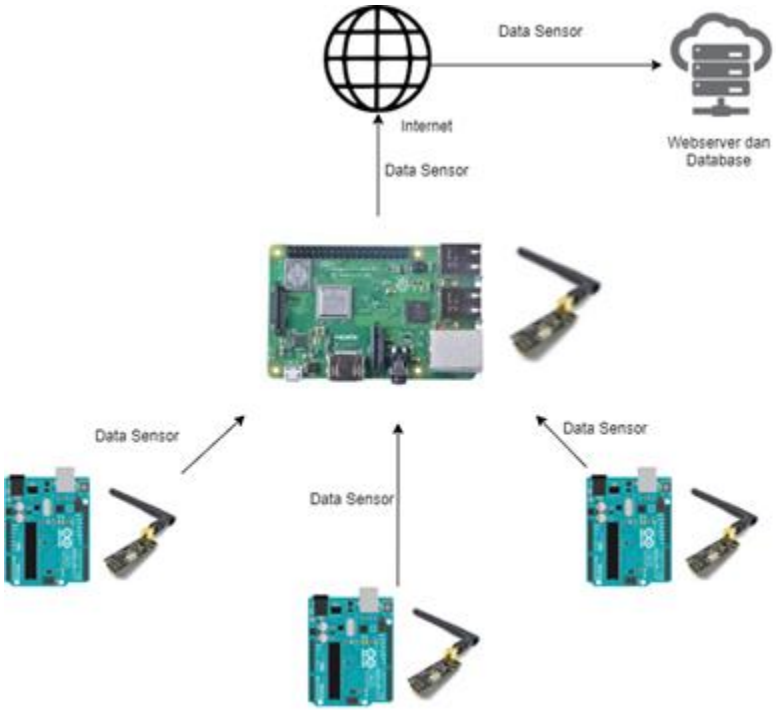
Base station merupakan node dimana berfungsi sebagain media perantara untuk menerima data dari beberapa node, sehingga data yang dikumpulkan dapat diterima dan dikirimkan ke suatu penyimpanan data yaitu database sehingga data tersebut dapat ditampilkan untuk mengetahui kondisi lingkungan yang ada.

Dalam berkomunikasi dapat dilihat melalui Gambar 3.1, media komunikasi antara node yang mengumpulkan data sensor dan *base station* yaitu raspberry pi, yaitu menggunakan modul nRF24L01, modul tersebut merupakan media komunikasi jarak jauh secara nirkabel yang menggunakan gelombang frekuensi radio 2,4 GHz, dimana modul ini menggunakan konsumsi daya yang rendah, selain itu juga kecepatan transmisi modul ini mencapat 2Mbps dan menggunakan tegangan 3,3V–5V untuk menggunakan modul ini. Modul ini juga mendukung antarmuka SPI(*Serial Paralel Interface*) sehingga modul ini dapat terhubung dengan mikrokontroller arduino dan juga raspberry pi.

Di arsitektur sistem secara umum ini terdapat juga node sensor yang berperan penting untuk mendapatkan data-data dari

ssensor yang akan diimplementasi dengan beberapa sensor yaitu sensor gas CO MQ-7, sensor gas kualitas udara MQ-135 dan sensor DHT11 untuk mengukur temperatur dan kelembapan suatu kondisi lingkungan.

Setelah *base station* dapat menerima data sensor yang dikirimkan oleh node sensor maka *base station* dapat meneruskan data yang dikumpulkan dengan mengirimkan data tersebut melalui jaringan internet ke suatu server yang memiliki database, setelah data itu tersimpan maka data tersebut akan ditampilkan di sebuah sistem monitoring sensor berbasis website agar dapat di amati.



Gambar 3.1 Arsitektur Umum Sistem

3.3 Perancangan Node Sensor

Pada bagian ini akan dijelaskan mengenai node sensor, node sensor berfungsi sebagai node yang mengumpulkan dari *sensing* kondisi lingkungan, yaitu mengumpulkan data-data dari beberapa sensor, yang akan diimplementasikan menggunakan mikrokontroller arduino uno, arduino akan terhubung dengan beberapa sensor yaitu sensor MQ-7 untuk mengukur konsentrasi gas karbonmonoksida, sensor MQ-135 untuk mendeteksi gas amonia, natrium dioksida, alkohol/ethanol, benzena, karbondioksida, gas belerang/sulfur-hidroksida dan asap atau gas-gas lainnya di udara, dan sensor DHT11 untuk mengukur tingkat temperatur atau suhu dan kelembapan udara (*humidity*). Untuk komunikasinya menggunakan modul nRF24L01.

Pembahasan di sub bab ini juga menjelaskan mekanisme cara kerja node sensor dalam mengumpulkan data sensor dan mengirimkan data sensor dengan diagram alir.

3.3.1 Perancangan Rangkaian Utama

Rangkaian utama dari node sensor dan beberapa komponen-komponen penting untuk rancangannya akan dijelaskan dengan Tabel 3.1.

Tabel 3.1 Komponen Rangkaian Utama

No	Nama Komponen	Deskripsi	Jumlah
1	Arduino UNO	Merupakan sebuah mikrokontroller yang berfungsi sebagai komponen utama dan bisa disebut sebagai otak dari komponen itu sendiri, arduino yang digunakan adalah arduino uno rev 3.	1 buah
2	Modul nRF24L01	Merupakan modul komunikasi secara nirkabel	1 buah

		yang memanfaatkan frekuensi gelombang radio 2,4GHz.	
3	MQ-7	Merupakan sensor yang berfungsi untuk mengukur konsentrasi gas CO(karbonmoniksida).	1 buah
4	MQ-135	Merupakan sensor yang berfungsi untuk mengukur kualitas udara secara umum dapat mendeteksi gas-gas seperti gas amonia, natrium dioksida, alkohol/ethanol, benzena, karbondioksida, gas belerang/sulfur-hidroksida dan asap atau gas-gas lainnya di udara.	1 buah
5	DHT11	Merupakan modul sensor yang berfungsi untuk mengukur temperatur dan kelembapan suatu lingkungan.	1 buah
6	*Breadboard*	Sebuah papan yang biasa disebut juga sebagai *project board* yang berfungsi untuk merangkai rangkaian elektronik.	1 buah
7	Kabel *Jumper*	Sebuah kabel elektrik yang menghubungkan kompenen dan komponen lainnya.	10-15 buah

Setiap komponen-komponen akan dirangkai menjadi satu rangkaian utama sehingga menjadi satu rangkaian yaitu node

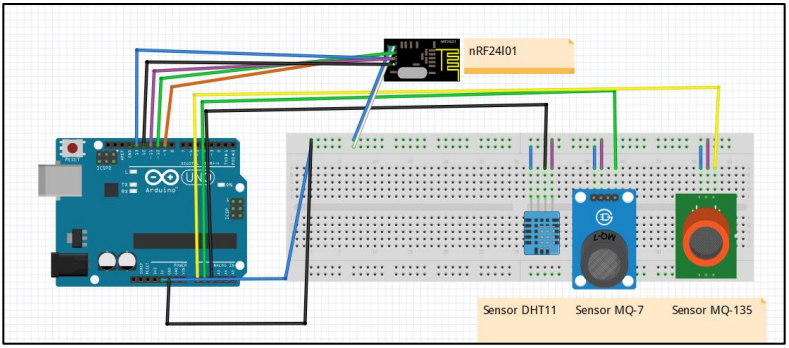
sensor, karena sistem membutuhkan 3 node sensor maka, dari komponen tersebut membutuhkan 3 kali komponen tersebut. Berikut merupakan rancangan node sensor setelah dirangkai menjadi satu node sensor. Di komponen tersebut juga terdapat modul nRF24L01 dimana berfungsi sebagai modul komunikasi antara node sensor dan *base station* dengan memanfaatkan frekuensi gelombang radio. Untuk terhubungnya koneksi pin antar komponen bisa dilihat berdasarkan Tabel 3.2 berikut.

Tabel 3.2 Koneksi *PIN* Antar Komponen

PIN	**Modul/Komponen**	**Deskripsi**
	POWER	
5V	*Breadboard (VCC)*	Sebagai *VCC*
GND	*Breadoard (GND)*	Sebagai *GROUND*
	ANALOG IN	
A0	*MQ-7*	Terhubung dengan sensor MQ-7 analog input
A1	*MQ-135*	Terhubung dengan sensor MQ-135 analog input
A2	*DHT11*	Terhubung dengan pin tengah analog input sensor DHT11
	PIN	
9	*nRF24L01*	Terhubung dengan nRF24L01 pin CE
10	*nRF24L01*	Terhubung dengan nRF24L01 pin CSN
11	*nRF24L01*	Terhubung dengan nRF24L01 pin MOSI

12	*nRF24L01*	Terhubung dengan nRF24L01 pin MISO
13	*nRF24L01*	Terhubung dengan nRF24L01 pin SCK

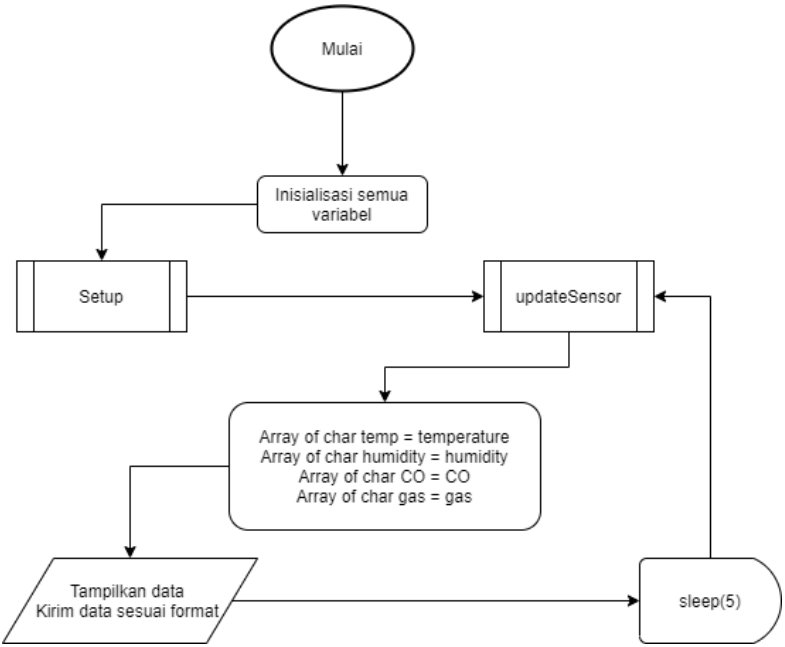
Setelah rangkaian dan komponen semuanya terhubung melalui rancangan seperti sebelumnya dijelaskan, maka rancangan rangkaian node sensor, dapat digambarkan seperti Gambar 3.2 berikut.



Gambar 3.2 Rancangan Node Sensor

3.3.2 Perancangan Diagram Alir Node Sensor

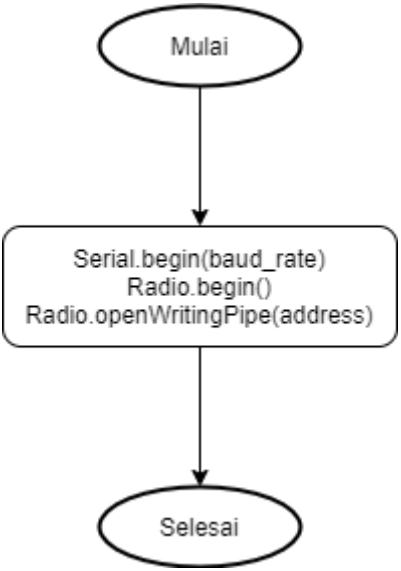
Diagram alir node sensor merupakan gambaran secara umum bagaimana node sensor bekerja, yaitu cara kerja program yang dirancang sehingga node sensor dapat mengirimkan data dari beberapa sensor ke *base station* (raspberry pi). Diagram alir node sensor dapat dilihat melalui Gambar 3.3.



Gambar 3.3 Diagram Alir Node Sensor

Secara umum, gambaran logika jalannya program terbagi menjadi 2 di pemrograman arduino, yaitu subproses *setup* dan *updateSensor*. Subproses setup berfungsi sebagai menginisiasi jalannya mikrokontroller yang berjalan hanya satu kali saat arduino dihidupkan, sedangkan *updateSensor* yang berjalan di fungsi *loop* arduino berfungsi untuk mengeksekusi bagian-bagian program yang dijalankan secara berulang-ulang. Dalam subproses *updateSenso*r terdapat 4 fungsi yang akan dipanggil, fungsi-fungsi tersebut digunakan untuk mendapatkan masing-masing nilai dari sensor. Nilai-nilai sensor yang didapatkan akan disimpan sebuah variabel agar dapat disesuaikan format pengiriman data, dengan *base station* yang menerima data.

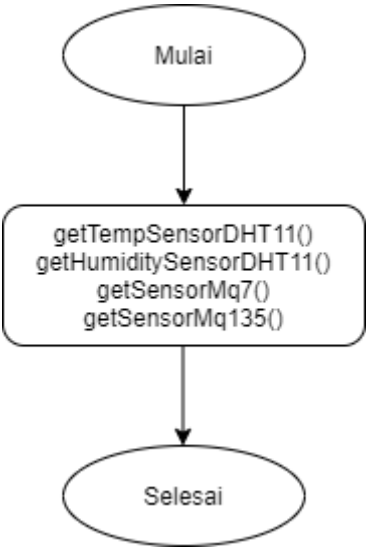
3.3.2.1 Diagram Alir Subproses Setup



Gambar 3.4 Diagram Alir Subproses Setup

Pada subproses setup ini, berperan sebagai inisialisasi pertama kali ketika arduino dihidupkan atau tersambung dengan suatu daya listrik, yaitu program menjalankan *Serial.begin(baud_rate),* nilai *baud_rate* adalah nilai yang digunakan untuk menjalankan komponen, nilai yang digunakan adalah 9600, sedangkan Radio.begin untuk menginisialisasi nRF24L01 dan *Radio.openWritingPipe(address)* untuk menginisialisasi alamat tujuan frekuensi gelombang agar dapat mengirim data. Untuk setiap node sensor *address* yang dimiliki juga berbeda-beda sehingga node sensor memiliki jalur frekuensi yang berbeda untuk mengirimkan data. Gambar 3.4 merupakan diagram alir subproses setup.

3.3.2.2 Diagram Alir Subproses updateSensor



Gambar 3.5 Diagram Alir Subproses updateSensor

Pada subproses ini, setiap sensor akan terus diperbarui data-data yang diterima yang dijalankan di fungsi *loop*, fungsi ini akan selalu berjalan selama mikrokontroller arduino hidup, fungsi-fungsi yang di panggil yaitu *getTempSensorDHT11()*, fungsi ini berperan untuk mendapatkan data temperatur dari sensor dht11, *getHumiditySensorDHT11()* berperan untuk mendapatkan data kelembapan dari sensor dht11, *getSensorMq7()* berperan untuk mendapatkan data konsentrasi gas CO dari sensor mq-7, *getSensorMq135()* berperan untuk mendapatkan data konsentrasi gas dari sensor mq-135.

3.4 Perancangan *Base station*

Pada bagian ini akan dijelaskan mengenai rancangan rangkaian utama dari *base station* yaitu menggunakan raspberry pi, dengan media komunikasinya menggunakan nRF24L01. Pada

perancangan *base station* juga dijelaskan mengenai susunan rangkaian komponen-komponen terhubung sehingga menjadi rangkaian yang dapat menerima data dari node sensor dan juga dapat mengirimkan data ke database.

Pembahasan lain mengenai perancangan *base station* yaitu membahas perancangan rangkaian dan juga mekanisme kerja *base station* yang digambarkan melalui diagram alir.

3.4.1 Perancangan Rangkaian *Base station*

Rangkaian dari *base station* dan beberapa komponen-komponen penting untuk rancangannya akan dijelaskan dengan Tabel 3.3.

Tabel 3.3 Komponen Rangkaian *Base station*

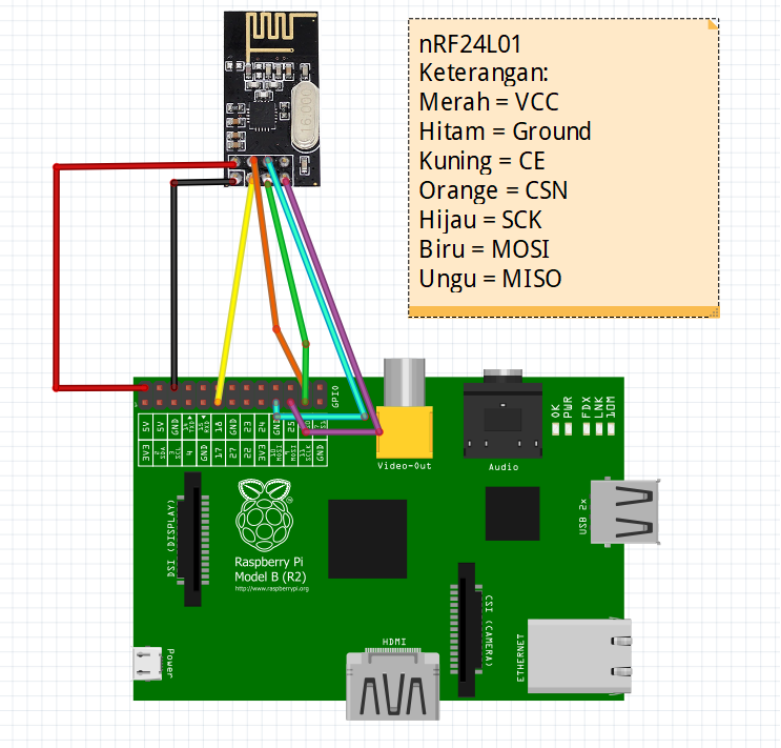
No	Nama Komponen	Deskripsi	Jumlah
1	Raspberry pi 3 B+	Merupakan mini komputer seukuran kartu atm yang dapat terhubung dengan *mouse* dan *keyboard*, untuk versi ini terdapat modul wifi sehingga dapat langsung terhubung ke internet secara nirkabel.	1 buah
2	Modul nRF24L01	Merupakan modul komunikasi secara nirkabel yang memanfaatkan frekuensi gelombang radio 2,4GHz.	1 buah
7	Kabel *Jumper (Female to Female)*	Sebuah kabel elektrik yang menghubungkan kompenen dan komponen lainnya.	7 buah

Agar raspberry pi dapat terhubung dengan modul nRF24L01, maka dihubungkan melalui kabel jumper dengan menghubungkan pin-pin yang sesuai, raspberry pi sendiri memiliki GPIO, yaitu pin atau terminal tambahan untuk input atau output agar dapat terhubung dengan komponen atau perangkat lain. Untuk lebih jelasnya koneksi pin ditampilkan di tabel berikut.

Tabel 3.4 Koneksi *PIN/GPIO* Antar Komponen

PIN/GPIO	**Modul/Komponen**	**Deskripsi**
	POWER	
2/5V	*nRF24L01 (VCC)*	Sebagai *VCC*
6/GND	*nRF24L01 (GND)*	Sebagai *GROUND*
	GPIO	
11/GPIO 17	*nRF24L01* PIN CE	Terhubung dengan nRF24L01 pin CE
24/GPIO 8	*nRF24L01* PIN CSN	Terhubung dengan nRF24L01 pin CSN
19/GPIO 10	*nRF24L01* PIN MOSI	Terhubung dengan nRF24L01 pin MOSI
21/GPIO 9	*nRF24L01* PIN MISO	Terhubung dengan nRF24L01 pin MISO
23/GPIO 11	*nRF24L01* PIN SCK	Terhubung dengan nRF24L01 pin SCK

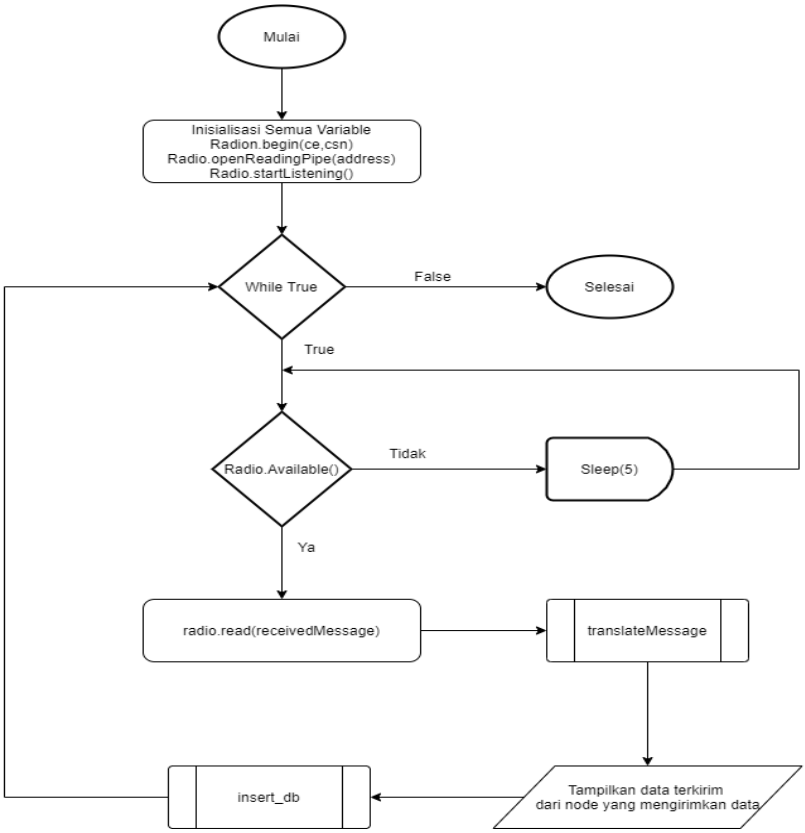
Setelah komponen ranngkaian dan koneksi pin dihubungkan, maka dapat dibuat rancangan untuk *base station* yaitu raspberry pi dan modul nRF24L01, rancangan yang diimplementasikan menggunakan beberapa kabel *jumper female to female*, dengan koneksi GPIO raspberry pi, sehingga modul nRF24L01 dapat terhubung, rancangannya dapat dilihat melalui Gambar 3.6.



Gambar 3.6 Rancangan *Base station*

3.4.2 Perancangan Diagram Alir *Base station*

Pada bagian ini akan dijelaskan mengenai digaram alir tentang alur dan cara kerja, bagaimana *base station* dapat menerima data dari beberapa node sensor melalui modul nRF24L01 dengan memanfaatkan frekuensi gelombang radio. Diagram ini digunakan sebagai acuan untuk membuat alur program yang akan diimplementasikan.



Gambar 3.7 Diagram Alir *Base station*

Pada Gambar 3.7, mula-mula program menginisialisasi semua variabel yaitu inisialisasi nRF24L01 agar dapar terhubung dengan raspberry pi, dengan menyesuaikan parameter radio.begin(ce,csn) dengan pin ce dan csn yang telah dijelaskan pada tabel 3.4. Setelah menginisialisasi program menjalankan *looping while true* dikarenakan program diharapkan dapat berjalan terus-menerus sampai program dihentikan. Setelah itu program mengecek apakah data sedang masuk atau tidak melalui fungsi *radio.available()* jika tidak maka masuk ke tahapan untuk *delay* selama 5 detik dan mengecek kembali apakah program telah menerima data dari node sensor. Jika menerima pesan maka selanjutnya mengubah pesan menjadi karakter *string* yang dapat dibaca karena data yang diterima yaitu standard utf-8. Setalah diubah maka dapat menampilkan data tersebut berasal dari node berapa, dan data-data yang dikirimkan. Setelah itu maka dapat mengirimkan data ke *database*.

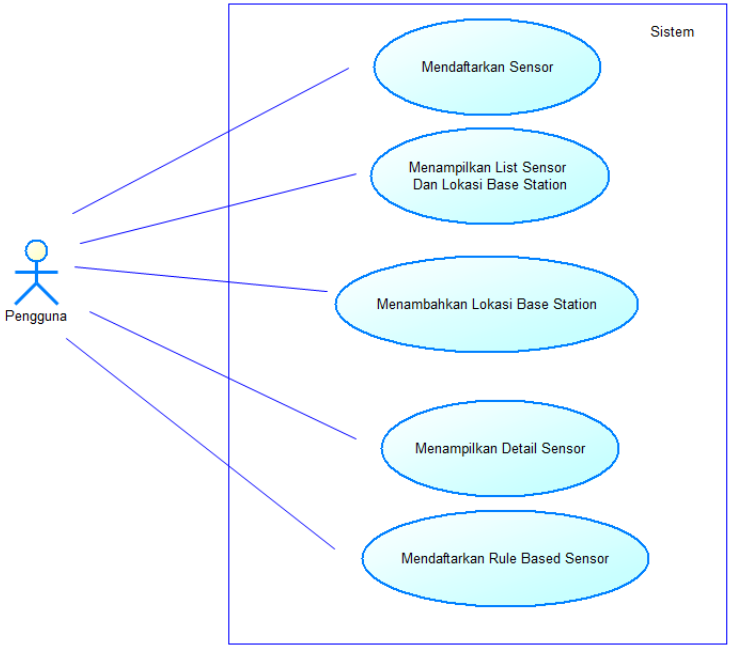
3.5 Perancangan Monitoring Sensor

Monitoring Sensor dibangun berbasis website, sistem monitoring ini berfungsi untuk memantau kondisi lingkungan berdasarkan data-data sensor yang dikumpulkan, sehingga dapat divisualisasikan. Node-node sensor yang diimplementasikan, data-data sensor yang dikumpulkan, ditampilkan melalui sistem monitoring sensor.

Penggunaan monitoring sensor ini terdapat beberapa fungsi, beberapa kegunaan antara lain, yaitu mendaftarkan sensor, menampilkan list sensor dan peta lokasi *base station*, menambahkan lokasi *base station*, menampilkan informasi detail data-data setiap sensor, dan mendaftarkan *rule based* informasi kondisi lingkungan.

Pembahasan lain mengenai monitoring sensor ini meliputi, diagram kasus penggunaan monitoring sensor, perancangan basis data, dan perancangan antarmuka pengguna.

3.5.1 Perancangan Diagram Kasus Penggunaan



Gambar 3.8 Diagram Kasus Penggunaan

Pada bagian ini dijelaskan mengenai diagram kasus penggunaan (*use case diagram*), fungsi diagram kasus penggunaan ini yaitu untuk mendeskripsikan apa saja interkasi dari seorang user atau pengguna dengan sistem yang akan dibuat. Kasus penggunaan yang akan dibuat yaitu meliputi, mendaftarkan sensor, menampilkan list sensor, menampilkan detail sensor dan mendaftarkan *rule based* sensor.

3.5.1.1 Mendaftarkan Sensor

Pada kasus penggunaan mendaftarkan sensor, pengguna mendaftarkan node sensor serta sensor-sensor yang akan diterima

datanya atau sensor-sensor yang aktif. Pengguna pertama kali memilih menu mendaftarkan sensor sehingga masuk ke menu *form* untuk mendaftarkan sensor, rincian *form* yang harus diisi antara lain adalah nama sensor, deskripsi sensor, dan pilihan sensor yang diaktifkan oleh node sensor(temperatur, kelembapan, gas mq-135, gas CO mq-7). Sensor yang didaftarkan akan ditampilkan di halaman *dashboard* utama.

3.5.1.2 Menampilkan *List* Sensor Dan Lokasi *Base Station*

Pada kasus penggunaan menampilkan *list* sensor, merupakan halaman utama dari sistem monitoring yaitu menampilkan seluruh node sensor yang didaftarkan. Kasus penggunaan ini menampilkan beberapa informasi node sensor yaitu latest update(update terakhir data sensor yang masuk ke database), data-data sensor terakhir yang masuk ke database yaitu temperatur, kelembapan, gas dari sensor mq-135, dan gas CO dari sensor mq-7. Di kasus penggunaan ini juga menampilkan lokasi dari *base station.*

3.5.1.3 Menambahkan Lokasi *Base Station*

Kasus penggunaan menambahkan lokasi *base station*, merupakan kasus penggunaan berfungsi untuk menambahkan lokasi *base station* saat ini, untuk menambahkannya, sistem secara otomatis bisa menambahkan melalui lokasi saat ini saat mengakses website monitoring sensor, ataupun raspberry pi sebagai *base station* dapat langsung menambahkan lokasinya.

3.5.1.4 Menampilkan Detail Sensor

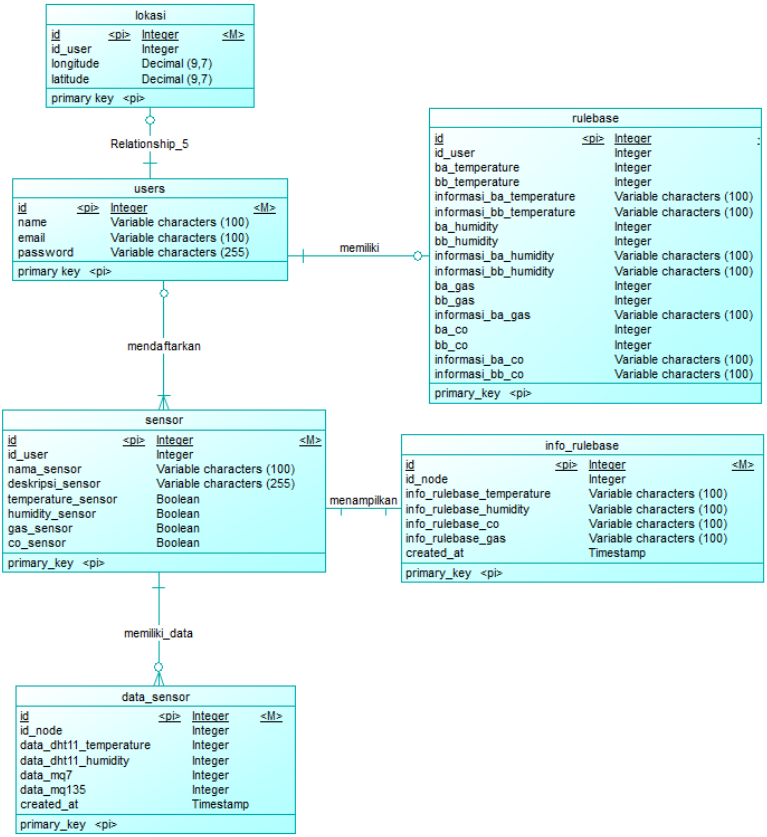
Untuk kasus penggunaan menampilkan detail sensor, merupakan informasi detail visualisasi data dari sensor-sensor yang terdaftar, yaitu pengguna memilih detail sensor di halaman utama di *card* node sensor, dan bisa memilih detail sensor sesuai node sensor yang diinginkan. Rincian yang ditampilkan yaitu nama sensor dari node sensor, deskripsi sensor, dan status sensor, selain itu terdapat grafik-grafik dari sensor temperatur, kelembapan, gas

sensor mq-135 dan gas CO sensor mq-7. Di halaman detail sensor juga terdapat informasi kondisi lingkungan sesuai *rule based* yang didaftarkan.

3.5.1.5 Mendaftarkan Rule Based Sensor

Pada kasus penggunaan mendaftarkan *rule based* sensor, pengguna dapat menetapkan suatu informasi mengenai data-data dari beberapa sensor sehingga menampilkan informasi nilai data sensor tersebut berupa informasi kondisi lingkungan. Contoh jika data sensor melebihi dari nilai yang ditetapkan dan juga mendaftarkan informasi yaitu jika suhu lebih besar dari 50 derajat, dan informasi yang ditetapkan suhu panas, maka *rule based* informasi data sensor ditampilkan ke halaman detail sensor.

3.5.2 Perancangan Basis Data

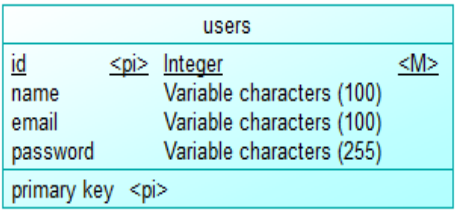


Gambar 3.9 Skema Basis Data

Perancangan basis data merupakan satu tahap dalam pembuatan aplikasi atau website yang berfungsi untuk menggambarkan struktur dan kerangkan database yang digunakan dalam pembuatan website monitoring sensor. Di gambar diatas merupakan skema database yang akan diimplementasikan. Sistem manajemen basis data yang digunakan adalah Mysql. Pada sub bab

ini dijelaskan secara detail setiap tabel dan atribut-atribut yang terdapat di tabel atau entitas tersebut.

3.5.2.1 Tabel users



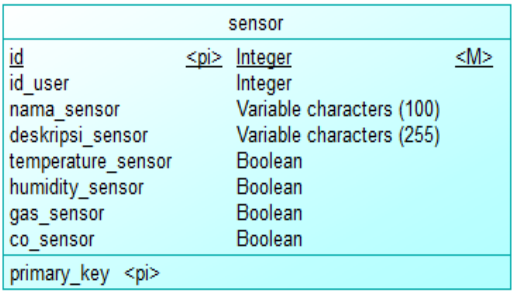
Gambar 3.10 Skema Tabel *users*

Fungsi dari tabel users adalah berguna untuk menyimpan data-data user atau pengguna agar dapat *login* di sistem monitoring server.. Data-data yang disimpan yaitu id, nama user, email user, dan *password*. Untuk detail attribut tabel *users* dapat dilihat melalui Tabel 3.5.

Tabel 3.5 Detail Tabel *users*

No	Nama Attribut	Tipe Data	Deskripsi
1.	*id*	*integer*	Merupakan *primary key* dari tabel *users*. Diatur sebagai *auto increment.*
2.	*name*	*varchar(100)*	Sebagai tanda pengenal pengguna yang didaftarkan saat registrasi.
3.	*email*	*varchar(100)*	Merupakan *email* pengguna untuk *login* di sistem.
4.	*password*	*varchar(255)*	Password digunakan pengguna agar dapat *login* ke sistem.

3.5.2.2 Tabel sensor



Gambar 3.11 Skema Tabel *sensor*

Fungsi dari tabel *sensor* adalah untuk menyimpan node sensor yang memiliki beberapa sensor yang diaktifkan atau akan dijalankan oleh node sensor. Dengan menyimpan data-data di tabel ini, maka dapat diketahui sensor mana saja yang terdaftar. Untuk detail attribut tabel *sensor* dapat dilihat melalui Tabel 3.6.

Tabel 3.6 Detail Tabel *sensor*

No	Nama Attribut	Tipe Data	Deskripsi
1.	*id*	*integer*	Merupakan *primary key* dari tabel sensor. Diatur sebagai *auto increment.*
2.	*id_user*	*integer*	Merupakan *foreign key* dari tabel *users*.
3.	*nama_sensor*	*varchar(100)*	Sebagai nama untuk pengenal untuk node sensor.
4.	*deskripsi_sensor*	*varchar(255)*	Merupakan deskripsi node sensor yang didaftarkan.
5.	*temperature_sensor*	*boolean*	Merupakan pendanda (*flag*) untuk menandakan

			apakah sensor suhu dimiliki oleh node sensor atau tidak.
6.	*humidity_sensor*	*boolean*	Merupakan pendanda (*flag*) untuk menandakan apakah sensor kelembapan dimiliki oleh node sensor atau tidak.
7.	*gas_sensor*	*boolean*	Merupakan pendanda (*flag*) untuk menandakan apakah sensor gas mq-135 dimiliki oleh node sensor atau tidak.
8.	*co_sensor*	*boolean*	Merupakan pendanda (*flag*) untuk menandakan apakah sensor gas co dimiliki oleh node sensor atau tidak.

3.5.2.3 Tabel data_sensor

data_sensor			
id	<pi>	Integer	<M>
id_node		Integer	
data_dht11_temperature		Integer	
data_dht11_humidity		Integer	
data_mq7		Integer	
data_mq135		Integer	
created_at		Timestamp	
primary_key <pi>			

Gambar 3.12 Skema Tabel *data_sensor*

Fungsi dari tabel *data_sensor* adalah berguna untuk menyimpan data-data sensor yang dikumpulkan oleh node sensor. Tabel ini berguna untuk menyimpan data-data yang dikirimkan langsung oleh *base station* yaitu raspberry pi. Untuk detail attribut tabel *data_sensor* dapat dilihat melalui Tabel 3.7.

Tabel 3.7 Detail Tabel *data_sensor*

No	Nama Attribut	Tipe Data	Deskripsi
1.	*id*	*integer*	Merupakan *primary key* dari tabel *data_sensor*. Diatur sebagai *auto increment.*
2.	*id_node*	*integer*	Merupakan *foreign key* dari tabel sensor.
3.	*data_dht11_temperature*	*integer*	Merupakan data tentang suhu yang didapatkan node sensor.
4.	*data_dht11_humidity*	*integer*	Merupakan data tentang kelembapan yang didapatkan node sensor.
5.	*data_mq7*	*integer*	Merupakan data tentang konsentrasi gas karbondioksida dari sensor mq-7 yang didapatkan node sensor.
6.	*data_mq135*	*integer*	Merupakan data tentang konsentrasi gas dari sensor mq-135 yang didapatkan node sensor..

7.	*created_at*	*timestamp*	Merupakan attribut yang menyimpan waktu dan tanggal saat data sensor masuk ke basis data.

3.5.2.4 Tabel rulebase

rulebase			
id	<pi>	Integer	
id_user		Integer	
ba_temperature		Integer	
bb_temperature		Integer	
informasi_ba_temperature		Variable characters (100)	
informasi_bb_temperature		Variable characters (100)	
ba_humidity		Integer	
bb_humidity		Integer	
informasi_ba_humidity		Variable characters (100)	
informasi_bb_humidity		Variable characters (100)	
ba_gas		Integer	
bb_gas		Integer	
informasi_ba_gas		Variable characters (100)	
ba_co		Integer	
bb_co		Integer	
informasi_ba_co		Variable characters (100)	
informasi_bb_co		Variable characters (100)	
primary_key <pi>			

Gambar 3.13 Skema Tabel *rulebase*

Fungsi dari tabel *rulebase* adalah berguna untuk menyimpan *rulebase* nilai-nilai batas tertentu setiap sensor agar menjadi acuan untuk menampilkan informasi kondisi lingkungan sesuai data sensor tersebut. Untuk detail attribut tabel *rulebase* dapat dilihat melalui Tabel 3.8.

Tabel 3.8 Detail Tabel *rulebase*

No	Nama Attribut	Tipe Data	Deskripsi
1.	*id*	*integer*	Merupakan *primary key* dari tabel *rulebase*.

			Diatur sebagai *auto increment.*
2.	*id_user*	*integer*	Merupakan *foreign key* dari tabel *user*.
3.	*ba_temperature*	*integer*	Berisi batas atas nilai suhu sebagai acuan informasi kondisi lingkungan
4.	*bb_temperature*	*integer*	Berisi batas bawah nilai suhu sebagai acuan informasi kondisi lingkungan
5.	*informasi_ba_temperature*	*varchar(100)*	Berisi informasi yang ditetapkan untuk ditampilkan mengenai data sensor suhu saat data di batas atas nilai yang ditetapkan.
6.	*informasi_bb_temperature*	*varchar(100)*	Berisi informasi yang ditetapkan untuk ditampilkan mengenai data sensor suhu saat data di batas bawah nilai yang ditetapkan.
7.	*ba_humidity*	*integer*	Berisi batas atas nilai kelembapan sebagai acuan informasi kondisi lingkungan
8.	*bb_humidity*	*integer*	Berisi batas bawah nilai kelembapan

			sebagai acuan informasi kondisi lingkungan
9.	*informasi_ba_humidity*	*varchar(100)*	Berisi informasi yang ditetapkan untuk ditampilkan mengenai data sensor kelembapan saat data di batas atas nilai yang ditetapkan.
10.	*informasi_bb_humidity*	*varchar(100)*	Berisi informasi yang ditetapkan untuk ditampilkan mengenai data sensor kelembapan saat data di batas bawah nilai yang ditetapkan.
11.	*ba_gas*	*integer*	Berisi batas atas nilai konsentrasi gas sensor mq-135 sebagai acuan informasi kondisi lingkungan
12.	*bb_gas*	*integer*	Berisi batas bawah nilai konsentrasi gas sensor mq-135 sebagai acuan informasi kondisi lingkungan
13.	*informasi_ba_gas*	*varchar(100)*	Berisi informasi yang ditetapkan untuk ditampilkan mengenai data sensor konsentrasi

			gas dari sensor mq-135 saat data di batas atas nilai yang ditetapkan.
14.	*informasi_bb_gas*	*varchar(100)*	Berisi informasi yang ditetapkan untuk ditampilkan mengenai data sensor konsentrasi gas dari sensor mq-135 saat data di batas bawah nilai yang ditetapkan.
15.	*ba_co*	*integer*	Berisi batas atas nilai konsentrasi gas karbondioksida sensor mq-7 sebagai acuan informasi kondisi lingkungan
16.	*bb_gas*	*integer*	Berisi batas bawah nilai konsentrasi gas karbondioksida sensor mq-7 sebagai acuan informasi kondisi lingkungan
17.	*informasi_ba_gas*	*varchar(100)*	Berisi informasi yang ditetapkan untuk ditampilkan mengenai data konsentrasi gas karbondioksida sensor mq-7 saat data di batas atas

			nilai yang ditetapkan.
18.	*informasi_bb_gas*	*varchar(100)*	Berisi informasi yang ditetapkan untuk ditampilkan mengenai data konsentrasi gas karbondioksida dari sensor mq-7 saat data di batas bawah nilai yang ditetapkan.

3.5.2.5 Tabel info_rulebase

info_rulebase			
id	<pi>	Integer	<M>
id_node		Integer	
info_rulebase_temperature		Variable characters (100)	
info_rulebase_humidity		Variable characters (100)	
info_rulebase_co		Variable characters (100)	
info_rulebase_gas		Variable characters (100)	
created_at		Timestamp	
primary_key <pi>			

Gambar 3.14 Skema Tabel *info_rulebase*

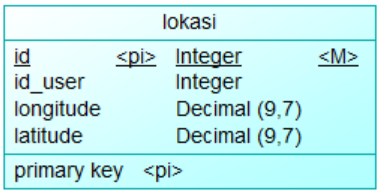
Fungsi dari tabel *info_rulebase* adalah berguna untuk menyimpan hasil dari menjalankan sebuah prosedur di *database*, yaitu mengecek data di tabel *data_sensor* dengan kondisi *rulebase* yang di simpan di tabel *rulebase*, sehingga hasil dari prosedur itu di masukkan ke tabel *info_rulebase* untuk ditampilkan di sistem monitoring sensor. Untuk detail attribut tabel *info_rulebase* dapat dilihat melalui Tabel 3.9.

Tabel 3.9 Detail Tabel *info_rulebase*

No	Nama Attribut	Tipe Data	Deskripsi
1.	*id*	*integer*	Merupakan *primary key* dari tabel *info_rulebase*. Diatur sebagai *auto increment.*
2.	*id_node*	*integer*	Merupakan id_node (inputan prosedur) saat prosedur dijalankan.
3.	*Info_rulebase_temperature*	*varchar(100)*	Merupakan informasi kondisi lingkungan mengenai suhu lingkungan yang dihasilkan dari prosedur mengupdate info rulebase
4.	*Info_rulebase_humidity*	*varchar(100)*	Merupakan informasi kondisi lingkungan mengenai kelembapan lingkungan yang dihasilkan dari prosedur mengupdate info rulebase
5.	*Info_rulebase_gas*	*varchar(100)*	Merupakan informasi kondisi lingkungan mengenai

			konsentrasi gas dari sensor mq-135 di lingkungan yang dihasilkan dari prosedur mengupdate info rulebase.
6.	*Info_rulebase_co*	*varchar(100)*	Merupakan informasi kondisi lingkungan mengenai konsentrasi gas karbondioksida dari sensor mq-7 di lingkungan yang dihasilkan dari prosedur mengupdate info rulebase
7.	*created_at*	*timestamp*	Merupakan attribut yang menyimpan waktu dan tanggal saat data yang dihasilkan prosedur mengupdate info rulebase masuk ke basis data.

3.5.2.6 Tabel lokasi



Gambar 3.15 Skema Tabel *lokasi*

Fungsi dari tabel lokasi adalah berguna untuk menyimpan lokasi dari *base station*, isi dari tabel ini berupa id sebagai id lokasi dan bertindak sebagai *primary key*, terdapat id_user yang merupakan *foreign key* dari tabel users dan terdapat juga *longitude* dan *latitude* dari lokasi yang disimpan. Berikut Tabel 3.10 merupakan penjelasan dari tabel lokasi.

Tabel 3.10 Detail Tabel *lokasi*

No	Nama Attribut	Tipe Data	Deskripsi
1.	*id*	*integer*	Merupakan *primary key* dari tabel *lokasi*. Diatur sebagai *auto increment.*
2.	*id_user*	*integer*	Merupakan *foreign key* dari tabel *users*.
3.	*longitude*	*decimal(100)*	Berisi data *longitude* dari lokasi yang ditambahkan.
4.	*latitude*	*decimal(9,7)*	Berisi data *latitude* dari lokasi yang ditambahkan.

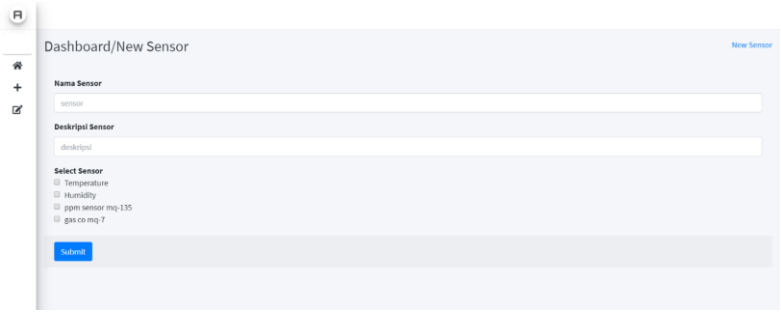
3.5.3 Perancangan Antarmuka Pengguna

Perancangan antar muka pengguna merupakan suatu langkah dalam membuat website atau aplikasi yang sesuai dengan kebutuhan pengguna. Perancangan antarmuka ini berfungsi sebaga gambaran umum website monitoring server direpresentasikan atau ditampilkan sehingga menjadi rancangan antar muka yang sesuai dengan tujuan dan kebutuhan.

Website monitoring server yang dibangun, akan dibangun beberapa halaman/tampilan yaitu halaman form mendaftarkan sensor, halaman menampilkan list sensor, halaman menampilkan detail sensor dan halaman untuk mendaftarkan *rule based* sensor.

3.5.3.1 Halaman Mendaftarkan Sensor

Halaman ini merupakan halaman yang berfungsi untuk mendaftarkan sensor yang akan digunakan menjadi node sensor, halaman ini berupa *form* isian, yang berupa nama sensor, deskripsi sensor serta sensor-sensor yang akan didaftarkan. Halaman mendaftarkan sensor dapat dilihat Gambar 3.16.

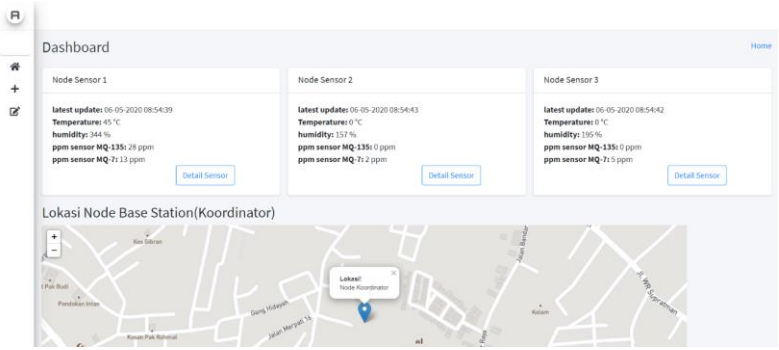


Gambar 3.16 Halaman Mendaftarkan Sensor

3.5.3.2 Halaman Menampilkan *List* Sensor Dan Lokasi *Base Station*

Halaman menampilkan list sensor, merupakan halaman utama dari website monitoring sensor ini. Pada halaman ini menampilkan list node sensor yang telah didaftarkan pada halaman

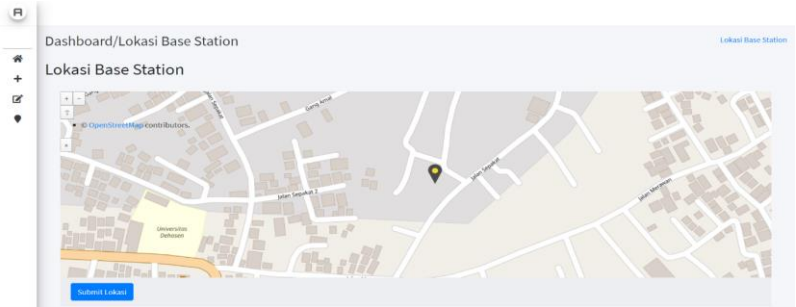
sebelummya yaitu halaman mendaftarkan sensor 3.5.3.1. Halaman ini menampilkan beberapa informasi yaitu *latest update* data sensor yang masuk, nilai-nilai dari beberapa sensor. Di halaman ini juga menampilkan lokasi node *base station*. Dan juga terdapat *button* untuk dapat menuju ke halaman detail sensor yang akan dijelaskan lebih rinci berikutnya. Gambar 3.17 merupakan halaman saat menampilkan *list* sensor.



Gambar 3.17 Halaman Menampilkan *List* Sensor

3.5.3.3 Halaman Menambahkan Lokasi *Base Station*

Halaman menambahkan lokasi base station, berfungsi untuk menambahkan lokasi dari base station, lokasi ini berdasarkan lokasi base station saat sedang diakses. Halaman ini berupa halaman yang menampilkan peta dari lokasi saat itu diakses. Untuk lebih jelasnya gambar menampilkan detail sensor dapat dilihat di Gambar 3.18.



Gambar 3.18 Halaman Menambahkan Lokasi *Base Station*

3.5.3.4 Halaman Menampilkan Detail Sensor

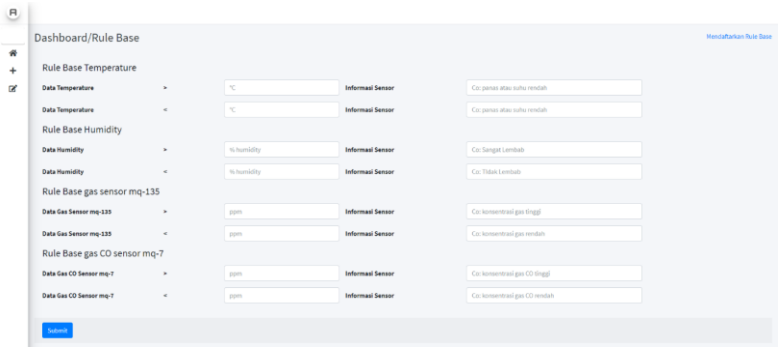
Halaman menampilkan detail sensor, berfungsi untuk menampilkan detail sensor yaitu ada beberapa informasi yaitu informasi node sensor dengan sensor-sensor yang aktif, informasi detail *rule based* yang didaftarkan dan juga grafik-grafik dari nilai-nilai sensor yang terdapat di *database*. Untuk lebih jelasnya gambar menampilkan detail sensor dapat dilihat di Gambar 3.19.

Gambar 3.19 Halaman Menampilkan Detail Sensor

3.5.3.5 Halaman Mendaftarkan Rule Based

Pada halaman ini yaitu berfungsi untuk mendaftarkan *rule based* sensor, yaitu nilai batas dari nilai-nilai sensor sehingga kita ingin mendapatkan informasi dari sensor tersebut. Isian form rule

based sensor ini berupa batas bawah nilai sensor, batas bawah nilai sensor, informasi sensor jika berada di batas atas sensor, dan informasi sensor jika berada di batas bawah sensor. Untuk lebih jelasnya dapat dilihat melalui Gambar 3.20.



Gambar 3.20 Halaman Mendaftarkan *Rule Based* Sensor

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Bab ini membahas mengenai implementasi sistem yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya Implementasi yang dijelaskan pada bab ini meliputi lingkungan implementasi, implementasi perancangan rangkaian, *pseudocode* program, implementasi antarmuka sistem, dan sebagainya.

4.1 Lingkungan Implementasi

Lingkungan implementasi merupakan suatu lingkungan dimana sistem akan bangun. Pada lingkungan implementasi ini akan dijabarkan menjadi 2 yaitu, lingkungan implementasi perangkat keras dan lingkungan implementasi perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Pada bagian ini dijelaskan perangkat keras yang digunakan untuk membangun sistem. Lingkungan implementasi perangkat keras yang akan dibangun secara lebih rinci dijelaskan pada Gambar 4.1 dibawah ini.

Tabel 4.1 Tabel Lingkungan Implementasi Perangkat Keras

Perangkat	**Detail**
Perangkat Mikrokontroler	**Mikrokontroler:** • Atmega 328 **Model:** • Arduino UNO R3 **Tegangan:** • 5 – 12 V **Memory Flash:** • 32 KB **SRAM:** • 2KB

	Sensor: • DHT11(suhu dan kelembapan), MQ-7(gas karbondioksida), MQ-135(gas kualitas udara).
Perangkat *Base Station*(Raspberry pi)	**Model:** • Raspberry pi 3 B+ **Manufaktur:** • Raspberry pi foundation **Prosessor:** • Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz **Tegangan:** • 5V/2,5A **Koneksi:** • Wireless LAN, Bluetooth 4.2 **Interface:** • GPIO **Sistem Operasi:** • NOOBS(New Out Of Box Software)
Perangkat *Wireless Transceiver*	**Model:** • nRF24L01+SingleChip2.4GHz Transceiver **Manufaktur:** • Nordic Semiconductor **Tegangan:** • 1.9-3.6V **Frekuensi:** • 2.4GHzISMBand **Interface:** • SPI

Server Cloud	**Model:** • Virtual Private Server **Manufaktur:** • Digital Ocean **Disk Size:** • 25 GB **Core:** • 1 VCpu **Ram:** • 1 GB **Sistem Operasi:** • Ubuntu 18.04.3

4.1.2 Lingkungan Implementasi Perangkat Lunak

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 di jelaskan pada Tabel 4.2 di bawah ini.

Tabel 4.2 Tabel Lingkungan Implemantasi Perangkat Lunak

Perangkat Lunak	Detail
Arduino IDE	Arduino IDE adalah sebuah IDE yang di gunakan untuk melakukan kompilasi terhadapa kode program Bahasa C untuk mikrokontroler Arduino serta digunakan juga untuk mengupload program ke dalam Arduino. Pada Arduino IDE juga bisa melihat serial monitoring yang berfungsi untuk melihat hasil dari

	program yang berjalan pada arduino
Visual Studio Code	Visual studio code merupakan text editor yang digunkan untuk membuat program monitoring sensor berbasis website dengan *framework* laravel
Basis Data Mysql	MySQL adalah sebuah *database management system* (manajemen basis data) menggunakan perintah dasar *SQL (Structured Query Language)* bersifat *open source*, sehingga dapat digunakan secara gratis. Mysql digunakan untuk menyimpan data-data sensor.
Web Framework Laravel	Laravel merupakan framework berbasis php untuk membangun sebuah website, dengan menggunakan laravel membangun sebuah website akan berjalan sangat efektif dan efisien, laravel memiliki arsitektur sistem yaitu MVC(Model,View,Controller). Laravel digunakan sebagain framework untuk membangun website monitoring sensor yaitu layanan dashboard._
Web Server NGINX	NGINX merupakan webserver yang berfungsi sebagai tempat request http atau proxy diteruma dan diteruskan, Webserver ini bersifat open

	source, sehingga dapat bebas diakses. Webserver nginx berperan untuk tempat request masuk dan mengarahkan website monitoring sensor ketika user mengakses sistem tersebut.
nRF24 Library di Arduino	Library yang digunakan agar Arduino dapat berkomunikasi dengan modul nRF24L01. Library ini sudah mencakup fungsi-fungsi yang dibutuhkan seperti pengiriman data, pembacaan transmisi, ACK payloads, dan pengaturan kekuatan transmisi.
Thonny, Python IDE	Thonny python IDE merupakan text editor sederhana khusus menjalankan program python, untuk membuat program python di raspberry pi.

4.2 Implementasi Node Sensor

Pada bagian ini akan dijelaskan implementasi mengenai node sensor yang telah diimplementasikan melalui perancangan node sensor yang di jelaskan di bab 3.3. Implementasi node sensor meliputi implementasi rangkaian utama dan implementasi fungsi node sensor yaitu program yang dibuat.

4.2.1 Implementasi Rangkaian Utama

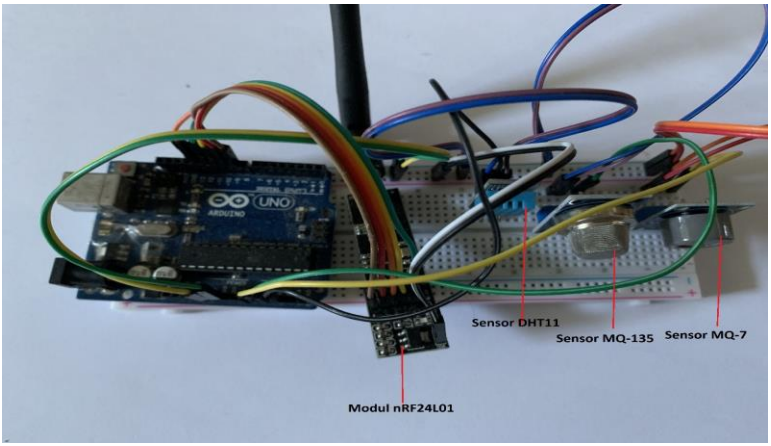
Node sensor diimplementasikan menggunakan mikrokontroller arduino uno rev 3. Mikrokontroller arduino dirancang dan dihubungkan dengan komponen-komponen lain sehingga terbentuak sebuah alat yaitu node sensor yang dapat mengumpulkan data dan mengirim data melalui jaringan nirkabel.

Tabel 4.1 Berikut merupakan komponen-komponen yang digunakan untuk membangun node sensor.

Tabel 4.1 Komponen Rangkaian Implementasi Node Sensor

Perangkat Keras		**Jumlah**
Mikrokontroller	Arduino Uno Rev 3	3 buah
Sensor	DHT11, MQ-7, MQ-135	Setiap set masing-masing 3 buah
Kabel	Kabel USB *type a to type b*	3 buah
	Kabel *jumper*	Setiap set masing-masing 17-20 buah
Board	*Breadboard*	3 buah
Transceiver	Modul nRF24L01	3 buah

Komponen yang digunakan yaitu modul nRF24L01 berfungsi sebagai media transmisi untuk mengirim data via frekuensi gelombang radio ke *base station*. Dan juga terdapat modul-modul sensor untuk mendapatkan data dari kondisi lingkungan yaitu suhu, kelembapan, gas karbondioksida, dan gas-gas yang menyebabkan kualitas udara kurang baik. Gambar 4.1 merupakan implementasi dari rangkaian node sensor yang dibangun.



Gambar 4.1 Implementasi Rangkaian Utama Node Sensor

4.2.2 Implementasi Fungsi Node Sensor

Pada bagian ini akan dijelaskan mengenai implementasi program yang diimplementasikan, implementasi program node sensor sendiri menggunakan bahasa pemrograman arduino yang berbasis dari bahasa C/C++.

Bahasa pemrograman arduino sendiri merupakan kontrol dari setiap arduino yang kita jalankan, bisa dikatakan merupakan otak dan perintah dijalankan di dalam program itu sendiri ketika program tersebut telah di*compile* dan selanjutnya di*upload* ke arduino. Pemrograman arduino itu sendiri terbagi menjadi 2 bagian penting yaitu fungsi *setup* dan fungsi *loop.*

4.2.2.1 Fungsi *setup*

Fungsi *setup* sendiri berjalan ketika arduino pertama kali dihidupkan. Didalam fungsi ini terdapat beberapa penting yaitu dengan mengaktifkan baud_rate untuk serial.begin dan menginisialisasi modul nRF24L01 dengan fungsi radio.begin(). Dan juga terdapat sensor dht yang harus diinisialisasi terlebih dahulu dengan fungsi dht.begin(). Untuk lebih jelasnya terdapat di kode sumber berikut.

```
initialize all sensor pin
initialize all sensor variable
initialize RF24 Radio(9,10)

FUNCTION setup()
   open serial.begin(9600)
   open dht.begin()
   open radio.begin()
   open radio.openWritingPipe(pipe_address)
ENDFUNCTION
```

Kode Sumber 4.1 *Pseudocode* Fungsi *setup*

4.2.2.2 Fungsi *loop*

Untuk fungsi *loop* akan berjalan ketika arduino masih terhubung dengan sumber daya sampai arduino tidak hidup atau tidak mendapatkan daya lagi. Di dalam fungsi *loop* sendiri ada beberapa fungsi-fungsi yang dipanggil yang berguna untuk mendapatkan data-data sensor. Untuk lebih jelasnya bisa dilihat melalui kode sumber berikut.

```
initialize all sensor pin
initialize all sensor variable

FUNCTION loop()
   set send_data ← “id_node|”
   set SensorTemperature ← call
       getTempSensorDHT11()
   set SensorHumidity ← call
        getHumiditySensorDHT11()
   set SensorValueMq7 ← call getSensorMq7()
   set SensorValueMq135 ← call getSensorMq135()
   itoa(SensorTemperature,SaveValueTemperature,10)
   itoa(SensorHumidity,SaveValueHumidity,10)
   itoa(SensorValueMq7,SaveValueMq7,10)
   itoa(SensorValueMq135,SaveValueMq135,10)
   strcat(send_data,SaveValueMq7)
   strcat(send_data,”|”)
   strcat(send_data,SaveValueTemperature)
   strcat(send_data,”|”)
```

```
18     strcat(send_data,SaveValueHumidity)
19     strcat(send_data,”|”)
20     strcat(send_data,SaveValueMq135)
21     radio.write(&send_data,sizeof(send_data))
22     delay(5000)
23 ENDFUNCTION
```

Kode Sumber 4.2 *Pseudocode* Fungsi *loop*

Pada fungsi *loop* sendiri sebelum mengirim data-data sensor ke *base station*, maka harus disesuaikan susunan format, data mula-mula diambil dari sensor dan diubah ke *array of char* dengan fungsi itoa(), setelah itu barulah di gabungkan dengan *string* dari variabel *send_data*, untuk formatnya yaitu *“id_node|ValueMq7|ValueTempDHT11|ValueHumidityDHT11|ValueMq135”*. Setelah terbentuk format tersebut barulah dikirim datanya dengan radio.write() yang akan mengirimkan datanya ke *base station*.

4.2.2.3 Fungsi *getTempSensorDHT11*

Fungsi berikut merupakan fungsi untuk mendapatkan data sensor yang berkaitan dengan nilai suhu suatu lingkungan

```
1  initialize all sensor variable
2
3  FUNCTION getTempSensorDHT11()
4      set t_celcius ← dht.readTemperature()
5      return t_celcius
6  ENDFUNCTION
```

Kode Sumber 4.3 *Pseudocode* Fungsi *getTempSensorDHT11*

4.2.2.4 Fungsi *getHumiditySensorDHT11*

Fungsi berikut merupakan fungsi untuk mendapatkan data sensor yang berkaitan dengan nilai kelembapan udara, nilai ini didapatkan dari *sensing* oleh sensor DHT11.

```
1  initialize all sensor variable
2
3  FUNCTION getHumiditySensorDHT11()
```

```
    set humidity ← dht.readHumidity()
    return humidity
ENDFUNCTION
```

Kode Sumber 4.4 *Pseudocode* Fungsi *getHumiditySensorDHT11*

4.2.2.5 Fungsi *getSensorMq7*

Fungsi berikut merupakan fungsi untuk mendapatkan data sensor yang berkaitan dengan nilai konsentrasi gas karbondioksida, nilai ini didapatkan dari *sensing* oleh sensor MQ-7.

```
initialize all sensor variable

FUNCTION getSensorMq7()
    set sensorvalue ← analogread(pinSensor)
    set VRL ← sensorvalue*(5/1024)
    set Rs ← (5 * RL/VRL) - RL
    set ppm ← 100 * pow(Rs/Ro,-1.53)
    return ppm
ENDFUNCTION
```

Kode Sumber 4.5 *Pseudocode* Fungsi *getSensorMq7*

4.2.2.6 Fungsi *getSensorMq135*

Fungsi berikut merupakan fungsi untuk mendapatkan data sensor yang berkaitan dengan nilai konsentrasi gas-gas yang mempengaruhi kualitas udara. Nilai ini didapatkan dari *sensing* oleh sensor MQ-135.

```
initialize all sensor variable

FUNCTION getSensorMq135()
    set sensorvalue ← analogRead(pinSensor)
    return sensorvalue
ENDFUNCTION
```

Kode Sumber 4.6 *Pseudocode* Fungsi *getSensorMq135*

4.3 Implementasi *Base Station*

Pada bagian ini akan dijelaskan implementasi mengenai *base station* yang telah diimplementasikan melalui perancangan *base station* yang di jelaskan di bab 3.4. Implementasi node sensor meliputi implementasi rangkaian utama dan implementasi fungsi node sensor yaitu program yang dibuat.

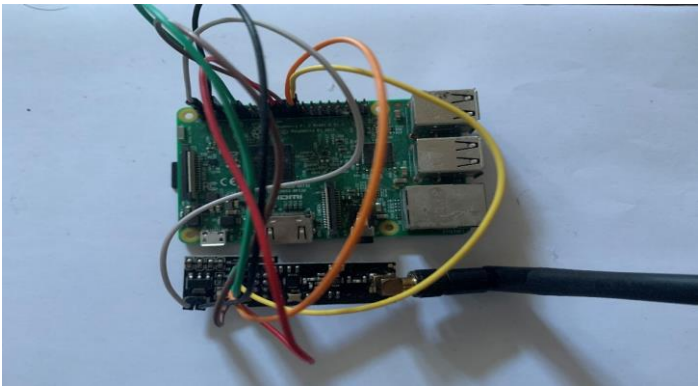
4.3.1 Implementasi Rangkaian Utama

Pada base station, diimplementasikan menggunakan raspberry pi 3 b+, *base station* ini berfungsi sebagai media perantara komunikasi antara node sensor dan *database*, yaitu mendapatkan data node sensor dan mengirimkan data ke *database* melalui frekuensi gelombang radio menggunakan modul nRF24L01 sebagai modul komunikasinya. Untuk implementasi *base station* ada beberapa komponen-komponen yang dihubungkan sehingga menjadi suatu alat yang dapat menerima data dan mengirimkan data ke server. Berikut tabel komponen alat dari *base station*

Tabel 4.2 Komponen Rangkaian Implementasi *Base Station*

Perangkat Keras		**Jumlah**
Mikrokontroller	Rasppberry Pi 3 B+	1 buah
Kabel	Kabel *jumper*	7 buah
Transceiver	Modul nRF24L01	1 buah

Gambar 4.2 merupakan implementasi dari rangkaian node sensor yang dibangun setelah beberapa komponen-komponen tersebut dihubungkan dengan kabel *jumper*.



Gambar 4.2 Implementasi Rangkaian Utama *Base Station*

4.3.2 Implementasi Fungsi *Base Station*

Implementasi fungsi pada *base station* yaitu menjelaskan implementasi program yang telah dibuat dengan menjelaskannya menggunakan *pseudocode*, implementasi *base station* menggunakan program python untuk mendapatkan data yang dikirimkan oleh node sensor, dan mengirimkan datanya ke *database*.

4.3.2.1 Fungsi *main*

Fungsi main merupakan fungsi utama dari program python yang dijalankan oleh raspberry pi, yaitu mula-mula program menginisialisasi variabel untuk menghubungkan modul nRF24L01 dengan GPIO raspberry pi, setelah itu menjalankan program secara terus menerus sehingga dapat menerima data, untuk mengetahui apakah data masuk maka membuat kondisi jika nRF24L01 mendapatkan data dengan fungsi *radio.available()* setelah itu maka dilakukanlah proses untuk mengkonversi data yang didapat sehingga menjadi bentuk *standard unicode* yaitu string yang biasa kita jumpai, setelah itu karena data yang didapat masih berupa list dengan tipe data yaitu string maka kita perlu mengubahnya dulu ke bentuk integer agar dapat dimasukan ke database sesuai tipe data yang diset oleh *database*, untuk mengirimkan ke *database* dibuat

lagi fungsi *insert_db* untuk mengirimkan data yang diterima ke database server.

```
initialize variable pipe_address
radio.begin(0,17)
radio.openReadingPipe(1,pipes[0])
radio.openReadingPipe(2,pipes[1])
radio.openReadingPipe(3,pipes[2])

FUNCTION main()
    WHILE(True)
       WHILE(not radio.available(0))
             print("Belum ada data yang masuk")
             time.sleep(5)
       ENDWHILE
       IF radio.available() THEN
             radio.read(receivedMessage,
                 radio.getDynamicPayloadSize())
             FOR n in receivedMessage
                   IF n >= 32 and n<= 126
                       set string ← string + chr(n)
                   END IF
             END FOR
             set data = string.split("|")
             set datetime = time.strftime()
             set data = list(map(int,data))
             data.append(datetime)
             IF data[0] == 1 THEN
                   print("Data dari node sensor 1")
             END IF
             IF data[0] == 2 THEN
                   print("Data dari node sensor 2")
             END IF
             IF data[0] == 3 THEN
                   print("Data dari node sensor 3")
             END IF
             call insert_db(data)
       END IF
    END WHILE
ENDFUNCTION
```

Kode Sumber 4.7 *Pseudocode* Fungsi *main*

4.3.2.2 Fungsi *insert_db*

Fungsi *insert_db* merupakan fungsi berguna untuk mengirimkan data ke *database* server. Pada fungsi ini terdapat parameter data yaitu data yang dikirimkan dari node sensor, fungsi ini memanfaatkan *library* python yaitu *mysql.connector* agar bisa mengirimkan ke *database*.

```
import mysql.connector

FUNCTION insert_db(data)
    set mydb ← mysql.connector.connect(
         host ← “IP_HOST”
         user ← “USER_HOST”
         passwd ← “PASSWORD_HOST”
         database ← “DATABASE_HOST” )
    set mycursor ← mydb.cursor()
    set  sql ← "INSERT INTO data_sensor(column)
                VALUES(value)”
    set val ← data
    mycursor.execute(sql,val)
    mydb.commit()
ENDFUNCTION
```

Kode Sumber 4.8 *Pseudocode* Fungsi *insert_db*

4.4 Implementasi Monitoring Sensor

Implementasi website monitoring sensor menggunakan *framework* php yaitu laravel, untuk tampilannya menggunakan bootstrap sehingga dapat mempermudah untuk membuat tampilan-tampilan website monitoring sensor tersebut. Pada bagian ini juga dijelaskan mengenai *pseudocode* program dari kasus penggunaan, serta implementasi rancangan antarmuka.

4.4.1 Implementasi Kasus Penggunaan

Pada bagian ini akan dijelaskan mengenai implementasi dari kasus penggunaan yang sudah diimplementasikan. Berdasarkan dari perancangan diagram kasus penggunaan pada bab 3.5.1 ada beberapa kasus penggunaan yaitu mendaftarkan sensor, menampilkan list sensor dan lokasi *base station*, menambahkan

lokasi *base station*, menampilkan detail *list* sensor, mendaftarkan *rule based* sensor. Di implementasi kasus penggunaan ini dijelaskan mengenai *pseudocode* dari program ataupun fungsi-fungsi dari *controller* website monitoring sensor.

4.4.1.1 Mendaftarkan Sensor

Pada Implementasi kasus penggunaan mendaftarkan sensor merupakan implementasi dari perancangan kasus penggunaan pada bab 3.5.1.1. Fungsi yang terdapat disini adalah fungsi *store*.

Fungsi *store* di kasus penggunaan ini berguna untuk menambahkan data-data sensor mana saja yang akan digunakan untuk di sebuah node sensor. Fungsi ini menerima inputan data dari sebuah *form*, data-data tersebut berisi nama sensor, deskripsi sensor, dan *flag true/false* dari sensor *temperature*, *humidity*, gas (kualitas udara mq 135), gas co (konsentrasi gas co sensor mq7). Fungsi ini menyimpan data-data tersebut ke tabel sensor. Berikut kode sumber 4.9 dari kasus penggunaan mendaftarkan sensor.

```
1   FUNCTION store(request)
2       set input ← request->all()
3       set id_user ← Auth::user()->id
4       set sensor[] ← [ ‘id_user’ => id_user,
.           ‘nama_sensor’ => input[‘nama_sensor’],
.           ‘deskripsi_sensor’ =>
.                input[‘deskripsi_sensor’],
.           ‘temperature_sensor’ =>
.                input[‘temperature’],
.           ‘himidity_sensor’ => input[‘humidity’],
.           ‘gas_sensor’ => input[‘gas’],
.           ‘co_sensor’ => input[‘co’]]
5       call model->Sensor::insert(sensor)
6       load redirect(‘/home’)
7   ENDFUNCTION
```

Kode Sumber 4.9 *Pseudocode* Fungsi *store*

4.4.1.2 Menampilkan *List* Sensor Dan Lokasi *Base Station*

Pada Implementasi kasus penggunaan menampilkan *list* sensor dan lokasi *Base Station* merupakan implementasi dari perancangan kasus penggunaan pada bab 3.5.1.2. Fungsi yang terdapat disini adalah fungsi *view*.

Fungsi *view* di kasus penggunaan ini berguna untuk menampilkan data-data di halaman utama/halaman menampilkan *list* sensor, data-data yang ditampilkan berupa status sensor, lokasi *base station*, dan informasi data terakhir yang masuk yaitu data sensor dan *last update* data yang masuk. Berikut kode sumber 4.10 dari kasus penggunaan menampilkan *list* sensor dan lokasi *base station*.

```
1   FUNCTION view()
2       set id_user ← Auth::user()->id
3       set data[‘status’] ← DB::table(‘sensor’)
.             ->where(‘id_user’,id_user)->get()
4       set data[‘lokasi’] ← DB::table(‘lokasi’)
.             ->where(‘id_user’,id_user)->get()
5       set data[‘sensor’] ← DB::select(DB::raw(“
.            (SELECT * FROM data_sensor WHERE id_node = 1
.            ORDER BY created_at DESC LIMIT 1 ) UNION
.            (SELECT * FROM data_sensor WHERE id_node = 2
.            ORDER BY created_at DESC LIMIT 1 ) UNION
.            (SELECT * FROM data_sensor WHERE id_node = 3
.            ORDER BY created_at DESC LIMIT 1 ) ”))
6       return view(‘dashboard.view’, data)
7   ENDFUNCTION
```

Kode Sumber 4.10 *Pseudocode* Fungsi *view*

4.4.1.3 Menambahkan Lokasi *Base Station*

Pada Implementasi kasus penggunaan menambahkan lokasi *base station* merupakan implementasi dari perancangan kasus penggunaan pada bab 3.5.1.3. Fungsi yang terdapat disini adalah fungsi *add_lokasi*.

Fungsi *add_lokasi* di kasus penggunaan ini berguna untuk menambahkan lokasi dari *base station* yang berupa data *longitude*

dan *latitude* lokasi . Berikut kode sumber 4.11 dari kasus penggunaan menambahkan lokasi *base station.*

```
1   FUNCTION add_lokasi()
2       set input ← request->all()
3       set id_user ← Auth::user()->id
4       set lokasi_data[] ← [ 'id_user' => id_user,
.             'longitude' => input['longitude'],
.             'latitude' => input['latitude']]
5       set cek_lokasi ← DB::table('lokasi')
.               ->where('lokasi',lokasi)->first()
6        IF cek_lokasi THEN
7           call model->Lokasi::where('id_user',id_user)
.                     ->update(['longitude'=>
.                     input['longitude'],
.                    ['latitude'=> input['latitude']])
8        ELSE THEN
9              call Lokasi::insert(lokasi_data)
10       END IF
11      load redirect(/home)
12  ENDFUNCTION
```

Kode Sumber 4.11 *Pseudocode* Fungsi *add_lokasi*

4.4.1.4 Menampilkan Detail *List* Sensor

Pada Implementasi kasus penggunaan menampilkan detail *list* sensor merupakan implementasi dari perancangan kasus penggunaan pada bab 3.5.1.4. Fungsi yang terdapat disini adalah fungsi *view_detail, get_temperature, get_humidity, get_mq7, get_mq135.*

Fungsi *view_detail* di kasus penggunaan ini berguna untuk menampilkan detail informasi data-data sensor dari node sensor yang dipilih, data-data itu berupa data sensor yang ditampilkan melalui grafik, status sensor, dan informasi *rule based* sensor. Berikut kode sumber 4.12 fungsi *view_detail.*

```
1   FUNCTION view_detail(id)
2       set id_user ← Auth::user()->id
3       set data['id_node'] ← id
4       set data['status'] ← DB::table('sensor')
```

```
.            ->where(‘id_user’,id_user)
.            ->where(‘id’,id)->get()
5        set data[‘rulebase’] ←
.            DB::table(‘info_rulebase’)
.            ->where(‘id_node’,id)
.            ->orderBy(‘id’,’desc’)->first()
6        return  view(‘admin.detail_sensor’,data)
7    ENDFUNCTION
```

Kode Sumber 4.12 *Pseudocode* Fungsi *view_detail*

Selanjutnya ada beberapa fungsi yang berguna untuk menampilkan data-data melalui grafik, karena data yang diinginkan ditampilkan secara *real time* ketika data masuk, maka fungsi-fungsi ini dipanggil oleh *ajax* yang berfungsi memanggil request-request beberapa fungsi berikut.

Berikut merupakan kode sumber 4.13 fungsi *get_temperature*. Pada fungsi ini berguna untuk mendapatkan data-data sensor dari sensor dht11 yaitu nilai *temperature*, data-data yang telah didapatkan berupa labelnya yaitu *timestamps* data saat masuk ke basis data, dan juga data nilai *temperature* akan dikembalikan dalam bentuk *json*, sehingga dapat ditampilkan pada grafik.

```
1    FUNCTION get_temperature(id)
2        set data_sensor ← DB::table(‘data_sensor’)
.            ->where(‘id_node’,id)
.            ->limit(30)->orderBy(‘created_at’,’desc’)
.            ->get()->reverse()
3        set labels ← data_sensor->pluck(‘created_at’)
4        set data ← data_sensor
.            ->pluck(‘data_dht11_temperature’)
5        return  response()
.            ->json(compact(‘labels’,’data’))
6    ENDFUNCTION
```

Kode Sumber 4.13 *Pseudocode* Fungsi *get_temperature*

Berikut merupakan kode sumber 4.14 fungsi *get_humidity*. Pada fungsi ini berguna untuk mendapatkan data-data sensor dari

sensor dht11 yaitu nilai *humidity* atau kelembapan, data-data yang telah didapatkan berupa labelnya yaitu *timestamps* data saat masuk ke basis data, dan juga data nilai *humidity* akan dikembalikan dalam bentuk *json*, sehingga dapat ditampilkan pada grafik.

```
1   FUNCTION get_humidity(id)
2       set data_sensor ← DB::table(‘data_sensor’)
.             ->where(‘id_node’,id)
.             ->limit(30)->orderBy(‘created_at’,’desc’)
.             ->get()->reverse()
3       set labels ← data_sensor->pluck(‘created_at’)
4       set data ← data_sensor
.             ->pluck(‘data_dht11_humidity’)
5       return  response()
.             ->json(compact(‘labels’,’data’))
6   ENDFUNCTION
```

Kode Sumber 4.14 *Pseudocode* Fungsi *get_humidity*

Berikut merupakan kode sumber 4.15 fungsi *get_mq7*. Pada fungsi ini berguna untuk mendapatkan data-data sensor dari sensor mq7 yaitu nilai konsentrasi gas karbondioksida, data-data yang telah didapatkan berupa labelnya yaitu *timestamps* data saat masuk ke basis data, dan juga data nilai konsentrasi gas karbondioksida dari sensor mq7 akan dikembalikan dalam bentuk *json*, sehingga dapat ditampilkan pada grafik.

```
1   FUNCTION get_mq7(id)
2       set data_sensor ← DB::table(‘data_sensor’)
.             ->where(‘id_node’,id)
.             ->limit(30)->orderBy(‘created_at’,’desc’)
.             ->get()->reverse()
3       set labels ← data_sensor->pluck(‘created_at’)
4       set data ← data_sensor
.             ->pluck(‘data_mq7)
5       return  response()
.             ->json(compact(‘labels’,’data’))
6   ENDFUNCTION
```

Kode Sumber 4.15 *Pseudocode* Fungsi *get_mq7*

Berikut merupakan kode sumber 4.16 fungsi *get_mq135*. Pada fungsi ini berguna untuk mendapatkan data-data sensor dari sensor mq135 yaitu nilai konsentrasi gas kualitas udara ataupun nilai ppm dari kualitas udara, data-data yang telah didapatkan berupa labelnya yaitu *timestamps* data saat masuk ke basis data, dan juga data nilai kualitas udara(ppm) dari sensor mq135 akan dikembalikan dalam bentuk *json*, sehingga dapat ditampilkan pada grafik.

```
1   FUNCTION get_mq135(id)
2       set data_sensor ← DB::table(‘data_sensor’)
.             ->where(‘id_node’,id)
.             ->limit(30)->orderBy(‘created_at’,’desc’)
.             ->get()->reverse()
3       set labels ← data_sensor->pluck(‘created_at’)
4       set data ← data_sensor
.             ->pluck(‘data_mq135)
5       return  response()
.             ->json(compact(‘labels’,’data’))
6   ENDFUNCTION
```

Kode Sumber 4.16 *Pseudocode* Fungsi *get_mq135*

4.4.1.5 Mendaftarkan *Rule Based* Sensor

Pada Implementasi kasus penggunaan mendaftarkan *rule based* sensor merupakan implementasi dari perancangan kasus penggunaan pada bab 3.5.1.5. Fungsi yang terdapat disini adalah fungsi *rulebase_store*.

Fungsi *rulebase_store* di kasus penggunaan ini berguna untuk menambahkan data-data *rule based* tiap-tiap sensor, data-data tersebut berupa nilai batas atas dan nilai batas bawah dari suatu sensor dimana nilai tersebut sebagai acuan untuk menampilkan informasi apa yang ingin ditampilkan jika data berada di atas batas atas nilai ataupun di bawah batas bawah nilai sensor, data yang disimpan ini berperan juga sebagai pemberi informasi saat mekanisme *procedure* di basis data dijalankan. Berikut kode sumber 4.17 dari kasus penggunaan mendaftarkan *rule based* sensor.

```
FUNCTION rulebase_store(request)
    set input ← request->all()
    set id_user ← Auth::user()->id
    set rulebase_sensor[] ← [ ‘id_user’ => id_user,
        ‘ba_temperature’ => input[‘ba_temperature’],
        ‘informasi_ba_temperature’ =>
              input[‘informasi_ba_temperature’],
        ‘bb_temperature’ => input[‘bb_temperature’],
        ‘informasi_bb_temperature’ =>
              input[‘informasi_bb_temperature’],
        ‘ba_humidity’ => input[‘ba_humidity’],
        ‘informasi_ba_humidity’ =>
              input[‘informasi_ba_humidity’],
        ‘bb_humidity’ => input[‘bb_humidity’],
        ‘informasi_bb_humidity’ =>
              input[‘informasi_bb_humidity’],
        ‘ba_gas’ => input[‘ba_gas’],
        ‘informasi_ba_gas’ =>
              input[‘informasi_ba_gas’],
        ‘bb_gas’ => input[‘bb_gas’],
        ‘informasi_bb_gas’ =>
              input[‘informasi_bb_gas’],
        ‘ba_co’ => input[‘ba_co’],
        ‘informasi_ba_co’ =>
              input[‘informasi_ba_co’],
        ‘bb_co’ => input[‘bb_co’],
        ‘informasi_bb_co’ =>
              input[‘informasi_bb_co’]]
    call model->Rulebase::insert(rulebase_sensor)
    load redirect(/home)
ENDFUNCTION
```

Kode Sumber 4.17 *Pseudocode* Fungsi *rulebase_store*

Selanjutnya mengenai bagaimana *rule based* tiap-tiap informasi tersebut diolah dibahas melalui kode sumber berikut. Untuk mengolah dan menjadikan rule base tersebut berupa informasi maka diimplementasikan melalui sebuah prosedur di basis data agar *rule based* yang sudah didaftarkan berfungsi

sebagai batasan-batasan dalam nilai sensor, dan dapat memberikan informasi juga melalui prosedur basis data yang dijalankan.

```
1   Procedure UpdateRulebase(id_node)
2   BEGIN
3       Initial all variable
4       set date_time_now ← select now()
5       set ba_temperature ← (SELECT ba_temperature
.             FROM rulebase ORDER BY id DESC LIMIT 1);
6       set bb_temperature ← (SELECT bb_temperature
.             FROM rulebase ORDER BY id DESC LIMIT 1);
7       set ba_humidity ← (SELECT ba_humidity
.             FROM rulebase ORDER BY id DESC LIMIT 1);
8       set bb_humidity ← (SELECT bb_humidity
.             FROM rulebase ORDER BY id DESC LIMIT 1);
9       set ba_gas ← (SELECT ba_gas
.             FROM rulebase ORDER BY id DESC LIMIT 1);
10      set bb_gas ← (SELECT bb_gas
.             FROM rulebase ORDER BY id DESC LIMIT 1);
11      set ba_co ← (SELECT ba_co
.             FROM rulebase ORDER BY id DESC LIMIT 1);
12      set bb_co ← (SELECT bb_co
.             FROM rulebase ORDER BY id DESC LIMIT 1);
13      set jumlah_data_ba_temperature ← (SELECT
.             COUNT(*) FROM (SELECT * from data_sensor
.             WHERE id_node = id_node ORDER BY
.             created_at DESC LIMIT 50) AS d where
.             d.data_dht11_temperature >
.             ba_temperature);
14      set jumlah_data_bb_temperature ← (SELECT
.             COUNT(*) FROM (SELECT * from data_sensor
.             WHERE id_node = id_node ORDER BY
.             created_at DESC LIMIT 50) AS d where
.             d.data_dht11_temperature <
.             bb_temperature);
15      IF jumlah_data_ba_temperature >
.          jumlah_data_bb_temperature THEN
16            set info_temperature ← (SELECT
.             informasi_ba_temperature FROM rulebase
.             ORDER BY id DESC LIMIT 1);
17      ELSE IF jumlah_data_ba_temperature <
.          jumlah_data_bb_temperature THEN
```

```
18          set info_temperature ← (SELECT
 .          informasi_bb_temperature FROM rulebase
 .          ORDER BY id DESC LIMIT 1);
19     ELSE THEN
20          set info_temperature ← ‘normal’
21     END IF
23     set jumlah_data_ba_humidity ← (SELECT
 .          COUNT(*) FROM (SELECT * from data_sensor
 .          WHERE id_node = id_node ORDER BY
 .          created_at DESC LIMIT 50) AS d where
 .          d.data_dht11_humidity >
 .          ba_humidity);
24     set jumlah_data_bb_humidity ← (SELECT
 .          COUNT(*) FROM (SELECT * from data_sensor
 .          WHERE id_node = id_node ORDER BY
 .          created_at DESC LIMIT 50) AS d where
 .          d.data_dht11_humidity <
 .          bb_humidity);
25     IF jumlah_data_ba_humidity >
 .        jumlah_data_bb_humidity THEN
26          set info_humidity ← (SELECT
 .          informasi_ba_humidity FROM rulebase
 .          ORDER BY id DESC LIMIT 1);
27     ELSE IF jumlah_data_ba_humidity <
 .        jumlah_data_bb_humidity THEN
28          set info_humidity ← (SELECT
 .          informasi_bb_humidity FROM rulebase
 .          ORDER BY id DESC LIMIT 1);
29     ELSE THEN
30          set info_humidity ← ‘normal’
31     END IF
32     set jumlah_data_ba_gas ← (SELECT
 .          COUNT(*) FROM (SELECT * from data_sensor
 .          WHERE id_node = id_node ORDER BY
 .          created_at DESC LIMIT 50) AS d where
 .          d.data_mq135 > ba_gas);
33     set jumlah_data_bb_gas ← (SELECT
 .          COUNT(*) FROM (SELECT * from data_sensor
 .          WHERE id_node = id_node ORDER BY
 .          created_at DESC LIMIT 50) AS d where
 .          d.data_mq135 < bb_gas);
34     IF jumlah_data_ba_gas >
```

```
.          jumlah_data_bb_gas THEN
35            set info_gas ← (SELECT
.             informasi_ba_gas FROM rulebase
.             ORDER BY id DESC LIMIT 1);
36      ELSE IF jumlah_data_ba_gas <
.          jumlah_data_bb_gas THEN
37            set info_gas ← (SELECT
.             informasi_bb_gas FROM rulebase
.             ORDER BY id DESC LIMIT 1);
38      ELSE THEN
39            set info_gas ← 'normal'
40      END IF
41      set jumlah_data_ba_co ← (SELECT
.             COUNT(*) FROM (SELECT * from data_sensor
.             WHERE id_node = id_node ORDER BY
.             created_at DESC LIMIT 50) AS d where
.             d.data_mq7 > ba_co);
42      set jumlah_data_bb_co ← (SELECT
.             COUNT(*) FROM (SELECT * from data_sensor
.             WHERE id_node = id_node ORDER BY
.             created_at DESC LIMIT 50) AS d where
.             d.data_mq7 < bb_co);
43      IF jumlah_data_ba_co >
.          jumlah_data_bb_co THEN
44            set info_co ← (SELECT
.             informasi_ba_co FROM rulebase
.             ORDER BY id DESC LIMIT 1);
45      ELSE IF jumlah_data_ba_co <
.          jumlah_data_bb_co THEN
46            set info_co ← (SELECT
.             informasi_bb_co FROM rulebase
.             ORDER BY id DESC LIMIT 1);
47      ELSE THEN
48            set info_co ← 'normal'
49      END IF
50      INSERT INTO info_rulebase(id_node,
.            info_rulebase_temperature,
.            info_rulebase_humidity,
.            info_rulebase_gas,
.            info_rulebase_co,)
.            VALUES(id_node, info_temperature,
.            info_humidity, info_gas, info_co
```

```
.                    date_time-now)
51  END
```

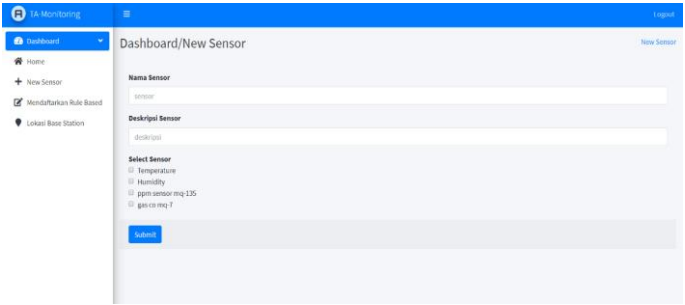
Kode Sumber 4.18 *Pseudocode* prosedur *UpdateRulebase*

4.4.2 Implementasi Antarmuka Pengguna Monitoring Sensor

Pada bagian ini menjelaskan tentang implementasi dari antarmuka pengguna yang telah diimplementasikan berdasarkan perancangan antarmuka pengguna pada bab 3.5.3. Berikut merupakan hasil dari implementasi rancangan antar pengguna.

4.4.2.1 Halaman Mendaftarkan Sensor

Halaman ini merupakan implementasi dari rancangan antarmuka pengguna pada 3.5.3.1 yaitu halaman mendaftarkan sensor. Halaman ini berguna untuk mendaftarkan sensor yang akan digunakan sebagai node sensor. Gambar 4.3 merupakan implementasi dari halaman mendaftarkan sensor.

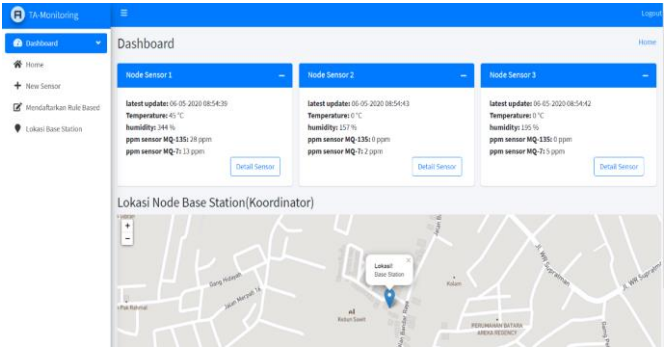


Gambar 4.3 Implementasi Halaman Mendaftarkan Sensor

4.4.2.2 Halaman Menampilkan *List* Sensor Dan Lokasi *Base Station*

Halaman ini merupakan implementasi dari rancangan antar muka pengguna pada 3.5.3.2 yaitu halaman menampilkan *list* sensor dan lokasi *base station*. Halaman ini juga berperan sebagai halaman utama dan menampilkan daftar sensor yang berfungsi sebagai node sensor dan juga menampilkan lokasi dari *base station*.

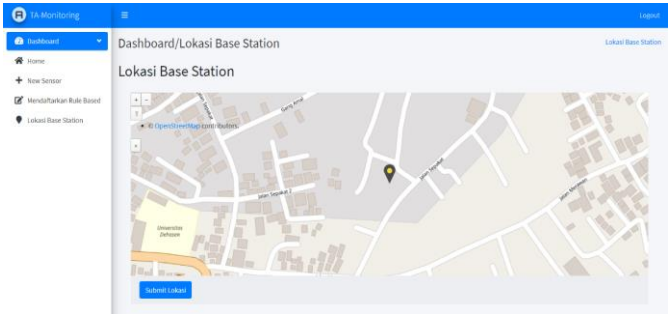
Gambar 4.4 merupakan implementasi dari halaman menampilkan *list* sensor dan lokasi *base station*.



Gambar 4.4 Implementasi Halaman Menampilkan *List* Sensor Dan Lokasi *Base Station*.

4.4.2.3 Halaman Menambahkan Lokasi *Base Station*

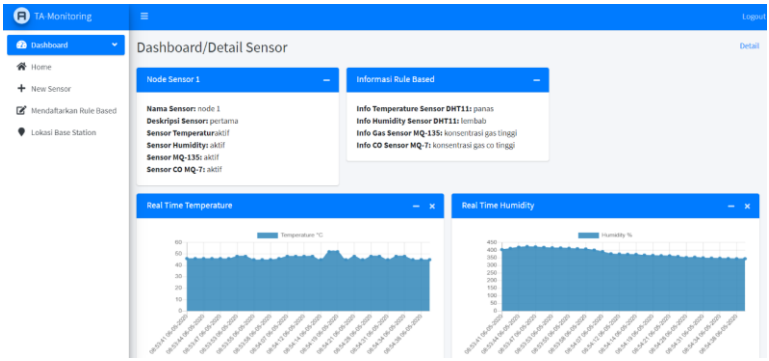
Halaman ini merupakan implementasi dari rancangan antarmuka pengguna pada 3.5.3.3 yaitu halaman menambahkan lokasi *base station*. Halaman ini berfungsi untuk menambahkan lokasi *base station* yaitu dengan menampilkan lokasi saat ini. Gambar 4.5 merupakan implementasi dari halaman menambahkan lokasi *base station*.



Gambar 4.5 Implementasi Halaman Menambahkan Lokasi *Base Station*

4.4.2.4 Halaman Menampilkan Detail Sensor

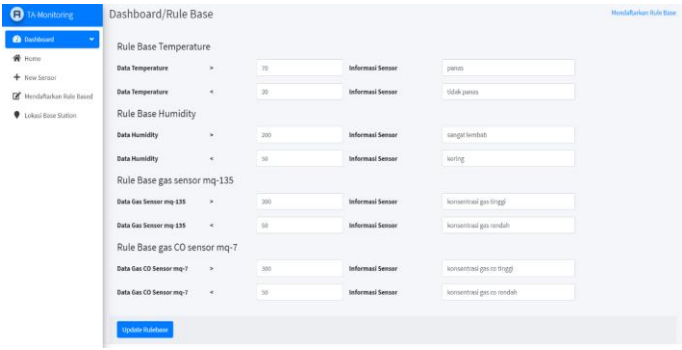
Halaman ini merupakan implementasi dari rancangan antarmuka pengguna pada 3.5.3.4 yaitu halaman menampilkan detail sensor. Halaman ini berfungsi untuk menampilkan detail sensor berupa grafik-grafik dari data sensor yang dikumpukan dan juga selain itu juga terdapat status sensor dan informasi *rule based* yang ditentukan. Gambar 4.6 merupakan implementasi dari halaman menampilkan detail sensor.



Gambar 4.6 Implementasi Halaman Menampilkan Detail Sensor

4.4.2.5 Halaman Mendaftarkan *Rule Based*

Halaman ini merupakan implementasi dari rancangan antarmuka pengguna pada 3.5.3.5 yaitu halaman mendaftarkan *rule based.* Halaman ini berfungsi untuk mendaftarkan *rule based* agar dapat menampilkan informasi kondisi keaadaan lingkungan berdasarkan data sensor. Gambar 4.7 merupakan implementasi dari halaman mendaftarkan *rule based*.



Gambar 4.7 Implementasi Halaman Mendaftarkan *Rule Based*

(Halaman ini sengaja dikosongkan)

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijabarkan mengenai uji coba dan evaluasi mengenai tugas akhir yang telah dikerjakan. Secara garis besar terdapat 3 jenis uji coba yang akan dilakukan yaitu uji coba studi kasus di jalan raya, fungsional, dan uji coba performa. Mekanisme dan cara uji coba akan di jelaskan melalui lingkungan pengujian uji coba dan skenario uji coba, selanjutnya pada bagian akhir akan dijelaskan mengenai evaluasi hasil uji coba yang telah dilakukan.

5.1 Lingkungan Uji Coba

Lingkungan uji coba menjelaskan mengenai peralatan dan perangkat yang digunakan dalam melakukan pengujian sistem. Untuk melakukan uji coba dilakukan sesuai implementasi yang telah dilakukan yaitu menggunakan 3 node sensor, 1 *base station* (media komunikasi antar node ke basis data). Untuk lebih jelasnya lingkungan uji coba memiliki spesifikasi sebagai berikut:

- Tiga node sensor memiliki beberapa komponen sebagai berikut :
    - Mikrokontroller Arduni UNO R3
    - Modul sensor suhu, kelembapan, kualitas udara, gas karbon monoksida
    - Modul nRF24L01
    - Papan kerja (*breadboard*)
    - Kabel *jumper* + kabel USB *Type A to Type B*
    - *Power bank* kapasitas 10000 mAh *output* 5V
- *Base station* memiliki komponen sebagai berikut:
    - Raspberry Pi 3 B+
    - Modul nRF24L01
    - Kabel *jumper*
    - Jaringan internet via WiFi Telkom 10 Mbps

Untuk lokasi pengujian dilakukan di sekitar jalan raya rumah penulis. Untuk jarak node sensor secara umum untuk dilakukan uji coba fungsionalitas yaitu dengan *base station* memiliki jarak 5 meter dengan *base station* yang diletakkan di teras rumah penulis. Untuk jarak masing-masing node sensor yaitu 2 meter. Untuk rentang waktu pengambilan data selama 20 menit untuk pengambilan data tiap pagi, siang, sore, dan malam hari.

Gambar 5.1 Lokasi Uji Coba

5.2 Skenario Uji Coba Studi Kasus Jalan Raya

Pada skenario uji coba studi kasus jalan raya ini, merupakan uji coba yang dilakukan dengan menjalankan setiap sistem secara utuh yaitu dari node sensor, *base station*, dan juga website monitoring sensor dalam melakukan monitoring data-data sensor. Pada studi kasus ini akan dilakukan pada rentang waktu yang berbeda-beda yaitu pagi, siang, sore, dan malam hari. Skenario uji coba ini bertujuan untuk mendapatkan data-data dari tiap-tiap sensor yaitu memantau kondisi di pagi, siang, sore, dan malam hari.

Untuk node sensor nya akan diberi jarak masing-masing yaitu 1-2 meter, untuk jarak node sensor ke *base station* yaitu 5-7 meter. Node sensor diletakkan dipinggir jalan sekitar rumah

penulis, dan waktu yang di uji coba berkisar selama 20 menit untuk tiap rentang waktunya yaitu pagi, siang, sore, dan malam hari.

Hasil uji coba memantau kondisi jalan saat di pagi hari dapat dilihat melalui Tabel 5.1 dibawah. Berikut merupakan hasil uji coba saat di pagi hari.

Tabel 5.1 Hasil Uji Coba Studi Kasus Jalan Raya Di Pagi Hari

Node	**Rata-Rata Sensor DHT11 Suhu (°C)**	**Rata-rata Sensor DHT11 kelembapan (%)**	**Rata-Rata Sensor MQ-7 Gas CO (ppm)**	**Rata-Rata Sensor MQ-135 Kualitas udara (ppm)**
1	25	89,1	8,3	276,4
2	25	95	8,7	276,3
3	27	83	8,5	278

Hasil uji coba memantau kondisi jalan saat di siang hari dapat dilihat melalui Tabel 5.2 dibawah. Berikut merupakan hasil uji coba saat di siang hari.

Tabel 5.2 Hasil Uji Coba Studi Kasus Jalan Raya Di Siang Hari

Node	**Rata-Rata Sensor DHT11 Suhu (°C)**	**Rata-rata Sensor DHT11 kelembapan (%)**	**Rata-Rata Sensor MQ-7 Gas CO (ppm)**	**Rata-Rata Sensor MQ-135 Kualitas udara (ppm)**
1	47,1	69,4	12,2	173,7
2	44,2	69,8	13,4	178
3	41	68	12,8	165

Hasil uji coba memantau kondisi jalan saat di sore hari dapat dilihat melalui Tabel 5.3 dibawah. Berikut merupakan hasil uji coba saat di sore hari.

Tabel 5.3 Hasil Uji Coba Studi Kasus Jalan Raya Di Sore Hari

Node	**Rata-Rata Sensor DHT11 Suhu (°C)**	**Rata-rata Sensor DHT11 kelembapan (%)**	**Rata-Rata Sensor MQ-7 Gas CO (ppm)**	**Rata-Rata Sensor MQ-135 Kualitas udara (ppm)**
1	31,7	74,4	10	185,6
2	32,3	70,7	12	199,2
3	32	86	12,1	153,2

Hasil uji coba memantau kondisi jalan saat di malam hari dapat dilihat melalui Tabel 5.4 Sdibawah. Berikut merupakan hasil uji coba saat di malam hari.

Tabel 5.4 Hasil Uji Coba Studi Kasus Jalan Raya Di Malam Hari

Node	**Rata-Rata Sensor DHT11 Suhu (°C)**	**Rata-rata Sensor DHT11 kelembapan (%)**	**Rata-Rata Sensor MQ-7 Gas CO (ppm)**	**Rata-Rata Sensor MQ-135 Kualitas udara (ppm)**
1	26,1	87,9	10,5	188,1
2	27,6	83	9,8	165,9
3	28	81	12,3	182,7

5.3 Skenario Uji Coba Fungsionalitas

Untuk skenario uji coba fungsionalitas merupakan uji coba dari fungsi-fungsi sistem monitoring sensor berdasarkan kasus-kasus penggunaan yang telah diimplementasikan. Berikut merupakan skenario uji coba fungsionalitas website monitoring sensor.

5.3.1 Skenario Uji Coba (UJ–F01) – Mendaftarkan Sensor

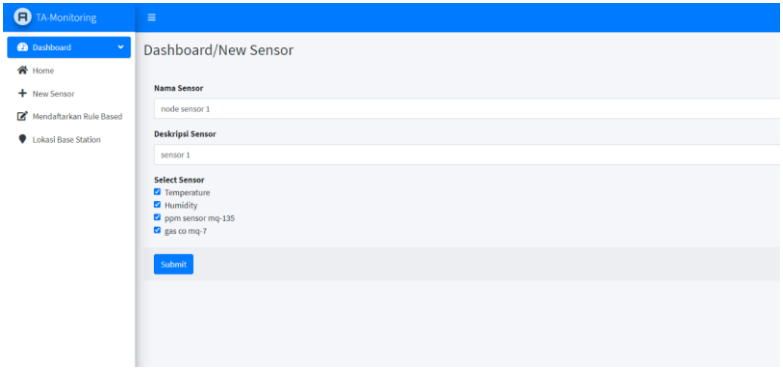
Uji coba mendaftarkan sensor berfungsi untuk menguji apakah sistem dapat menjalankan fungsi dalam mendaftarkan sensor. Skenario uji coba ini diawali dengan pengguna telah *login* ke dalam sistem monitoring sensor. Tabel 5.5 merupakan skenario uji coba mendaftarkan sensor.

Tabel 5.5 Skenario Uji Coba Mendaftarkan Sensor

ID	**UJ-F01**
Nama	Uji Coba Mendaftarkan Sensor
Tujuan Uji Coba	Menguji fungsional website monitoring sensor dalam mendaftarkan sensor
Kondisi Awal	Pengguna telah login dan masuk ke halaman utama website.
Skenario	1. Pengguna *login* ke sistem 2. Website menampilkan halaman utama 3. Pengguna memilih menu *new sensor* 4. Pengguna mengisi data-data sesuai *isian dari form* 5. Pengguna mengklik tombol *submit* 6. Website akan mengarahkan ke halaman utama
Masukan	Data-data berupa nama sensor, deskripsi sensor, pilihan sensor yang akan diaktifkan.
Keluaran	Node sensor dan sensor-sensor yang dipilih ditampilkan di halaman utama website

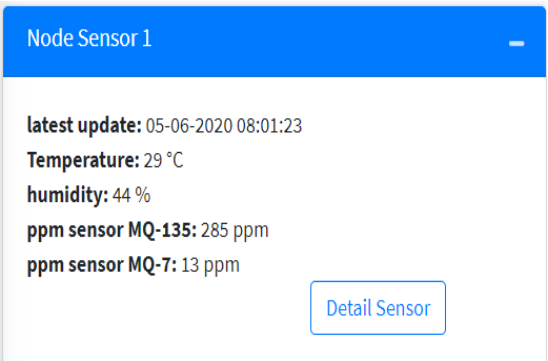
Hasil Yang Diharapkan	Berhasil mendaftarkan sensor dan menampilkannya di halaman utama.

Hasil uji coba pada skenario mendaftarkan senssor langkah-langkahnya yaitu pertama pengguna telah melakukan *login* ke dalam sistem dan selanjutnya pengguna memilih menu *new sensor* dan akan diarahkan ke halaman untuk mengisi form data sensor selanjutnya setelah pengguna melakukan *submit* data maka diarahkan ke halaman utama dengan menunjukkan informasi node sensor yang telah didaftarkan, hal ini menunjukkan bahwa fungsionalitas mendaftarkan sensor berjalan dengan baik.



Gambar 5.2 Uji Coba Mendaftarkan Sensor

Setelah mendaftarkan sensor, maka ditampilan utama akan muncul informasi node sensor dengan sensor-sensor yang didaftarkan seperti Gambar 5.3 berikut.



Gambar 5.3 Hasil Uji Coba UJ-F01 Mendaftarkan Sensor

5.3.2 Skenario Uji Coba (UJ–F02) – Menampilkan *List* Sensor Dan Lokasi *Base Station*

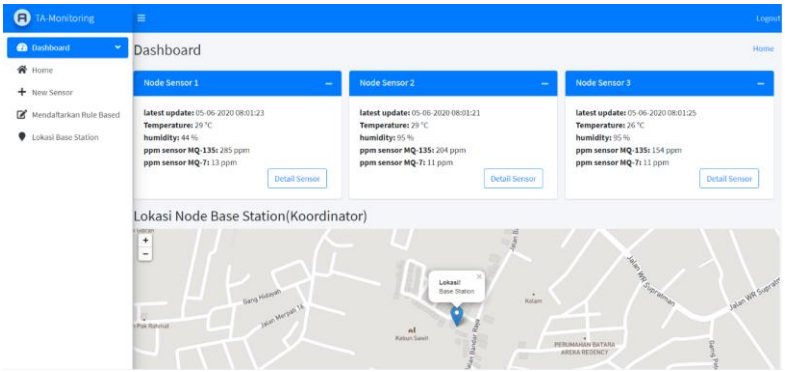
Selanjutnya yaitu uji coba menampilkan *list* sensor dan lokasi *base station* berfungsi untuk menguji apakah sistem dapat menjalankan fungsi dalam menampilkan *list* sensor yang telah disimpan di basis data dan juga dapat menampilkan lokasi *base station*. Tabel 5.6 merupakan skenario uji coba mendaftarkan sensor.

Tabel 5.6 Skenario Uji Coba Menampilkan *List* Sensor Dan Lokasi *Base Station*

ID	**UJ-F02**
Nama	Uji Coba Menampilkan *List* Sensor Dan Lokasi *Base Station*
Tujuan Uji Coba	Menguji fungsional website dalam menampilkan *list* sensor yang didaftarkan dan menampilkan lokasi *base station.*
Kondisi Awal	Pengguna membuka halaman monitoring sensor.
Skenario	1. Pengguna membuka website 2. Pengguna mengisi data dan *login*

	3. Pengguna diarahkan ke halaman utama
Masukan	-
Keluaran	Data *list* node sensor dan data-data sensor yang didaftarkan, juga lokasi *base station.*
Hasil Yang Diharapkan	Tampilan *list* node sensor dan data-data sensor, juga menampilkan lokasi dari *base station.*

Langkah-langkah dalam melakukan uji coba ini yaitu pengguna masuk ke halaman *login* sistem dan mengisi *email* dan *password* untuk *login* setelah berhasil maka pengguna dapat melihat tampilan halaman utama website yang menampikan *list* dari node sensor yang didalamnya berupa data-data terakhir dari sensor yang masuk dan juga terdapat informasi lokasi dari *base station.* Hasil uji coba ini menunjukkan uji coba dari fungsional menampilkan *list* sensor dan lokasi *base station* berjalan dengan baik. Untuk lebih jelasnya dapat dilihat melalui Gambar 5.4 berikut



Gambar 5.4 Hasil Uji Coba UJ-F02 Menampilkan *List* Sensor Dan Lokasi *Base Station*

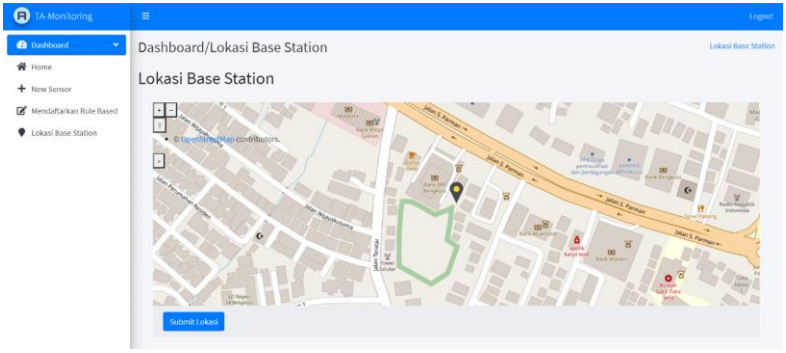
5.3.3 Skenario Uji Coba (UJ–F03) – Menambahkan Lokasi *Base Station*

Skenario selanjutnya yaitu uji coba menambahkan lokasi dari *base* station berfungsi untuk menguji apakah sistem dapat menambahkan lokasi *base station* dan disimpan di basis data. Skenario uji coba ini diawali dengan pengguna telah *login* ke dalam sistem monitoring sensor. Tabel 5.7 merupakan skenario uji coba menambahkan lokasi *base station*.

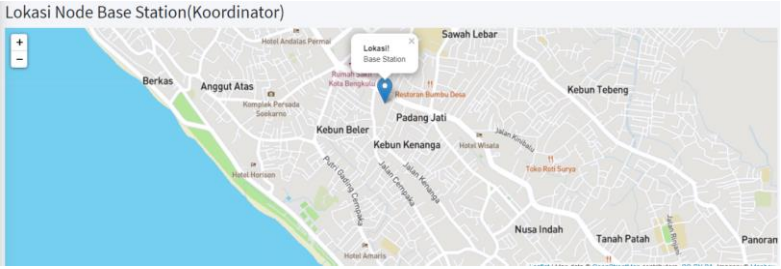
Tabel 5.7 Skenario Uji Coba Menambahkan Lokasi *Base Station*

ID	**UJ-F03**
Nama	Uji Coba Menambahkan Lokasi *Base Station*
Tujuan Uji Coba	Menguji fungsional website dalam menambahkan lokasi *base station* ke sistem.
Kondisi Awal	Pengguna telah *login* dan diarahkan di halaman utama website.
Skenario	1. Pengguna memilih menu lokasi *base station* 2. Website menampilkan peta lokasi. 3. Pengguna mengizinkan untuk mengakses lokasi. 4. Website menampilkan lokasi saat ini. 5. Pengguna mengklik tombol submit lokasi 6. Pengguna diarahkan ke halaman utama
Masukan	Lokasi saat ini di peta yaitu berupa latitude dan longitude yang otomatis diambil datanya oleh sistem.
Keluaran	Data lokasi disimpan dan dapat ditampilkan di halaman utama.
Hasil Yang Diharapkan	Menampilkan lokasi dari *base station* di halaman utama dari lokasi yang telah didaftarkan.

Berdasarkan skenario uji coba pengguna telah masuk ke halaman utama melalui *login* setelah itu pengguna memilih menu lokasi *base station* dan diarahkan ke halaman untuk menambahkan lokasi, setelah masuk maka pengguna mengizinkan sistem untuk mengetahui lokasi saat ini dan sistem langsung otomatis mendapatkan nilai dari latitude dan longitude lokasi, setelah pengguna *submit lokasi* maka data latitude dan longitude tersimpan di basis data dan dapat menampilkan lokasi di halaman utama sistem. Hal ini menunjukkan uji coba fungsionalitas menambahkan lokasi *base station* berjalan dengan baik. Berikut Gambar 5.5 saat menambahkan lokasi dan Gambar 5.6 hasil uji coba dalam menambahkan lokasi *base station*.



Gambar 5.5 Uji Coba Menambahkan Lokasi *Base Station*



Gambar 5.6 Hasil Uji Coba UJ-F03 Menambahkan Lokasi *Base Station*

5.3.4 Skenario Uji Coba (UJ–F04) – Menampilkan Detail Sensor

Skenario selanjutnya yaitu uji coba menampilkan detail sensor berfungsi untuk menguji apakah sistem dapat menampilkan detail sensor yaitu berupa grafik-grafik tiap-tiap sensor yang terdapat dalam node sensor dan juga menampilkan *informasi rule based* sensor . Skenario uji coba ini diawali dengan pengguna telah *login* ke dalam sistem monitoring sensor. Tabel 5.8 merupakan skenario uji coba menampilkan detail sensor.

Tabel 5.8 Skenario Uji Coba Menampilkan Detail Sensor

ID	**UJ-F04**
Nama	Uji Coba Menampilkan Detail Sensor
Tujuan Uji Coba	Menguji fungsional website dalam menampilkan detail sensor.
Kondisi Awal	Pengguna telah *login* dan diarahkan di halaman utama website.
Skenario	1. Website menampilkan halaman utama. 2. Pengguna mengklik tombol detail sensor yang terdapat di *card* tampilan menu utama website. 3. Pengguna diarahkan ke halaman detail sensor sesuao node sensor yang dipilih.
Masukan	-
Keluaran	Halaman detail sensor berupa status sensor, grafik-grafik sensor, dan informasi *rule based* sensor.
Hasil Yang Diharapkan	Menampilkan halaman detail sensor dan informasi terkait dari detail sensor tersebut.

Uji coba berdasarkan skenario uji coba menampilkan detail sensor yaitu mula-mula pengguna telah *login* dan diarahkan ke halaman utama, setelah itu pengguna mengklik *button* di *card* informasi sensor yaitu detail sensor setelah itu pengguna diarahkan ke halaman detail sensor yang menampilkan grafik-grafik data sensor. Hal ini menunjukkan uji coba fungsional menampilkan detail sensor berjalan dengan baik. Berikut merupakan Gambar 5.7 hasil uji coba menampilkan detail sensor.

Gambar 5.7 Hasil Uji Coba UJ-F04 Menampilkan Detail Sensor

5.3.5 Skenario Uji Coba (UJ–F05) – Mendaftarkan *Rule Based* Sensor

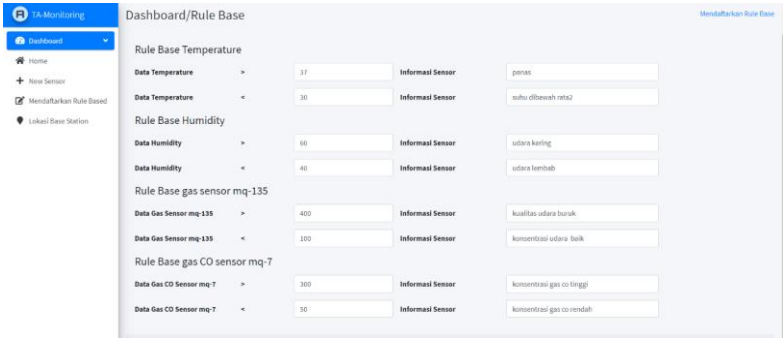
Skenario terakhir yaitu uji coba mendaftarkan *rule based* sensor yang berfungsi untuk menguji apakah sistem dapat mendaftarkan *rule based* sensor. Skenario uji coba ini diawali dengan pengguna telah *login* ke dalam sistem monitoring sensor. Tabel 5.9 merupakan skenario uji coba mendaftarkan *rule based* sensor.

Tabel 5.9 Skenario Uji Coba Mendaftarkan *Rule Based* Sensor

ID	**UJ-F05**
Nama	Uji Coba Mendaftarkan *Rule Based* Sensor

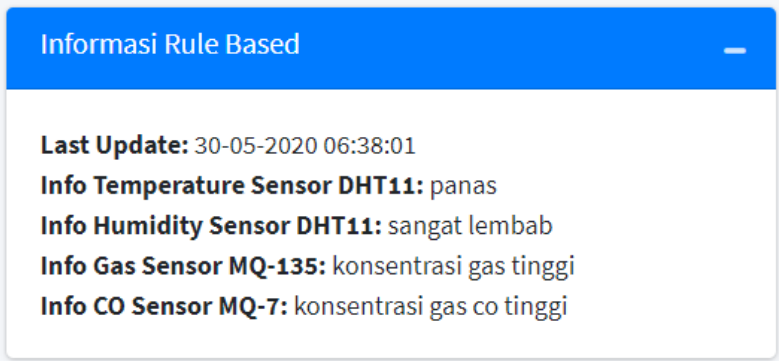
Tujuan Uji Coba	Menguji fungsional website untuk mendaftarkan *rule based* sensor.
Kondisi Awal	Pengguna telah *login* dan diarahkan di halaman utama website.
Skenario	1. Website menampilkan halaman utama 2. Pengguna memilih menu mendaftarkan *rule based* sensor. 3. Website menampilkan halaman *form* isian data untuk mendaftarkan *rule based* sensor 4. Pengguna mengisi data sesuai isian *form*. 5. Pengguna mengklik tombol *submit.* 6. Pengguna diarahkan ke halaman utama.
Masukan	Isian data berupa batas atas, dan batas bawah nilai sensor yang akan diberi informasi nya sesuai keinginan pengguna.
Keluaran	*Rule based* didaftarkan dan kembali ke halaman utama.
Hasil Yang Diharapkan	Data *rule based* disimpan ke basis data dan diolah melalui prosedur di basis data selanjutnya dapat ditampilkan informasi *rule based* di halaman detail sensor.

Uji coba mendaftarkan *rule based* berdasarkan skenario uji coba yaitu pengguna telah *login* ke dalam website dan diarahkan ke halaman utama kemudian pengguna memilih menu mendaftarkan *rule based*, dan pengguna akan ditampilkan *form* isian untuk *rule based* tiap-tiap sensor setelah mengisi *form* isian maka pengguna mengklik *button submit* dan selesai mendaftarkan *rule based*, dan website dapat menampilkan informasi *rule based* yang berhasil didaftarkan dengan melalui prosedur basis data. Hasil uji coba ini menunjukkan uji coba fungsionalitas mendaftarkan rule based berjalan dengan baik. Berikut Gambar 5.8 dari hasil uji coba dalam mendaftarkan *rule based.*



Gambar 5.8 Uji Coba UJ-F05 Mendaftarkan *Rule Based*

Hasil informasi *rule based* yang didaftarkan akan ditampilkan di detail sensor untuk masinf-masing node sensor dari *rule based* yang telah didaftarkan. Gambar 5.9 merupakan hasil dari uji coba dalam mendaftarkan *rule based* sensor.



Gambar 5.9 Hasil Uji Coba UJ-F05 Mendaftarkan *Rule Based*

5.4 Skenario Uji Coba Performa

Skenario uji coba performa terbagi menjadi 2 yaitu uji coba performa dalam jarak pengiriman dan uji coba *delay realtime* data masuk ke database, Skenario uji coba performa ini bertujuan untuk

mengetahui ketahanan atau relialbilitas dari sistem yang telah dibangun. Berikut merupakan skenario uji coba performa.

5.4.1 Skenario Uji Coba Performa – Akurasi Jarak Pengiriman

Pada skenario uji performa yaitu menguji akurasi jarak pengiriman dari node sensor ke *base station* dan menguji data yang masuk ke basis data apakah sesuai atau tidak yaitu mengirim data dengan *delay* waktu sebesar 5 detik, dan dilakukan selama 2 menit, uji coba ini dilakukan 10 kali percobaan. Untuk jarak node sensor dilakukan dengan jarak 1 meter, 5 meter dan 10 meter. Berikut Tabel 5.10 merupakan tabel dari hasil uji coba jarak pengiriman.

Tabel 5.10 Hasil Uji Coba Performa – Jarak Pengiriman

Akurasi pengiriman		
Jarak ≤ 1 meter	**Jarak ± 5 meter**	**Jarak ± 10 meter**
99 %	87,2 %	82,2 %

Berdasarkan hasil uji coba semakin jauh jarak pengiriman menyebabkan akurasi pengiriman juga semakin menurun, untuk uji coba 20 meter dan seterusnya tidak dilanjutkan dikarenakan koneksi dari modul nRF24L01 node sensor ke *base station* sering terputus dan harus di*restart* ulang.

5.4.2 Skenario Uji Coba Performa - *Delay Realtime* Pengiriman

Pada skenario uji coba performa yaitu *delay realtime* pengiriman yaitu bertujuan untuk mengetahui waktu *delay* sistem saat mengirim data dari node sensor ke *base station* dan data tersimpan ke basis data. Skenario uji coba ini dilakukan dengan menguji 1, 2, dan 3 node sensor terhubung ke *base station*, setiap uji coba dilakukan 10 kali dengan 1 kali percobaan 1 menit pengiriman dengan parameter delay node sensor 5 detik, 3 detik dan 1 detik. Berikut merupakan hasil uji coba performa *delay*

realtime pengiriman dengan parameter delay node sensor 5 detik melalui Tabel 5.11.

Tabel 5.11 Hasil Uji Coba Performa – *Delay Realtime* Pengiriman Dengan Delay Node Sensor 5 Detik

Uji *Delay Realtime*	***Delay Realtime* Pengiriman**					
	1 Node Sensor	**2 Node Sensor**		**3 Node Sensor**		
Node	Node 1	Node 1	Node 2	Node 1	Node 2	Node 3
Rata-rata node(detik)	5,066	5,065	5,091	5,909	6,491	5,666
Rata-rata keseluruhan (detik)	5,066	5,078		6,022		

Berikut merupakan hasil uji coba performa *delay realtime* pengiriman dengan parameter delay node sensor 3 detik melalui Tabel 5.12.

Tabel 5.12 Hasil Uji Coba Performa – *Delay Realtime* Pengiriman Dengan Delay Node Sensor 3 Detik

Uji *Delay Realtime*	***Delay Realtime* Pengiriman**					
	1 Node Sensor	**2 Node Sensor**		**3 Node Sensor**		
Node	Node 1	Node 1	Node 2	Node 1	Node 2	Node 3
Rata-rata node(detik)	3,268	3,866	3,851	3,297	5,090	3,327
Rata-rata keseluruhan (detik)	3,268	3,858		3,905		

Berikut merupakan hasil uji coba performa *delay realtime* pengiriman dengan parameter delay node sensor 1 detik melalui Tabel 5.13.

Tabel 5.13 Hasil Uji Coba Performa – *Delay Realtime* Pengiriman Dengan Delay Node Sensor 1 Detik

Uji *Delay Realtime*	***Delay Realtime* Pengiriman**					
	1 Node Sensor	**2 Node Sensor**		**3 Node Sensor**		
Node	Node 1	Node 1	Node 2	Node 1	Node 2	Node 3
Rata-rata node(detik)	1,091	1,101	1,092	1,150	2,114	1,580
Rata-rata keseluruhan (detik)	1,091	1,097		1,615		

Berdasarkan hasil uji coba performa *delay realtime* pengiriman maka dapat disimpulkan bahwa semakin banyak node sensor maka semakin lama *delay realtime* pengiriman data.

5.5 Evaluasi Umum Skenario Uji Coba

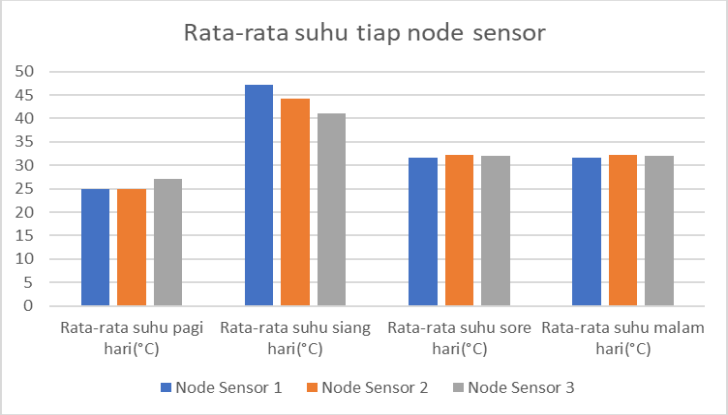
Berikut adalah evaluasi dari hasil uji coba dari masing skenario. Terdapat 3 point yang akan di evaluasi yaitu uji coba studi kasus jalan raya, uji coba fungsionalitas, dan uji coba performa. Evaluasi ini merupakan rangkuman dari beberapa hasil dari skenario uji coba yang diuji cobakan.

5.5.1 Evaluasi Uji Coba Studi Kasus Jalan Raya

Berikut merupakan rangkuman dari hasil uji coba yang telah dilakukan dalam uji coba studi kasus di jalan raya dengan sistem yang telah dibuat beberapa data yang diambil yaitu nilai suhu,

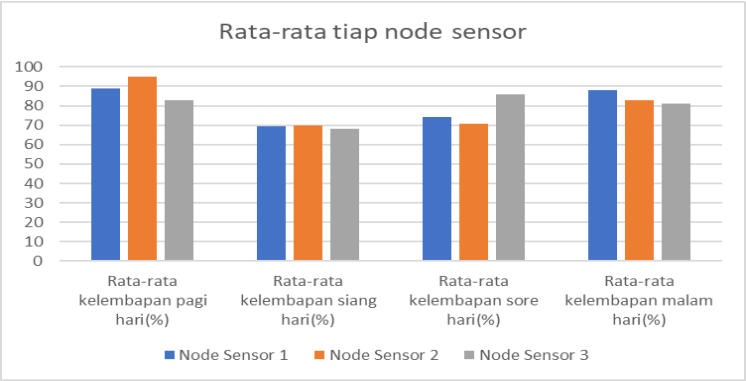
kelembapan, kualitas udara, dan konsentrasi dari gas karbon monoksida.

Gambar 5.10 merupakan evaluasi nilai suhu dari tiap-tiap node sensor dalam melakukan uji coba. Berdasarkan grafik tersebut dapat disimpulkan suhu paling tinggi saat siang hari dan paling rendah saat di pagi hari.



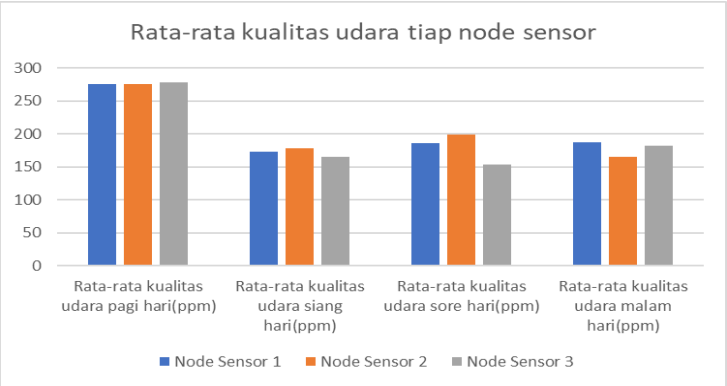
Gambar 5.10 Grafik Rata-Rata Suhu Setiap Node Sensor

Gambar 5.11merupakan evaluasi nilai kelembapan dari tiap-tiap node sensor dalam melakukan uji coba. Berdasarkan grafik tersebut dapat disimpulkan bahwa nilai kelembapan dipagi hari lebih tinggi dan kelembapan di siang hari yang paling rendah.



Gambar 5.11 Grafik Rata-Rata Kelembapan Setiap Node Sensor

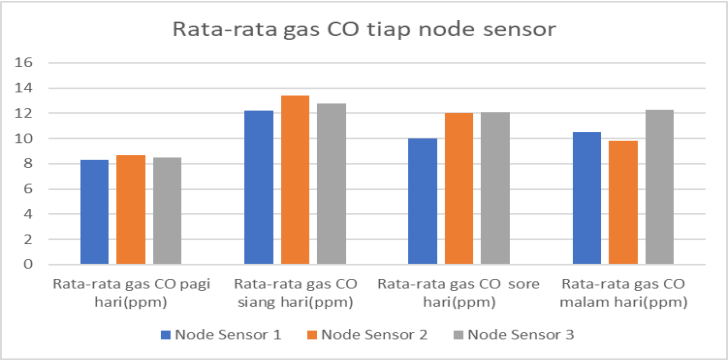
Gambar 5.12 merupakan evaluasi nilai kualitas udara dari tiap-tiap node sensor dalam melakukan uji coba. Berdasarkan dari grafik tersebut nilai kualitas udara di pagi hari paling tinggi dan rata-rata kualitas udara di siang, sore, dan malam hari tidak terlalu signifikan berbeda.



Gambar 5.12 Grafik Rata-Rata Kualitas Udara Setiap Node Sensor

Gambar 5.13 merupakan evaluasi nilai konsentrasi gas CO(karbon monoksida) dari tiap-tiap node sensor dalam

melakukan uji coba. Berdasarkan grafik tersebut nilai konsentrasi gas CO yaitu karbon monoksida paling tinggi saat di siang hari dan paling rendah di saat pagi hari.



S

Gambar 5.13 Grafik Rata-Rata Gas CO Setiap Node Sensor

5.5.2 Evaluasi Uji Coba Fungsionalitas

Untuk uji coba fungsionalitas melalui skenario uji coba fungsionalitas untuk menguji fungsi-fungsi dan implementasi dari monitoring sensor yang berfungsi menampilkan data-data sensor. Berikut merupakan hasil uji coba fungsionalitas dijelaskan di Tabel 5.14.

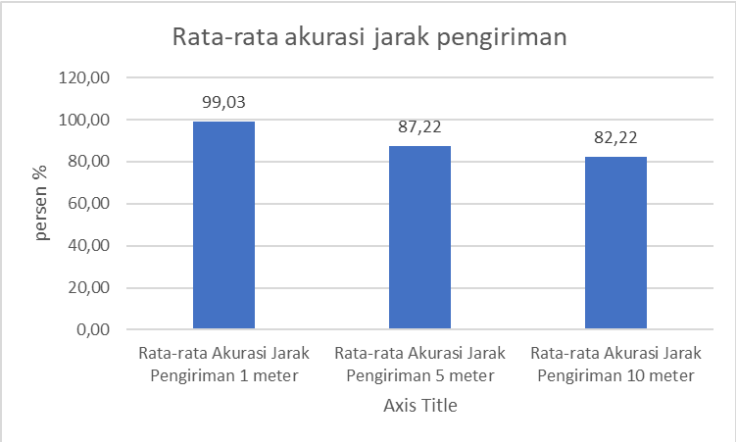
Tabel 5.14 Evaluasi Hasil Uji Coba Fungsionalitas

No	Kode Uji Coba	Evaluasi
1	UJ-F01	Dapat mendaftarkan sensor dan menampilkan sensor yang didaftarkan
2	UJ-F02	Dapat menampilkan *list* sensor dan lokasi *base station*
3	UJ-F03	Dapat menambahkan lokasi *base station*
4	UJ-F04	Dapat menampilkan detail sensor

5	UJ-F05	Dapat mendaftarkan *rule based* sensor dan menampilkan informasi dari *rule based* sensor

5.5.3 Evaluasi Uji Coba Performa

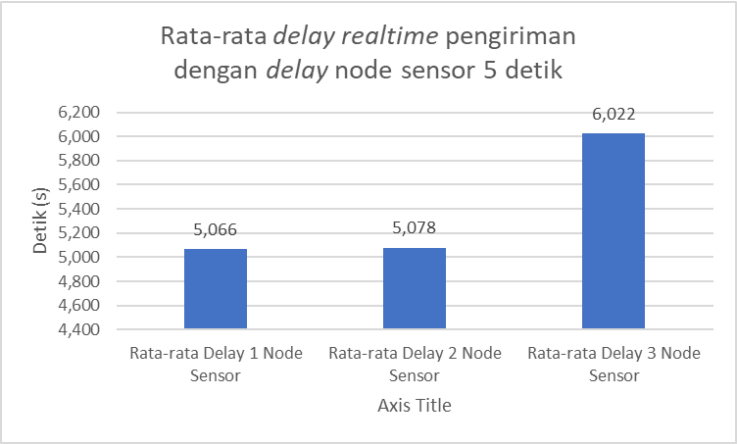
Evaluasi uji coba performa menjelaskan tentang rangkuman dari hasil uji coba performa. Uji coba performa terdiri dari uji coba performa akurasi jarak pengiriman dan *delay realtime* pengiriman data.Yang pertama dijelaskan yaitu uji coba rata-rata akurasi jarak pengiriman. Gambar 5.14 merupakan grafik dari hasil uji coba untuk mengukur akurasi jarak pengiriman, berdasarkan grafik semakin jauh jarak node sensor ke *base station* menyebabkan akurasi pengiriman menurun.



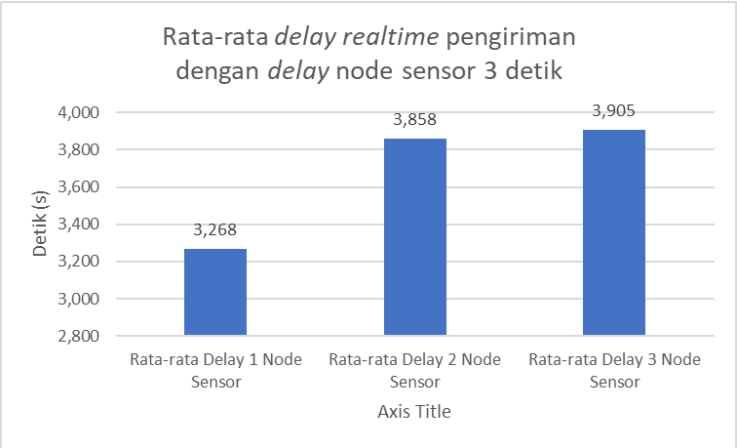
Gambar 5.14 Grafik Rata-Rata Akurasi Jarak Pengiriman

Selanjutnya mengenai evaluasi dari uji coba mengetahui *delay realtime* pengiriman data dari node sensor ke basis data. Gambar 5.15 merupakan hasil uji coba dari *delay realtime* pengiriman data dengan setiap uji cobanya menambahkan node sensor. Berdasarkan grafik tersebut semakin banyak node sensor maka *delay* pengiriman data akan semakin lama. Kestabilan

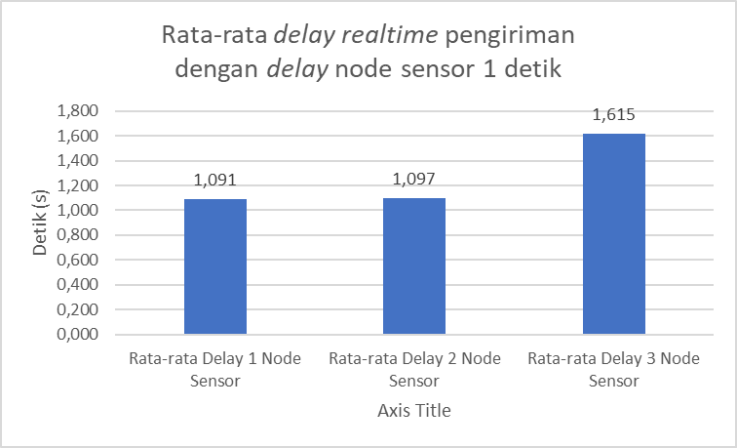
pengiriman data juga dipengaruhi faktor kecepatan internet yang stabil atau tidak, juga koneksi server tempat basis data menyimpan data.



Gambar 5.15 Grafik Rata-Rata *Delay Realtime* Pengiriman dengan delay node sensor 5 detik



Gambar 5.16 Grafik Rata-Rata *Delay Realtime* Pengiriman dengan delay node sensor 3 detik



Gambar 5.17 Grafik Rata-Rata *Delay Realtime* Pengiriman dengan delay node sensor 1 detik

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang diperoleh dari tugas akhir yang telah dikerjakan dan saran terkait pengembangan dari tugas akhir ini yang dapat dilakukan pada masa yang akan datang.

6.1 Kesimpulan

Kesimpulan yang diperoleh dari hasil uji coba dan evaluasi pada tugas akhir ini adalah sebagai berikut:

1. Mikrokontroller arduino uno R3 yang berfungsi sebagai node sensor dapat mendapatkan data-data sensor untuk melakukan monitoring.
2. Tiga node sensor dapat mengirimkan secara bersamaan dengan *base station* melalui komunikasi via modul nRF24L01.
3. Raspberry pi 3 B+ yang berfungsi sebagai *base station* dapat menerima data dan mengirimkan data ke *database* melalui jaringan internet.
4. Sistem monitoring sensor yaitu layanan *dashboard* untuk menampilkan data-data sensor secara *realtime* dapat berjalan dengan baik dan dapat memonitoring node sensor yang tersebar.
5. Untuk performa dari rancangan sistem yang telah dibangun yaitu sebagai berikut.
    a. Akurasi pengiriman data berdasarkan jarak pengiriman adalah sebagai berikut ≤1 m yaitu 99,03%, ±5 m yaitu 87,22%, ±10 m yaitu 85,22%. Dapat disimpulkan semakin jauh jarak pengiriman dari node sensor ke *base station* maka akurasi semakin menurun.
    b. *Delay* pengiriman data dengan menambahkan node sensor. Dapat disimpulkan semakin banyak node sensor dalam pengiriman data maka *delay* pengiriman data semakin lama.

6.2 Saran

Saran yang diberikan dari hasil uji coba dan evaluasi pada tugas akhir ini adalah sebagai berikut:

1. Menambahkan mekanisme pengiriman data ke server dengan berbagai mekanisme, agar saat koneksi dari *base station* terputus masih dapat mengirimkan ke *database*
2. Menambahkan jumlah *base station* atau node koordinator jika *base station* mati maka pengiriman data ke *database* dapat dilakukan oleh *base station* yang lain.

DAFTAR PUSTAKA

[1] Murthy,Manoj.2004.”Ad Hoc Wireless Sensor Networks Architecture and Protocols”.New Jersey:Bernard M.Godwin.

[2] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *Proceedings of the 33rd annual Hawaii international conference on sistem sciences*, 2000, pp. 10-pp.

[3] Alamri Atif et. all.2013." A Survey on Sensor-Cloud: Architecture, Applications, and Approaches".Saudi Arabia:King Saud University.

[4] Ganti Raghu K. dkk.2011."Mobile Crowdsensing: Current State and Future Challenges".*IEEE Comm. Mag.*

[5] Michael Friendly.2008. "Milestones in the history of thematic cartography, statistical graphics, and data visualization". Tersedia: http://datavis.ca/milestones/index.php?page=varieties+of+data+visualization. [Diakses:10-Juni-2020].

[6] Cellan-Jones Rory “Raspberry Pi” [Daring].Tersedia:https://id.wikipedia.org/wiki/Raspberry_Pi#cite_note-bbc.co.uk_BBC-dot.Rory:A1-3 [Diakses 22-Januari-2020]

[7] A. Kadir.2014 *Buku Pintar Pemrograman Arduino*. Yogyakarta: Mediakom.

[8] A. Rahmat, “Belajar Pemrograman Dasar Arduino.” [Online]. Tersedia: https://kelasrobot.com/belajar-pemrograman-dasar-arduino/. [Diakses: 15-Januari-2020].

[9] “nRF24L01” [Daring]. Tersedia:http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01 [Diakses 04-Januari-2020]

[10] L.Drake Fred, “General Python FAQ” [Daring]. Tersedia:https://docs.python.org/3/faq/general.html#what-is-python-good-for. [Diakses 22-Januari-2020].

[11] Rionardi Antonius, Dr. Teda Hudaya, ST, MEngSc. Dr. Ir. Tatang Hernas Soerawidjaja. “Hidrogenasi Elektrokimia Hidrokarbon Terpen”. Bandung: Universitas Katolik Parahyangan.

[12] Handayani Heny, "Pendeteksi Kualitas Udara dengan Sensor MQ-135 berbasis Mikrokontroller Tersedia:https://www.slideshare.net/HenyHandayani1/pendeteksi-kualitas-udara-dengan-sensor-mq-135-berbasis-microcontroller. [Diakses: 24-April-2020].

[13] Aji Saptaji, "Mengukur Suhu dan Kelembapan Udara dengan Sensor DHT11 dan Arduino" Tersedia: http://saptaji.com/2016/08/10/mengukur-suhu-dan-kelembaban-udara-dengan-sensor-dht11-dan-arduino/ . [Diakses : 24-April-2020]

[14] Anonim "Laravel Introduction" Tersedia secara daring :https://www.w3schools.in/laravel-tutorial/intro/ . [Diakses: 20-April-2020].

[15] Thidi "Mengenal Framework Twitter Bootstrap dan Penggunaannya pada Website" Tersedia: https://thidiweb.com/pengertian-bootstrap/ . [Diakses: 24-April-2020]

[16] Bakni Michel. 2020. "File:Wireless Sensor Network General Structure.svg". Tersedia : https://commons.wikimedia.org/wiki/File:Wireless_Sensor_Network_General_Structure.svg . [Diakses : 10 Juni 2020]

[17] Bukarev Roman. 2018. "IoT Mashup of Sensors, Cloud Software, and Machine Learning on the MapR Data Platform". Tersedia : https://mapr.com/blog/cool-fitness-tracker-using-iot-sensors-cloud-ml-on-mapr/ . [Diakses : 10- Juni-2020]

[18] Gardi Andre. 2018. "How to Use Chart.js". Tersedia : https://medium.com/javascript-in-plain-english/exploring-chart-js-e3ba70b07aa4. [Diakses : 10-Juni-2020]

(Halaman ini sengaja dikosongkan)

LAMPIRAN A

RINCIAN HASIL UJI COBA PERFORMA

Tabel 7.1 Rincian Hasil Uji Coba Performa Akurasi Jarak Pengiriman ≤ 1 meter

Pengiriman Data ≤ 1 meter			Percobaan ke-										Akurasi
			1	2	3	4	5	6	7	8	9	10	
Node 1	Jumlah Data	Masuk	24	22	24	24	24	24	23	24	24	24	98,75 %
		Tidak	0	2	0	0	0	0	1	0	0	0	
Node 2	Jumlah Data	Masuk	24	24	23	24	23	24	24	24	24	24	99,17 %
		Tidak	0	0	1	0	1	0	0	0	0	0	
Node 3	Jumlah Data	Masuk	24	24	24	23	24	24	23	24	24	24	99,17 %
		Tidak	0	0	0	1	0	0	1	0	0	0	
Akurasi Rata-Rata Keseluruhan													**99 %**

Tabel 7.2 Rincian Hasil Uji Coba Performa Akurasi Jarak Pengiriman ± 5 meter

Pengiriman Data ± 5 meter			Percobaan ke-										Akurasi
			1	2	3	4	5	6	7	8	9	10	
Node 1	Jumlah Data	Masuk	16	19	19	19	6	12	20	24	21	21	73,75 %
		Tidak	8	5	5	5	18	12	4	0	3	3	
Node 2	Jumlah Data	Masuk	23	24	24	20	22	21	21	23	24	24	94,17 %
		Tidak	1	0	0	4	2	3	3	1	0	0	
Node 3	Jumlah Data	Masuk	24	24	24	20	20	20	22	23	24	24	93,75 %
		Tidak	0	0	0	4	4	4	2	1	0	0	
Akurasi Rata-Rata Keseluruhan													**87,22 %**

Tabel 7.3 Rincian Hasil Uji Coba Performa Akurasi Jarak Pengiriman ± 10 meter

Pengiriman Data ± 10 meter			Percobaan ke-										Akurasi
			1	2	3	4	5	6	7	8	9	10	
Node 1	Jumlah Data	Masuk	23	24	18	20	15	15	16	13	22	21	77,92 %
		Tidak	1	0	6	4	9	9	8	11	2	3	
Node 2	Jumlah Data	Masuk	23	24	18	19	24	18	24	24	23	24	92,08 %
		Tidak	1	0	6	5	0	6	0	0	1	0	
Node 3	Jumlah Data	Masuk	24	24	23	24	24	18	6	12	12	17	76,67 %
		Tidak	0	0	1	0	0	6	18	12	12	7	
Akurasi Rata-Rata Keseluruhan													**82,22 %**

Tabel 7.4 Rincian Hasil Uji Coba Performa *Delay Realtime* Pengiriman Dengan *Delay* Node Sensor 5 Detik

Uji *Delay Realtime*		***Delay Realtime* Pengiriman**					
		1 Node Sensor	**2 Node Sensor**		**3 Node Sensor**		
Node		Node 1	Node 1	Node 2	Node 1	Node 2	Node 3
Percobaan Ke	**1**	4,723	5,009	4,987	5,212	5,396	4,963
	2	4,997	5,084	5,208	4,991	5,210	5,020
	3	4,917	5,085	5,120	5,061	5,063	5,106
	4	5,091	5,038	5,207	5,046	5,209	5,146
	5	5,084	4,967	5,035	5,046	5,029	4,886
	6	4,861	5,064	4,921	5,048	5,029	5,063
	7	5,487	5,157	5,066	9,260	10,878	10,613
	8	5,166	4,996	5,178	6,450	6,066	6,169
	9	5,068	5,057	5,060	5,562	8,656	5,013
	10	5,269	5,197	5,126	7,418	8,370	4,681
Rata-rata node(detik)		5,066	5,065	5,091	5,909	6,491	5,666
Rata-rata keseluruhan (detik)		5,066	5,078		6,022		

Tabel 7.5 Rincian Hasil Uji Coba Performa *Delay Realtime* Pengiriman Dengan *Delay* Node Sensor 3 Detik

Uji *Delay Realtime*		***Delay Realtime* Pengiriman**					
		1 Node Sensor	**2 Node Sensor**		**3 Node Sensor**		
Node		Node 1	Node 1	Node 2	Node 1	Node 2	Node 3
Percobaan Ke	**1**	3,284	2,018	2,050	3,985	5,091	3,130
	2	3,254	2,324	2,199	3,481	5,089	3,290
	3	3,090	3,221	3,409	3,091	5,091	3,235
	4	3,314	3,284	3,221	3,090	5,090	3,457
	5	3,092	6,075	5,964	3,202	5,091	3,402
	6	3,092	4,124	4,267	3,144	5,091	3,489
	7	3,369	5,122	5,022	3,202	5,091	3,346
	8	3,314	3,972	3,861	3,424	5,090	3,290
	9	3,593	4,504	4,404	3,203	5,090	3,457
	10	3,275	4,013	4,113	3,147	5,091	3,179
Rata-rata node(detik)		3,268	3,866	3,851	3,297	5,090	3,327
Rata-rata keseluruhan (detik)		3,268	3,858		3,905		

Tabel 7.6 Rincian Hasil Uji Coba Performa *Delay Realtime* Pengiriman Dengan *Delay* Node Sensor 1 Detik

Uji *Delay Realtime*		***Delay Realtime* Pengiriman**					
		1 Node Sensor	**2 Node Sensor**		**3 Node Sensor**		
Node		Node 1	Node 1	Node 2	Node 1	Node 2	Node 3
Percobaan Ke	**1**	1,091	1,103	1,090	1,154	2,106	1,690
	2	1,089	1,109	1,090	1,133	2,115	1,634
	3	1,102	1,100	1,090	1,154	2,116	1,536
	4	1,086	1,100	1,090	1,133	2,106	1,599

	5	1,086	1,100	1,090	1,154	2,125	1,564
	6	1,087	1,090	1,090	1,133	2,128	1,582
	7	1,099	1,122	1,111	1,218	2,087	1,535
	8	1,086	1,090	1,090	1,112	2,106	1,553
	9	1,086	1,100	1,090	1,133	2,115	1,553
	10	1,096	1,100	1,090	1,175	2,133	1,553
Rata-rata node(detik)		1,091	1,101	1,092	1,150	2,114	1,580
Rata-rata keseluruhan (detik)		1,091	1,097		1,615		

LAMPIRAN B

KODE SUMBER

1. Kode Sumber Node Sensor 1

```
#include <SPI.h>
#include <nRF24L01.h>
#include <printf.h>
#include <RF24.h>
#include <RF24_config.h>
#include <stdio.h>
//sensor mq135
#define pin_sensor_mq135 A1

//Sensor DHT11
#include "DHT.h" //library sensor yang telah diim
portkan
#define DHTPIN A2     //Pin apa yang digunakan
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);

const uint64_t pipe = 0xF0F0F0F0A1LL;

//global variable untuk sensor Mq-7
#define pinSensor A0
long RL = 1000; // 1000 Ohm
long Ro = 830; // 830 ohm ( SILAHKAN DISESUAIKAN)

//
RF24 radio(9,10);

void setup() {
  // put your setup code here, to run once:

  Serial.begin(9600);            /*Setting baudra
te of Serial Port to 9600*/
  dht.begin(); //prosedur memulai pembacaan modul
e sensor
```

```
30.
31.   radio.begin();
32.
33.   radio.setPALevel(RF24_PA_MIN);
34.   radio.setChannel(0x76);
35.   radio.openWritingPipe(pipe);
36.   radio.enableDynamicPayloads();
37.   radio.powerUp();
38.
39. }
40.
41. void loop() {
42.   // put your main code here, to run repeatedly:

43.   //keterangan variable send_data -
    > nomor_node|ppmCOMq7|CelciusDHT11|HumidityDHT11|
    ppmMQ135
44.   char send_data[15] = "1|";
45.   char SaveValueMq7[10];
46.   char SaveValueMq135[10];
47.
48.   char SaveValueTempDHT11[10];
49.   char SaveValueHumidityDHT11[10];
50.   int SensorValueMq7 = getSensorMq7();
51.   int SensorValueTempDHT11 = getTempSensorDHT11()
    ;
52.   int SensorValueHumidityDHT11 = getHumiditySenso
    rDHT11();
53.   int SensorValueMq135 = getSensorMq135();
54.
55.   Serial.println(SensorValueMq135);
56.   Serial.println(SensorValueMq7);
57.   Serial.println(SensorValueTempDHT11);
58.
59.
60.   //convert int to char
61.   itoa(SensorValueMq7,SaveValueMq7,10);
62.   itoa(SensorValueMq135,SaveValueMq135,10);
63.   itoa(SensorValueTempDHT11,SaveValueTempDHT11,10
    );
64.   itoa(SensorValueHumidityDHT11,SaveValueHumidity
    DHT11,10);
```

```
65.
66.
67.   Serial.println(SaveValueMq7);
68.   Serial.print("Start Send data from node 1 : \n"
    );
69.   //concat char sensor mq7
70.   strcat(send_data,SaveValueMq7);
71.   strcat(send_data,"|");
72.   strcat(send_data,SaveValueTempDHT11);
73.   strcat(send_data,"|");
74.   strcat(send_data,SaveValueHumidityDHT11);
75.   strcat(send_data,"|");
76.   strcat(send_data,SaveValueMq135);
77.
78.
79.
80.   Serial.println(send_data);
81.   radio.write(&send_data, sizeof(send_data));
82.
83.
84.   delay(5000);
85. }
86.
87. int getSensorMq7(){
88.   int sensorvalue = analogRead(pinSensor); // mem
    baca nilai ADC dari sensor
89.   float VRL= sensorvalue*5.00/1024;  // mengubah
    nilai ADC ( 0 - 1023 ) menjadi nilai voltase ( 0
    - 5.00 volt )
90.
91.   float Rs = ( 5.00 * RL / VRL ) - RL;
92.
93.   float ppm = 100 * pow(Rs / Ro,-
    1.53); // ppm = 100 * ((rs/ro)^-1.53);
94.   Serial.print("CO : ");
95.   Serial.print(ppm);
96.   Serial.println(" ppm");
97.   return ppm;
98. //  Serial.println();
99. }
100.
101.        int getTempSensorDHT11(){
```

```
102.         // put your main code here, to run repeat
edly:
103.
104.          //Pembacaan dalam format celcius (c)
105.          float celcius_1 = dht.readTemperature();

106.          //convert float to int agar bisa di assi
gn array of char
107.          int t_celcius = celcius_1;
108.          //mengecek pembacaan apakah terjadi kega
galan atau tidak
109.          if (isnan(celcius_1)) {
110.            Serial.println("Pembacaan data dari mo
dule sensor gagal!");
111.          }
112.
113.          //pembacaan nilai pembacaan data suhu
114.          Serial.print("Suhu : ");
115.          Serial.print(celcius_1); //format deraja
t celcius
116.          Serial.print(" 'C ");
117.          return(t_celcius);
118.         }
119.
120.        int getHumiditySensorDHT11(){
121.
122.          float humidity_1 = dht.readHumidity();
123.          int t_humidity = humidity_1;
124.          if (isnan(t_humidity)) {
125.            Serial.println("Pembacaan data dari mo
dule sensor gagal!");
126.          }
127.
128.          return t_humidity;
129.         }
130.
131.        int getSensorMq135(){
132.          int sensorValue;
133.
134.          sensorValue = analogRead(pin_sensor_mq13
5);        // read analog input pin 0
135.          Serial.print("Data Sensor Mq135=");
```

```
136.         Serial.print(sensorValue, DEC);
          // prints the value read
137.         Serial.println(" PPM");
138.
139.         return(sensorValue);
140.
141.         }
```

2. Kode Sumber Node Sensor 2

```
1.  #include <SPI.h>
2.  #include <nRF24L01.h>
3.  #include <printf.h>
4.  #include <RF24.h>
5.  #include <RF24_config.h>
6.  #include <stdio.h>
7.
8.  //sensor mq135
9.  #define pin_sensor_mq135 A1
10.
11. //Sensor DHT11
12. #include "DHT.h" //library sensor yang telah diim
    portkan
13. #define DHTPIN A2     //Pin apa yang digunakan
14. #define DHTTYPE DHT11   // DHT 11
15. DHT dht(DHTPIN, DHTTYPE);
16. const uint64_t pipe = 0xF0F0F0F0A2LL;
17.
18. //global variable untuk sensor Mq-7
19. #define pinSensor A0
20. long RL = 1000; // 1000 Ohm
21. long Ro = 830; // 830 ohm ( SILAHKAN DISESUAIKAN)

22. //
23. RF24 radio(9,10);
24.
25. void setup() {
26.   // put your setup code here, to run once:
27.
```

```
28.   Serial.begin(9600);            /*Setting baudra
    te of Serial Port to 9600*/
29.   dht.begin(); //prosedur memulai pembacaan modul
    e sensor
30.
31.   radio.begin();
32.
33.   radio.setPALevel(RF24_PA_MIN);
34.   radio.setChannel(0x76);
35.   radio.openWritingPipe(pipe);
36.   radio.enableDynamicPayloads();
37.   radio.powerUp();
38.
39. }
40.
41. void loop() {
42.   // put your main code here, to run repeatedly:

43.   //keterangan variable send_data -
    > nomor_node|ppmCOMq7|CelciusDHT11|HumidityDHT11|
    ppmMQ135
44.   char send_data[15] = "2|";
45.   char SaveValueMq7[10];
46.   char SaveValueMq135[10];
47.
48.   char SaveValueTempDHT11[10];
49.   char SaveValueHumidityDHT11[10];
50.   int SensorValueMq7 = getSensorMq7();
51.   int SensorValueTempDHT11 = getTempSensorDHT11()
    ;
52.   int SensorValueHumidityDHT11 = getHumiditySenso
    rDHT11();
53.   int SensorValueMq135 = getSensorMq135();
54.
55.   Serial.println(SensorValueMq135);
56.   Serial.println(SensorValueMq7);
57.   Serial.println(SensorValueTempDHT11);
58.
59.
60.   //convert int to char
61.   itoa(SensorValueMq7,SaveValueMq7,10);
62.   itoa(SensorValueMq135,SaveValueMq135,10);
```

```
63.   itoa(SensorValueTempDHT11,SaveValueTempDHT11,10
    );
64.   itoa(SensorValueHumidityDHT11,SaveValueHumidity
    DHT11,10);
65.
66.
67.   Serial.println(SaveValueMq7);
68.   Serial.print("Start Send data from node 2 : \n"
    );
69.   //concat char sensor mq7
70.   strcat(send_data,SaveValueMq7);
71.   strcat(send_data,"|");
72.   strcat(send_data,SaveValueTempDHT11);
73.   strcat(send_data,"|");
74.   strcat(send_data,SaveValueHumidityDHT11);
75.   strcat(send_data,"|");
76.   strcat(send_data,SaveValueMq135);
77.
78.
79.
80.   Serial.println(send_data);
81.   radio.write(&send_data, sizeof(send_data));
82.
83.
84.   delay(5000);
85. }
86.
87. int getSensorMq7(){
88.   int sensorvalue = analogRead(pinSensor); // mem
    baca nilai ADC dari sensor
89.   float VRL= sensorvalue*5.00/1024;  // mengubah
    nilai ADC ( 0 - 1023 ) menjadi nilai voltase ( 0
    - 5.00 volt )
90.
91.   float Rs = ( 5.00 * RL / VRL ) - RL;
92.
93.   float ppm = 100 * pow(Rs / Ro,-
    1.53); // ppm = 100 * ((rs/ro)^-1.53);
94.   Serial.print("CO : ");
95.   Serial.print(ppm);
96.   Serial.println(" ppm");
97.   return ppm;
```

```
98. //  Serial.println();
99. }
100.
101.       int getTempSensorDHT11(){
102.        // put your main code here, to run repeat
    edly:
103.
104.         //Pembacaan dalam format celcius (c)
105.         float celcius_1 = dht.readTemperature();

106.         //convert float to int agar bisa di assi
    gn array of char
107.         int t_celcius = celcius_1;
108.         //mengecek pembacaan apakah terjadi kega
    galan atau tidak
109.         if (isnan(celcius_1)) {
110.           Serial.println("Pembacaan data dari mo
    dule sensor gagal!");
111.         }
112.
113.         //pembacaan nilai pembacaan data suhu
114.         Serial.print("Suhu : ");
115.         Serial.print(celcius_1); //format deraja
    t celcius
116.         Serial.print(" 'C ");
117.         return(t_celcius);
118.        }
119.
120.       int getHumiditySensorDHT11(){
121.
122.         float humidity_1 = dht.readHumidity();
123.         int t_humidity = humidity_1;
124.         if (isnan(t_humidity)) {
125.           Serial.println("Pembacaan data dari mo
    dule sensor gagal!");
126.         }
127.
128.         return t_humidity;
129.        }
130.
131.       int getSensorMq135(){
132.         int sensorValue;
```

```
133.
134.        sensorValue = analogRead(pin_sensor_mq13
    5);      // read analog input pin 0
135.        Serial.print("Data Sensor Mq135=");
136.        Serial.print(sensorValue);
     // prints the value read
137.        Serial.println(" PPM");
138.
139.        return(sensorValue);
140.
141.        }
```

3. Kode Sumber Node Sensor 3

```
1.  #include <SPI.h>
2.  #include <nRF24L01.h>
3.  #include <printf.h>
4.  #include <RF24.h>
5.  #include <RF24_config.h>
6.  #include <stdio.h>
7.
8.  //sensor mq135
9.  #define pin_sensor_mq135 A1
10.
11. //Sensor DHT11
12. #include "DHT.h" //library sensor yang telah diim
    portkan
13. #define DHTPIN A2     //Pin apa yang digunakan
14. #define DHTTYPE DHT11   // DHT 11
15. DHT dht(DHTPIN, DHTTYPE);
16. const uint64_t pipe = 0xF0F0F0F0A3LL;
17.
18. //global variable untuk sensor Mq-7
19. #define pinSensor A0
20. long RL = 1000; // 1000 Ohm
21. long Ro = 830; // 830 ohm ( SILAHKAN DISESUAIKAN)

22. //
23. RF24 radio(9,10);
```

```
24.
25. void setup() {
26.   // put your setup code here, to run once:
27.
28.   Serial.begin(9600);            /*Setting baudra
    te of Serial Port to 9600*/
29.   dht.begin(); //prosedur memulai pembacaan modul
    e sensor
30.
31.   radio.begin();
32.
33.   radio.setPALevel(RF24_PA_MIN);
34.   radio.setChannel(0x76);
35.   radio.openWritingPipe(pipe);
36.   radio.enableDynamicPayloads();
37.   radio.powerUp();
38.
39. }
40.
41. void loop() {
42.   // put your main code here, to run repeatedly:

43.   //keterangan variable send_data -
    > nomor_node|ppmCOMq7|CelciusDHT11|HumidityDHT11|
    ppmMQ135
44.   char send_data[15] = "3|";
45.   char SaveValueMq7[10];
46.   char SaveValueMq135[10];
47.
48.   char SaveValueTempDHT11[10];
49.   char SaveValueHumidityDHT11[10];
50.   int SensorValueMq7 = getSensorMq7();
51.   int SensorValueTempDHT11 = getTempSensorDHT11()
    ;
52.   int SensorValueHumidityDHT11 = getHumiditySenso
    rDHT11();
53.   int SensorValueMq135 = getSensorMq135();
54.
55.   Serial.println(SensorValueMq135);
56.   Serial.println(SensorValueMq7);
57.   Serial.println(SensorValueTempDHT11);
58.
```

```
59.
60.   //convert int to char
61.   itoa(SensorValueMq7,SaveValueMq7,10);
62.   itoa(SensorValueMq135,SaveValueMq135,10);
63.   itoa(SensorValueTempDHT11,SaveValueTempDHT11,10
    );
64.   itoa(SensorValueHumidityDHT11,SaveValueHumidity
    DHT11,10);
65.
66.
67.   Serial.println(SaveValueMq7);
68.   Serial.print("Start Send data from node 3 : \n"
    );
69.   //concat char sensor mq7
70.   strcat(send_data,SaveValueMq7);
71.   strcat(send_data,"|");
72.   strcat(send_data,SaveValueTempDHT11);
73.   strcat(send_data,"|");
74.   strcat(send_data,SaveValueHumidityDHT11);
75.   strcat(send_data,"|");
76.   strcat(send_data,SaveValueMq135);
77.
78.
79.
80.   Serial.println(send_data);
81.   radio.write(&send_data, sizeof(send_data));
82.
83.
84.   delay(5000);
85. }
86.
87. int getSensorMq7(){
88.   int sensorvalue = analogRead(pinSensor); // mem
    baca nilai ADC dari sensor
89.   float VRL= sensorvalue*5.00/1024;  // mengubah
    nilai ADC ( 0 - 1023 ) menjadi nilai voltase ( 0
    - 5.00 volt )
90.
91.   float Rs = ( 5.00 * RL / VRL ) - RL;
92.
93.   float ppm = 100 * pow(Rs / Ro,-
    1.53); // ppm = 100 * ((rs/ro)^-1.53);
```

```
94.   Serial.print("CO : ");
95.   Serial.print(ppm);
96.   Serial.println(" ppm");
97.   return ppm;
98. //  Serial.println();
99. }
100.
101.       int getTempSensorDHT11(){
102.        // put your main code here, to run repeat
    edly:
103.
104.         //Pembacaan dalam format celcius (c)
105.         float celcius_1 = dht.readTemperature();

106.         //convert float to int agar bisa di assi
    gn array of char
107.         int t_celcius = celcius_1;
108.         //mengecek pembacaan apakah terjadi kega
    galan atau tidak
109.         if (isnan(celcius_1)) {
110.           Serial.println("Pembacaan data dari mo
    dule sensor gagal!");
111.         }
112.
113.         //pembacaan nilai pembacaan data suhu
114.         Serial.print("Suhu : ");
115.         Serial.print(celcius_1); //format deraja
    t celcius
116.         Serial.print(" 'C ");
117.         return(t_celcius);
118.        }
119.
120.       int getHumiditySensorDHT11(){
121.
122.         float humidity_1 = dht.readHumidity();
123.         int t_humidity = humidity_1;
124.         if (isnan(t_humidity)) {
125.           Serial.println("Pembacaan data dari mo
    dule sensor gagal!");
126.         }
127.
128.         return t_humidity;
```

```
129.         }
130.
131.       int getSensorMq135(){
132.         int sensorValue;
133.
134.         sensorValue = analogRead(1);       // re
    ad analog input pin 0
135.         Serial.print("Data Sensor Mq135=");
136.         Serial.print(sensorValue, DEC);
          // prints the value read
137.         Serial.println(" PPM");
138.
139.         return(sensorValue);
140.
141.         }
```

4. Kode Sumber Base Station

```
1.  import RPi.GPIO as GPIO
2.  from lib_nrf24 import NRF24
3.  import time
4.  import spidev
5.  import mysql.connector
6.
7.
8.  def insert_db(data):
9.      mydb = mysql.connector.connect(
10.       host="HOST",
11.       user="USERNAME",
12.       passwd="PASSWORD",
13.       database="DATABASE"
14.     )
15.
16.     mycursor = mydb.cursor()
17.
18.     sql = "INSERT INTO data_sensor (id_node,data_
    mq7,data_dht11_temperature,data_dht11_humidity,da
```

```
    ta_mq135,created_at) VALUES (%s, %s,%s,%s,%s,%s)"

    val = data
    #print(val)
    mycursor.execute(sql, val)
    mydb.commit()
    print(mycursor.rowcount, "record inserted.")


GPIO.setmode(GPIO.BCM)
pipes = [[0xF0, 0xF0, 0xF0, 0xF0, 0xA1], [0xF0, 0xF0, 0xF0, 0xF0, 0xA2], [0xF0, 0xF0, 0xF0, 0xF0, 0xA3]]


radio = NRF24(GPIO, spidev.SpiDev())
radio.begin(0, 17)

radio.setPayloadSize(32)
radio.setChannel(0x76)
radio.setDataRate(NRF24.BR_1MBPS)
radio.setPALevel(NRF24.PA_MIN)

radio.setAutoAck(True)
radio.enableDynamicPayloads()
radio.enableAckPayload()

radio.openReadingPipe(1, pipes[0])
radio.openReadingPipe(2, pipes[1])
radio.openReadingPipe(3, pipes[2])
radio.printDetails()

radio.startListening()
count = 0
while True:
    while not radio.available(0):
        #print("belum ada data yang masuk")
        time.sleep(1 / 5000)


```

```
56.
57.
58.     if (radio.available()):
59.         print("Data Node Sensor Masuk >>>>>")
60.         receivedMessage = []
61.         radio.read(receivedMessage, radio.getDyna
    micPayloadSize())
62.         print(radio.getDynamicPayloadSize())
63.         print("Received: {}".format(receivedMessa
    ge))
64.         print("Translating the receivedMessage in
    to unicode characters")
65.         string = ""
66.         for n in receivedMessage:
67.             # Decode into standard unicode set
68.             if (n >= 32 and n <= 126):
69.                 string += chr(n)
70.         print(string)
71.         data = string.split("|")
72.         datetime = time.strftime('%Y-%m-
    %d %H:%M:%S')
73.         data = list(map(int,data))
74.         if(data[0] == 1):
75.             print("Data dari node sensor 1")
76.         if(data[0] == 2):
77.             print("Data dari node sensor 2")
78.         if(data[0] == 3):
79.             print("Data dari node sensor 3")

80.         data.append(datetime)
81.         print("list integer : {}".format(data))
82.         insert_db(data)
83.
84.
85.         count = count + 1
86.         #print("received message decodes to : {}"
    .format(string))
87.         print("data ke {}".format(count))
88.         print(datetime)
89.         print("-->><<--")
90.
```

5. Kode Sumber SensorController

```
1.  <?php
2.
3.  namespace App\Http\Controllers;
4.  use Auth;
5.  use App\Sensor;
6.  use App\Rulebase;
7.  use App\Lokasi;
8.  use Illuminate\Http\Request;
9.  use DB;
10. use Session;
11. use Carbon\Carbon;
12.
13. class SensorController extends Controller
14. {
15.     public function index(){
16.         return view('admin.new_sensor');
17.     }
18.
19.     public function store(Request $request){
20.         $input = $request->all();
21.         $id_user = Auth::user()->id;
22.
23.         $sensor[]=[
24.             'id_user' => $id_user,
25.             'nama_sensor' => $input['nama_sensor'
    ],
26.             'deskripsi_sensor' => $input['deskrip
    si_sensor'],
27.             'temperature_sensor' => $input['tempe
    rature'],
28.             'humidity_sensor' => $input['humidity
    '],
29.             'gas_sensor'=> $input['gas'],
30.             'co_sensor' => $input['co'],
31.
32.         ];
33.         Sensor::insert($sensor);
```

```
34.         Session::flash('sukses','berhasil menamba
    h node sensor');
35.         return redirect('/home');
36.
37.         // dd($input);
38.     }
39.
40.     public function view(){
41.         $id_user = Auth::user()->id;
42.
43.         $data['status'] = DB::table('sensor')-
    >where('id_user',$id_user)->get();
44.         $data['sensor'] = DB::select( DB::raw("(S
    ELECT * FROM data_sensor WHERE id_node = 1 ORDER
    BY created_at DESC LIMIT 1 ) UNION (SELECT * FROM
     data_sensor WHERE id_node = 2 ORDER BY created_a
    t DESC LIMIT 1)UNION (SELECT * FROM data_sensor W
    HERE id_node = 3 ORDER BY created_at DESC LIMIT 1
    )") );
45.         $data['lokasi'] = DB::table('lokasi')-
    >where('id_user',$id_user)->get();
46.         return view('dashboard.dashboard',$data);

47.
48.     }
49.
50.     public function view_detail($id){
51.         $id_user = Auth::user()->id;
52.         $data['id_node'] = $id;
53.         $data['sensor'] = DB::table('sensor')-
    >where('id_user',$id_user)->where('id',$id)-
    >get();
54.         $data['status'] = DB::table('sensor')-
    >where('id_user',$id_user)->where('id',$id)-
    >get();
55.
56.         if($data['sensor']['0']-
    >temperature_sensor == 1){
57.             $data['sensor']['0']-
    >temperature_sensor = "aktif";
58.         }
59.         else{
```

```
60.             $data['sensor']['0']-
    >temperature_sensor = "tidak aktif";
61.         }
62.         if($data['sensor']['0']-
    >humidity_sensor == 1){
63.             $data['sensor']['0']-
    >humidity_sensor = "aktif";
64.         }
65.         else{
66.             $data['sensor']['0']-
    >humidity_sensor = "tidak aktif";
67.         }
68.         if($data['sensor']['0']-
    >gas_sensor == 1){
69.             $data['sensor']['0']-
    >gas_sensor = "aktif";
70.         }
71.         else{
72.             $data['sensor']['0']-
    >gas_sensor = "tidak aktif";
73.         }
74.         if($data['sensor']['0']-
    >co_sensor == 1){
75.             $data['sensor']['0']-
    >co_sensor = "aktif";
76.         }
77.         else{
78.             $data['sensor']['0']-
    >co_sensor = "tidak aktif";
79.         }
80.
81.
82.         $data['rulebase'] = DB::table('info_ruleb
    ase')->where('id_node',$id)-
    >orderBy('id','desc')->first();
83.         // dd($data['rulebase']->created_at);
84.         $data['rulebase']-
    >created_at = Carbon::parse($data['rulebase']-
    >created_at)->format('d-m-Y H:i:s');
85.
86.
87.         // dd($data['sensor']);
```

```
88.         return view('admin.detail_sensor',$data);

89.
90.
91.     }
92.     public function view_rulebase(){
93.         $id_user = Auth::user()->id;
94.         $data['rulebase'] = DB::table('rulebase')
    ->where('id_user',$id_user)->first();
95.         if( $data['rulebase']){
96.             return view('admin.edit_rulebase',$da
    ta);
97.         }
98.         else{
99.             return view('admin.rulebase');
100.                }
101.
102.
103.            }
104.
105.            public function rulebase_store(Request
     $request){
106.                $input = $request->all();
107.                //dd($input);
108.                $id_user = Auth::user()->id;
109.
110.                $rulebase_sensor[]=[
111.                    'id_user' => $id_user,
112.                    'ba_temperature' => $input['ba
    _temperature'],
113.                    'informasi_ba_temperature' =>
    $input['informasi_ba_temperature'],
114.                    'bb_temperature' => $input['bb
    _temperature'],
115.                    'informasi_bb_temperature' =>
    $input['informasi_bb_temperature'],
116.                    //humidity
117.                    'ba_humidity' => $input['ba_hu
    midity'],
118.                    'informasi_ba_humidity' => $in
    put['informasi_ba_humidity'],
```

```
119.                    'bb_humidity' => $input['bb_hu
    midity'],
120.                    'informasi_bb_humidity' => $in
    put['informasi_bb_humidity'],
121.                    //gas mq-135
122.                    'ba_gas' => $input['ba_gas'],

123.                    'informasi_ba_gas' => $input['
    informasi_ba_gas'],
124.                    'bb_gas' => $input['bb_gas'],

125.                    'informasi_bb_gas' => $input['
    informasi_bb_gas'],
126.                    //gas CO mq-7
127.                    'ba_co' => $input['ba_co'],
128.                    'informasi_ba_co' => $input['i
    nformasi_ba_co'],
129.                    'bb_co' => $input['bb_co'],
130.                    'informasi_bb_co' => $input['i
    nformasi_bb_co'],
131.                ];
132.                Rulebase::insert($rulebase_sensor)
    ;
133.                Session::flash('sukses','berhasil
    menAMBAH lokasi');
134.                return redirect('/home');
135.
136.            }
137.
138.            public function rulebase_edit(Request
    $request){
139.                $input = $request->all();
140.                $id_user = Auth::user()->id;
141.
142.                Rulebase::where('id_user',$id_user
    )
143.                ->update([
144.                    'ba_temperature' => $input['ba
    _temperature'],
145.                    'informasi_ba_temperature' =>
    $input['informasi_ba_temperature'],
```

```
146.                    'bb_temperature' => $input['bb
    _temperature'],
147.                    'informasi_bb_temperature' =>
    $input['informasi_bb_temperature'],
148.                    //humidity
149.                    'ba_humidity' => $input['ba_hu
    midity'],
150.                    'informasi_ba_humidity' => $in
    put['informasi_ba_humidity'],
151.                    'bb_humidity' => $input['bb_hu
    midity'],
152.                    'informasi_bb_humidity' => $in
    put['informasi_bb_humidity'],
153.                    //gas mq-135
154.                    'ba_gas' => $input['ba_gas'],

155.                    'informasi_ba_gas' => $input['
    informasi_ba_gas'],
156.                    'bb_gas' => $input['bb_gas'],

157.                    'informasi_bb_gas' => $input['
    informasi_bb_gas'],
158.                    //gas CO mq-7
159.                    'ba_co' => $input['ba_co'],
160.                    'informasi_ba_co' => $input['i
    nformasi_ba_co'],
161.                    'bb_co' => $input['bb_co'],
162.                    'informasi_bb_co' => $input['i
    nformasi_bb_co'],
163.                ]);
164.                Session::flash('sukses','berhasil
    mengedit rulebase');
165.                return redirect('/home');
166.            }
167.
168.            public function lokasi(){
169.                return view('admin.lokasi_basestat
    ion');
170.            }
171.
172.            public function add_lokasi(Request $re
    quest){
```

```
173.                $input = $request->all();
174.                // dd($input);
175.                $id_user = Auth::user()->id;
176.                $lokasi_data[]=[
177.                    'id_user' => $id_user,
178.                    'longitude' => $input['longitu
    de'],
179.                    'latitude' => $input['latitude
    '],
180.
181.                ];
182.                $cek_lokasi = DB::table('lokasi')-
    >where('id_user',$id_user)->first();
183.                // dd($cek_lokasi);
184.                if($cek_lokasi){
185.                    Lokasi::where('id_user',$id_us
    er)
186.                    -
    >update(['longitude' =>$input['longitude'],
187.                    'latitude' =>$input['latitude'
    ]]);
188.                    Session::flash('sukses','berha
    sil mengedit lokasi');
189.                }
190.                else{
191.                    Session::flash('sukses','berha
    sil menambah lokasi');
192.                    Lokasi::insert($lokasi_data);

193.
194.                }
195.
196.                return redirect('/home');
197.            }
198.
199.            public function submit_rangetime(Reque
    st $request){
200.                $input = $request->all();
201.                $id_user = Auth::user()->id;
202.                $id = $input['node'];
203.
```

```
204.                $data['id_node'] = $input['node'];
205.                $data['start'] = $input['start'];
206.                $data['end'] = $input['end'];
207.
208.                $data['start'] = str_replace("T"," ",$input['start']);
209.                $data['end'] = str_replace("T"," ",$input['end']);
210.                $data['status'] = DB::table('sensor')->where('id_user',$id_user)->where('id',$id)->get();
211.
212.                // dd($data);
213.                return view('admin.detail_sensor_date',$data);
214.            }
215.
216.        }
```

6. Kode Sumber ChartsController

```
1.  <?php
2.
3.  namespace App\Http\Controllers;
4.
5.  use Illuminate\Http\Request;
6.  use App\DataSensor;
7.  use DB;
8.  use Carbon\Carbon;
9.  use DateTime;
10. use Auth;
11. use App\Sensor;
12. class ChartsApiController extends Controller
13. {
14.
15.     public function get_temperature($id)
16.     {
```

```
17.         $data_sensor = DB::table('data_sensor')-
    >where('id_node',$id)->limit(30)-
    >orderBy('created_at','desc')->get()-
    >reverse();
18.         $labels = $data_sensor-
    >pluck('created_at');
19.         $data = $data_sensor-
    >pluck('data_dht11_temperature');
20.
21.         //change format date time
22.         $i = 0;
23.         foreach($labels as $label){
24.            $label_change = Carbon::parse($label)-
    >format('H:i:s d-m-Y');
25.            $labels[$i] = $label_change;
26.            //dd($label_change);
27.             $i++;
28.         }
29.         //  dd($labels);
30.         return response()-
    >json(compact('labels', 'data'));
31.     }
32.
33.     public function get_humidity($id)
34.     {
35.         $data_sensor = DB::table('data_sensor')-
    >where('id_node',$id)->limit(30)-
    >orderBy('created_at','desc')->get()-
    >reverse();
36.         $labels = $data_sensor-
    >pluck('created_at');
37.         $data = $data_sensor-
    >pluck('data_dht11_humidity');
38.
39.         //change format date time
40.         $i = 0;
41.         foreach($labels as $label){
42.            $label_change = Carbon::parse($label)-
    >format('H:i:s d-m-Y');
43.            $labels[$i] = $label_change;
44.            //dd($label_change);
45.             $i++;
```

```
46.         }
47.         //  dd($labels);
48.         return response()-
    >json(compact('labels', 'data'));
49.     }
50.
51.     public function get_mq7($id)
52.     {
53.         $data_sensor = DB::table('data_sensor')-
    >where('id_node',$id)->limit(30)-
    >orderBy('created_at','desc')->get()-
    >reverse();
54.         $labels = $data_sensor-
    >pluck('created_at');
55.         $data = $data_sensor-
    >pluck('data_mq7');
56.
57.         //change format date time
58.         $i = 0;
59.         foreach($labels as $label){
60.
61.            $label_change = Carbon::parse($label)-
    >format('H:i:s d-m-Y');
62.            $labels[$i] = $label_change;
63.            //dd($label_change);
64.             $i++;
65.         }
66.         //  dd($labels);
67.         return response()-
    >json(compact('labels', 'data'));
68.     }
69.
70.     public function get_mq135($id)
71.     {
72.         $data_sensor = DB::table('data_sensor')-
    >where('id_node',$id)->limit(30)-
    >orderBy('created_at','desc')->get()-
    >reverse();
73.         $labels = $data_sensor-
    >pluck('created_at');
74.         $data = $data_sensor-
    >pluck('data_mq135');
```

```
75.
76.         //change format date time
77.         $i = 0;
78.         foreach($labels as $label){
79.
80.            $label_change = Carbon::parse($label)-
    >format('H:i:s d-m-Y');
81.            $labels[$i] = $label_change;
82.            //dd($label_change);
83.             $i++;
84.         }
85.         //  dd($labels);
86.         return response()-
    >json(compact('labels', 'data'));
87.     }
88.
89.     public function get_temperature_time(Request
    $request){
90.         $input = $request->all();
91.         $id = $input['id_node'];
92.         $id_user = Auth::user()->id;
93.
94.         $data_sensor = DataSensor::where('id_node
    ',$id)
95.                                 -
    >where('created_at', '>=', $input['start'])
96.                                 -
    >where('created_at', '<=', $input['end'])
97.                                 -
    >orderBy('created_at','desc')->get()-
    >reverse();
98.         $labels = $data_sensor-
    >pluck('created_at');
99.         $data = $data_sensor-
    >pluck('data_dht11_temperature');
100.
101.                //change format date time
102.                $i = 0;
103.                foreach($labels as $label){
104.
105.                    $label_change = Carbon::parse(
    $label)->format('H:i:s d-m-Y');
```

```
106.                    $labels[$i] = $label_change;
107.                    //dd($label_change);
108.                    $i++;
109.                }
110.
111.                return response()->json(compact('labels', 'data'));
112.            }
113.
114.            public function get_humidity_time(Request $request){
115.                $input = $request->all();
116.                $id = $input['id_node'];
117.                $id_user = Auth::user()->id;
118.
119.                $data_sensor = DataSensor::where('id_node',$id)
120.                                                    ->where('created_at', '>=', $input['start'])
121.                                                    ->where('created_at', '<=', $input['end'])
122.                                                    ->orderBy('created_at','desc')->get()->reverse();
123.                $labels = $data_sensor->pluck('created_at');
124.                $data = $data_sensor->pluck('data_dht11_humidity');
125.
126.                //change format date time
127.                $i = 0;
128.                foreach($labels as $label){
129.        S                $label_change = Carbon::parse($label)->format('H:i:s d-m-Y');
130.                    $labels[$i] = $label_change;
131.                    //dd($label_change);
132.                    $i++;
133.                }
134.
135.                return response()->json(compact('labels', 'data'));
136.            }
```

```
137.
138.            public function get_gas_time(Request $
    request){
139.                $input = $request->all();
140.                $id = $input['id_node'];
141.                $id_user = Auth::user()->id;
142.
143.                $data_sensor = DataSensor::where('
    id_node',$id)
144.                                    -
    >where('created_at', '>=', $input['start'])
145.                                    -
    >where('created_at', '<=', $input['end'])
146.                                    -
    >orderBy('created_at','desc')->get()-
    >reverse();
147.                $labels = $data_sensor-
    >pluck('created_at');
148.                $data = $data_sensor-
    >pluck('data_mq135');
149.
150.                //change format date time
151.                $i = 0;
152.                foreach($labels as $label){
153.
154.                    $label_change = Carbon::parse(
    $label)->format('H:i:s d-m-Y');
155.                    $labels[$i] = $label_change;
156.                    //dd($label_change);
157.                    $i++;
158.                }
159.
160.                return response()-
    >json(compact('labels', 'data'));
161.            }
162.
163.            public function get_co_time(Request $r
    equest){
164.                $input = $request->all();
165.                $id = $input['id_node'];
166.                $id_user = Auth::user()->id;
167.
```

```
168.                $data_sensor = DataSensor::where('
    id_node',$id)
169.                                    -
    >where('created_at', '>=', $input['start'])
170.                                    -
    >where('created_at', '<=', $input['end'])
171.                                    -
    >orderBy('created_at','desc')->get()-
    >reverse();
172.                $labels = $data_sensor-
    >pluck('created_at');
173.                $data = $data_sensor-
    >pluck('data_mq7');
174.
175.                //change format date time
176.                $i = 0;
177.                foreach($labels as $label){
178.
179.                    $label_change = Carbon::parse(
    $label)->format('H:i:s d-m-Y');
180.                    $labels[$i] = $label_change;
181.                    //dd($label_change);
182.                    $i++;
183.                }
184.
185.                return response()-
    >json(compact('labels', 'data'));
186.            }
187.        }
```

7. Kode Sumber Prosedur UpdateRulebase

```
1.  BEGIN
2.
3.  DECLARE ba_co INT DEFAULT 0;
4.  DECLARE bb_co INT DEFAULT 0;
5.  DECLARE ba_temperature INT DEFAULT 0;
6.  DECLARE bb_temperature INT DEFAULT 0;
```

```
7.  DECLARE ba_humidity INT DEFAULT 0;
8.  DECLARE bb_humidity INT DEFAULT 0;
9.  DECLARE ba_gas INT DEFAULT 0;
10. DECLARE bb_gas INT DEFAULT 0;
11. DECLARE jumlah_data_ba_temperature INT DEFAULT 0;

12. DECLARE jumlah_data_bb_temperature INT DEFAULT 0;

13. DECLARE jumlah_data_ba_humidity INT DEFAULT 0;
14. DECLARE jumlah_data_bb_humidity INT DEFAULT 0;
15. DECLARE jumlah_data_ba_gas INT DEFAULT 0;
16. DECLARE jumlah_data_bb_gas INT DEFAULT 0;
17. DECLARE jumlah_data_ba_co INT DEFAULT 0;
18. DECLARE jumlah_data_bb_co INT DEFAULT 0;
19.
20.
21. DECLARE info_temperature VARCHAR(40);
22. DECLARE info_humidity VARCHAR(40);
23. DECLARE info_gas VARCHAR(40);
24. DECLARE info_co VARCHAR(40);
25. DECLARE date_time_now VARCHAR(40);
26.
27. SET date_time_now = (SELECT NOW());
28.
29. SET ba_temperature = (SELECT ba_temperature FROM
    rulebase order by id desc LIMIT 1);
30. SET bb_temperature = (SELECT bb_temperature FROM
    rulebase ORDER BY id DESC LIMIT 1);
31. SET ba_humidity = (SELECT ba_humidity FROM ruleba
    se ORDER BY id DESC LIMIT 1);
32. SET bb_humidity = (SELECT bb_humidity FROM ruleba
    se ORDER BY id DESC LIMIT 1);
33. SET ba_gas = (SELECT ba_gas FROM rulebase ORDER B
    Y id DESC LIMIT 1);
34. SET bb_gas = (SELECT bb_gas FROM rulebase ORDER B
    Y id DESC LIMIT 1);
35. SET ba_co = (SELECT ba_co FROM rulebase ORDER BY
    id DESC LIMIT 1);
36. SET bb_co = (SELECT bb_co FROM rulebase ORDER BY
    id DESC LIMIT 1);
37.
38.
```

```
39.
40. set jumlah_data_ba_temperature = (SELECT COUNT(*)
     FROM (SELECT * from data_sensor WHERE id_node =
    id_node ORDER BY created_at DESC LIMIT 50) AS d w
    here d.data_dht11_temperature > ba_temperature);

41. set jumlah_data_bb_temperature = (SELECT COUNT(*)
     FROM (SELECT * from data_sensor WHERE id_node =
    id_node ORDER BY created_at DESC LIMIT 50) AS d w
    here d.data_dht11_temperature < bb_temperature);

42.
43. IF jumlah_data_ba_temperature > jumlah_data_bb_te
    mperature THEN
44.     set info_temperature = (SELECT informasi_ba_t
    emperature FROM rulebase ORDER BY id DESC LIMIT 1
    );
45. ELSEIF jumlah_data_ba_temperature < jumlah_data_b
    b_temperature THEN
46.     set info_temperature = (SELECT informasi_bb_t
    emperature FROM rulebase ORDER BY id DESC LIMIT 1
    );
47. ELSE
48.     set info_temperature = 'normal';
49. END IF;
50.
51.
52. set jumlah_data_ba_humidity = (SELECT COUNT(*) FR
    OM (SELECT * from data_sensor WHERE id_node = id_
    node ORDER BY created_at DESC LIMIT 50) AS d wher
    e d.data_dht11_humidity > ba_humidity);
53. set jumlah_data_bb_humidity = (SELECT COUNT(*) FR
    OM (SELECT * from data_sensor WHERE id_node = id_
    node ORDER BY created_at DESC LIMIT 50) AS d wher
    e d.data_dht11_humidity < bb_humidity);
54.
55. IF jumlah_data_ba_humidity > jumlah_data_bb_humid
    ity THEN
56.     set info_humidity = (SELECT informasi_ba_humi
    dity FROM rulebase ORDER BY id DESC LIMIT 1);
57. ELSEIF jumlah_data_ba_humidity < jumlah_data_bb_h
    umidity THEN
```

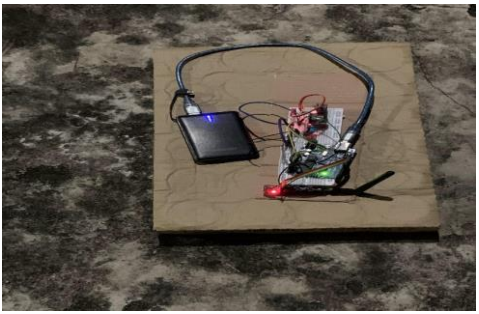
```
58.     set info_humidity = (SELECT informasi_bb_humi
    dity FROM rulebase ORDER BY id DESC LIMIT 1);
59. ELSE
60.     set info_humidity = 'normal';
61. END IF;
62.
63.
64. set jumlah_data_ba_gas = (SELECT COUNT(*) FROM (S
    ELECT * from data_sensor WHERE id_node = id_node
    ORDER BY created_at DESC LIMIT 50) AS d where d.d
    ata_mq135 > ba_gas);
65. set jumlah_data_bb_gas = (SELECT COUNT(*) FROM (S
    ELECT * from data_sensor WHERE id_node = id_node
    ORDER BY created_at DESC LIMIT 50) AS d where d.d
    ata_mq135 < bb_gas);
66.
67. IF jumlah_data_ba_gas > jumlah_data_bb_gas THEN
68.     set info_gas = (SELECT informasi_ba_gas FROM
    rulebase ORDER BY id DESC LIMIT 1);
69. ELSEIF jumlah_data_ba_gas < jumlah_data_bb_gas TH
    EN
70.     set info_gas = (SELECT informasi_bb_gas FROM
    rulebase ORDER BY id DESC LIMIT 1);
71. ELSE
72.     set info_gas = 'normal';
73. END IF;
74.
75. set jumlah_data_ba_co = (SELECT COUNT(*) FROM (SE
    LECT * from data_sensor WHERE id_node = id_node O
    RDER BY created_at DESC LIMIT 50) AS d where d.da
    ta_mq7 > ba_co);
76. set jumlah_data_bb_co = (SELECT COUNT(*) FROM (SE
    LECT * from data_sensor WHERE id_node = id_node O
    RDER BY created_at DESC LIMIT 50) AS d where d.da
    ta_mq7 < bb_co);
77.
78. IF jumlah_data_ba_co > jumlah_data_bb_co THEN
79.     set info_co = (SELECT informasi_ba_co FROM ru
    lebase ORDER BY id DESC LIMIT 1);
80. ELSEIF jumlah_data_ba_co < jumlah_data_bb_co THEN
```

```
81.     set info_co = (SELECT informasi_bb_co FROM ru
    lebase ORDER BY id DESC LIMIT 1);
82. ELSE
83.     set info_co = 'normal';
84. END IF;
85.
86.
87. INSERT INTO info_rulebase (id_node,info_rulebase_
    temperature,info_rulebase_humidity,info_rulebase_
    gas,info_rulebase_co,created_at)
88. VALUES (id_node,info_temperature,info_humidity,in
    fo_gas,info_co,date_time_now);
89.
90.
91. SELECT concat('info temperature is ',info_tempera
    ture);
92. SELECT concat('info humidity is ',info_humidity);

93. SELECT concat('info gas is ',info_gas);
94. SELECT concat('info co is ',info_co);
95.
96. END
```

LAMPIRAN C

DOKUMENTASI IMPLEMENTASI



(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Ubut Eka Putra lahir di Palembang Pada tanggal 31 Maret 1998. Penulis menempuh Pendidikan formal di SDN 08 Kota Bengkulu (2004-2010), SMPN 02 Kota Bengkulu (2010-2013), SMAN 05 Kota Bengkulu (2013-2016), dan Teknik Informatika ITS Surabaya (2016-2020). Bidang studi yang diambil oleh penulis saat berkuliah di Departemen Teknik Informatika ITS adalah Komputasi Berbasis Jaringan (KBJ). Selama menempuh perkuliahan, penulis juga pernah menjadi asisten dosen pada mata kuliah jaringan komputer dan sistem operasi. Selama berkuliah penulis juga memiliki ketertarikan pada bidang infrastruktur jaringan , komputasi awan, dan jaringan nirkabel. Penulis aktif dalam organisasi Badan Eksekutif Mahasiswa Fakultas Teknologi Informasi dan Komunikasi (2017-2018), dan Keluarga Muslim Informatika (2017-2019). Penulis juga aktif dalam kegiatan kepanitiaan seperti SCHEMATICS 2017 – 2018 sebagai staff, SCHEMATICS 2018 - 2019 sebagai Badan Pengurus Harian. Penulis juga pernah terlibat dalam proyek PPDB Provinsi Jawa Timur Tahun 2018-2020. Selama berkuliah, penulis juga menjadi administrator di Laboratorium Komputasi Berbasis Jaringan. Penulis dapat dihubungi melalui nomor *handphone* 087746283161 atau di email ubut31@gmail.com.