TUGAS AKHIR – KI141502

RANCANG BANGUN MONITORING SERVER DAN GATEWAY ADAPTIF UNTUK CLUSTER JARINGAN SENSOR NIRKABEL TERSEBAR

MUHAMMAD DIVI JAYA NURYANTO
NRP 5113100066

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

Dosen Pembimbing II
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

RANCANG BANGUN MONITORING SERVER DAN GATEWAY ADAPTIF UNTUK CLUSTER JARINGAN SENSOR NIRKABEL TERSEBAR

MUHAMMAD DIVI JAYA NURYANTO
NRP 5113100066

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

Dosen Pembimbing II
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

UNDERGRADUATE THESES – KI141502

DESIGN AND DEVELOPMENT OF MONITORING SERVER AND ADAPTIVE GATEWAY FOR DISTRIBUTED WIRELESS SENSOR NETWORK CLUSTER

MUHAMMAD DIVI JAYA NURYANTO
NRP 5113100066

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

Second Advisor
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D

Department of Informatics
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2017

LEMBAR PENGESAHAN

RANCANG BANGUN MONITORING SERVER DAN GATEWAY ADAPTIF UNTUK CLUSTER JARINGAN SENSOR NIRKABEL TERSEBAR

TUGAS AKHIR

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

Oleh:
MUHAMMAD DIVI JAYA NURYANTO
NRP : 5113100066

Disetujui oleh Pembimbing Tugas Akhir:

1. Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
   NIP: 19741022 200003 1 001 ..... (Pembimbing 1)
2. Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.
   NIP: 19770824 200604 1 001 ..... (Pembimbing 2)

SURABAYA
JUNI, 2017

RANCANG BANGUN MONITORING SERVER DAN GATEWAY ADAPTIF UNTUK CLUSTER JARINGAN SENSOR NIRKABEL TERSEBAR

Nama Mahasiswa	**: MUHAMMAD DIVI JAYA NURYANTO**
NRP	**: 5113100066**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Dosen Pembimbing 2	**: Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.**

ABSTRAK

Jaringan Sensor Nirkabel merupakan jaringan yang terbentuk dari beberapa node sensor yang terhubung pada sebuah gateway. Gateway berperan untuk menjadi perantara antara Node – node Sensor dengan server melalui sambungan internet. Saat melakukan pengambilan data menggunakan node sensor di daerah terpencil, kendala yang sering muncul adalah kegagalan pengiriman data oleh gateway karena koneksi internet yang kurang stabil. Selain itu, jumlah node sensor yang banyak menyebabkan relatif sulitnya dalam melakukan pengelolaan data.

Oleh karena itu, dalam Tugas Akhir ini dirancang Monitoring Server yang berguna untuk mempermudah proses cluster atau pengelompokkan jaringan sensor nirkabel tersebar. Dan juga dikembangkan sebuah Gateway Adaptif yang mampu menentukan proses pengiriman data ke server berdasarkan ketersediaan koneksi internet. Gateway Adaptif ini dapat

menentukan kapan pengiriman harus dilakukan melalui internet dan kapan pengiriman data harus dilakukan melalui sms. Jenis komunikasi yang digunakan antara node sensor dengan gateway adalah Bluetooth.

Sistem yang telah dibangun dalam Tugas Akhir ini meliputi Sensor Cluster Management, Android Gateway, SMS Receiver dan Node Sensor. Sensor Cluster Management merupakan sebuah Monitoring Server yang menyediakan layanan Virtualisasi Sensor untuk pengelolaan data dan pengelompokkan sensor – sensor tersebar. Android Gateway adalah sebuah Gateway Adaptif yang dibangun menggunakan ponsel cerdas berbasis Android, yang dapat berperan adaptif dengan mengirimkan data dalam bentuk sms jika koneksi internet tidak tersedia. SMS Receiver berguna untuk menerima data sms yang masuk dan kemudian mengolahnya untuk diteruskan ke server. Node Sensor dibuat menggunakan mikrokontroler Arduino dan dipasangi sensor, Node Sensor dapat terhubung dengan Android Gateway menggunakan komunikasi Bluetooth.

Berdasarkan uji coba, Sensor Cluster Management dapat bekerja dengan baik. Android Gateway dapat berperan adaptif dalam menentukan cara pengiriman data ke server berdasarkan ketersediaan koneksi internet dan dapat terhubung dengan Node Sensor menggunakan Bluetooth. Rata - rata waktu Bluetooth untuk terhubung adalah 3,072 detik dengan akurasi pengiriman data sebesar 92,62% (≤1 m), 87,62% (±5 m) dan 76,43% (±10 m). Pengiriman data menggunakan sms lebih lambat 7,218 detik dibandingkan menggunakan internet. Daya yang digunakan Node Sensor yaitu 776,16 mAh / jam (jeda 5 detik) dan 322,08 mAh / jam (jeda 10 detik). Penggunaan sms dapat menghemat daya Android Gateway sebesar 9% (450 mAh) dalam 3 jam.

Kata kunci: Jaringan Sensor Nirkabel, Sensor Cluster Management, Gateway Adaptif, Android, Arduino, Bluetooth.

DESIGN AND DEVELOPMENT OF MONITORING SERVER AND ADAPTIVE GATEWAY FOR DISTRIBUTED WIRELESS SENSOR NETWORK CLUSTER

Student's Name	**: MUHAMMAD DIVI JAYA NURYANTO**
Student's ID	**: 5113100066**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Second Advisor	**: Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.**

ABSTRACT

Wireless Sensor Network is a network formed from multiple sensor nodes connected to a gateway. The gateway plays an intermediary between the Sensor Nodes and the server through an internet connection. When performing data retrieval using sensor nodes in remote areas, the most common constraint is the failure of data transmission by the gateway due to the less stable internet connection. In addition, the number of sensor nodes that many cause relatively difficult in performing data management.

Therefore, in this Undergraduate Theses designed Monitoring Server that is useful to facilitate the process of clustering or grouping of wireless sensor networks spread. And also developed an Adaptive Gateway that is able to determine the process of sending data to the server based on the availability of internet connection. This Adaptive Gateway can specify when

delivery should be done via the internet and when data delivery should be done via sms. The type of communication used between the sensor nodes and the gateway is Bluetooth.

The system that has been built in this Undergraduate Theses includes Sensor Cluster Management, Android Gateway, SMS Receiver and Node Sensor. Cluster Management Sensor is a Monitoring Server that provides Sensor Virtualization services for data management and grouping of dispersed sensors. Android Gateway is an Adaptive Gateway built using Android-based smartphones, which can play an adaptive role by sending data in sms if Internet connection is not available. SMS Receiver is useful to receive incoming sms data and then process it to be forwarded to the server. Node Sensors created using Arduino microcontroller and fitted sensors, Node Sensor can connect with Android Gateway using Bluetooth communication.

Based on testing, the Sensor Cluster Management can work well. Android Gateway can play an adaptive role in determining how to send data to server based on internet connection availability and can connect with Node Sensor using Bluetooth. The average Bluetooth time to connect is 3.072 seconds with the data delivery accuracy of 92.62% (≤1 m), 87.62% (± 5 m) and 76.43% (± 10 m). Data transmission using sms is 7.218 seconds slower than using internet. Power used by Node Sensor is 776.16 mAh / hour (pause every 5 seconds) and 322.08 mAh / hour (pause every 10 seconds). The use of sms can save Android Gateway power by 9% (450 mAh) in 3 hours.

Keywords : Wireless Sensor Network, Sensor Cluster Management, Adaptive Gateway, Android, Arduino, Bluetooth.

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah Swt yang telah melimpahkan rahmat dan hidayah-Nya dan junjungan Nabi Muhammad SAW sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"**RANCANG BANGUN MONITORING SERVER DAN GATEWAY ADAPTIF UNTUK CLUSTER JARINGAN SENSOR NIRKABEL TERSEBAR**"

yang merupakan salah satu syarat dalam menempuh ujian sidang guna memperoleh gelar Sarjana Komputer. Selesainya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Bapak Sugiyanto (Almarhum) dan Ibu Eny Djulianingsih selaku orang tua penulis yang selalu memberikan dukungan doa, moral, dan material yang tak terhingga kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Bapak Nurfalakhi dan Ibu Himyatul Amanah selaku wali di Surabaya yang telah mendidik dan memberikan naungan selama masa kuliah hingga pengerjaan Tugas Akhir ini.
3. Bapak Waskitho Wibisono, S.Kom,. M.Eng., Ph.D selaku pembimbing I sekaligus dosen wali yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
4. Bapak Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D. selaku II yang telah membimbing dan memberikan

motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.

5. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala jurusan Teknik Informatika ITS.
6. Bapak Dr. Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator S1 di Jurusan Teknik Informatika ITS.
7. Seluruh dosen dan karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di ITS.
8. Ibu Eva Mursidah dan Ibu Sri Budiati yang selalu mempermudah penulis dalam peminjaman buku di RBTC.
9. Dianita Rosa dan Aisyah Rahma selaku saudara yang selalu mendukung dalam pengerjaan Tugas Akhir ini.
10. Annisa Arum dan Cita Mutiara selaku anak dari Bapak Nur yang sudah dianggap seperti kakak bagi penulis yang senantiasa memberi dukungan kepada penulis.
11. Teman-teman Keluarga Muslim Informatika, yang sudah banyak meluruskan penulis.
12. Teman-teman seperjuangan RMK NCC/KBJ, yang telah menemani dan menyemangati penulis.
13. Teman-teman administrator NCC/KBJ, yang telah menemani dan menyemangati penulis selama penulis menjadi administrator, menjadi rumah kedua penulis selama penulis berkuliah.
14. Teman-teman angkatan 2013, yang sudah mendukung saya selama perkuliahan.
15. Sahabat penulis yang tidak dapat disebutkan satu per satu yang selalu membantu, menghibur, menjadi tempat bertukar ilmu dan berjuang bersama-sama penulis.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan sehingga dengan kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Juni 2017

DAFTAR ISI

LEMBAR PENGESAHAN ..... **v**
ABSTRAK ..... **vii**
ABSTRACT ..... **ix**
KATA PENGANTAR ..... **xi**
DAFTAR ISI ..... **xiii**
DAFTAR GAMBAR ..... **xxi**
DAFTAR TABEL ..... **xxv**
DAFTAR KODE SUMBER ..... **xxix**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Permasalahan 3
1.4 Tujuan 4
1.5 Manfaat 4
1.6 Metodologi 5
1.6.1 Penyusunan Proposal 5
1.6.2 Studi Literatur 5
1.6.3 Analisis dan Desain Perangkat Lunak 6
1.6.4 Implementasi Perangkat Lunak 6
1.6.5 Pengujian dan Evaluasi 7
1.6.6 Penyusunan Buku 7
1.7 Sistematika Penulisan Laporan 7
BAB II TINJAUAN PUSTAKA **9**
2.1 Jaringan Sensor Nirkabel 9
2.2 Sensor Cloud 10
2.3 Visualisasi Data 10
2.4 *Gateway* Adaptif 11
2.5 Mikrokontroler Arduino UNO 12
2.6 Bluetooth 14
2.7 Piconet 14
2.8 Variabel Ketersediaan Koneksi Internet 15
2.9 Modul HC-06 16
2.10 Modul LM35 17

2.11 Modul DHT11 18
2.12 Modul Sensor Cahaya (LM393) 18
2.13 Modul Sensor Suara (KY-038) 19
2.14 Android 19
2.15 CodeIgniter 20
2.16 MySQL 20
2.17 Bootstrap 21
2.18 SMS *Gateway* 21
2.19 Gammu 22
2.20 Python 22
2.21 Highcharts 22
2.22 Volley *Library* (Android) 23
2.23 SmsManager *Library* (Android) 23
2.24 Network Connection Class *Library* (Android) 24
2.25 MySQLdb *Library* (Python) 24
2.26 Urllib2 *Library* (Python) 24
BAB III PERANCANGAN PERANGKAT LUNAK 25
3.1 Deskripsi Umum Sistem 25
3.2 Arsitektur Umum Sistem 26
3.3 Perancangan *Sensor Cluster Management* (Server) 28
3.3.1 Perancangan Diagram Kasus Penggunaan 29
3.3.1.1 Menampilkan Sensor Berdasarkan Kelompok 30
3.3.1.2 Mendaftarkan Kelompok Baru 30
3.3.1.3 Mendaftarkan Sensor Baru 30
3.3.1.4 Menampilkan Peta Lokasi Keseluruhan 31
3.3.1.5 Menampilkan Peta Lokasi Khusus 31
3.3.1.6 Menampilkan Detail Sensor 32
3.3.2 Perancangan Basis Data (*Sensor Cluster Management*) 32
3.3.2.1 Tabel *users* 33
3.3.2.2 Tabel *sensor_group* 34
3.3.2.3 Tabel *sensor* 35
3.3.2.4 Tabel *sensor_data* 38
3.3.3 Perancangan Antarmuka Pengguna (*Sensor Cluster Management*) 40

3.3.3.1 Halaman Utama 41
3.3.3.2 Halaman *Form New Group* 42
3.3.3.3 Halaman *Form New Sensor* 42
3.3.3.4 Halaman Peta Lokasi (*Maps*) 43
3.3.3.5 Halaman Detail Sensor 45
3.4 Perancangan Android *Gateway* 46
3.4.1 Perancangan Diagram Alir (Android *Gateway*) 47
3.4.1.1 Diagram Alir Subproses *handleMessage* 48
3.4.1.2 Diagram Alir Subproses *NormalTask* 50
3.4.1.3 Diagram Alir Subproses *StrictTask* 53
3.4.1.4 Diagram Alir Subproses *OnLocationChanged* 54
3.4.1.5 Diagram Alir Subproses *InternetAvailability* .. 55
3.4.2 Perancangan Antarmuka Pengguna (Android *Gateway*) 56
3.5 Perancangan SMS *Receiver* 57
3.5.1 Perancangan Diagram Alir (SMS *Receiver*) 58
3.5.2 Perancangan Basis Data (SMS *Receiver*) 60
3.5.3 Perancangan Susunan (*Format*) SMS 62
3.6 Perancangan *Node* Sensor 64
3.6.1 Perancangan Rangakaian Utama 64
3.6.2 Perancangan Diagram Alir (*Node* Sensor) 67
3.6.2.1 Diagram Alir Subproses *setup* 68
3.6.2.2 Diagram Alir Subproses *updateSensor* 69
BAB IV IMPLEMENTASI 71
4.1 Lingkungan Implementasi 71
4.1.1 Lingkungan Implementasi Perangkat Keras 71
4.1.2 Lingkungan Implementasi Perangkat Lunak 73
4.2 Implementasi *Sensor Cluster Management* 74
4.2.1 Implementasi Kasus Kegunaan 74
4.2.1.1 Menampilkan Sensor Berdasarkan Kelompok 75
4.2.1.2 Mendaftarkan Kelompok Baru 76
4.2.1.3 Mendaftarkan Sensor Baru 78
4.2.1.4 Menampilkan Peta Lokasi Keseluruhan 79
4.2.1.5 Menampilkan Peta Lokasi Khusus 80
4.2.1.6 Menampilkan Detail Sensor 82

4.2.2 Implementasi Fungsi Lain (API) 83
4.2.2.1 Fungsi Penerima Data Sensor 84
4.2.2.2 Fungsi Memperbarui Lokasi 86
4.2.3 Implementasi Antarmuka Pengguna *Sensor Cluster Management* 87
4.2.3.1 Halaman Utama 87
4.2.3.2 Halaman *Form New Group* 88
4.2.3.3 Halaman *Form New Group* 89
4.2.3.4 Halaman Peta Lokasi 89
4.2.3.5 Halaman Detail Sensor 91
4.3 Implementasi Android *Gateway* 93
4.3.1 Implementasi Fungsi dan Kelas (Android *Gateway*) 93
4.3.1.1 Fungsi *onCreate* 93
4.3.1.2 Fungsi *onResume* 95
4.3.1.3 Fungsi *handleMessage* 96
4.3.1.4 Kelas *NormalTask* 98
4.3.1.5 Kelas *StrictTask* 101
4.3.1.6 Fungsi *onLocationChanged* 102
4.3.1.7 Fungsi *InternetAvailability* 103
4.3.1.8 Fungsi *onSensorChanged* 103
4.3.2 Implementasi Antarmuka Pengguna (Android *Gateway*) 104
4.4 Implementasi SMS *Receiver* 105
4.4.1 Implementasi Konfigurasi *File* 105
4.4.1.1 Konfigurasi *File gammurc* 106
4.4.1.2 Konfigurasi *File smsdrc* 106
4.4.2 Implementasi Fungsi (SMS *Receiver*) 107
4.4.2.1 Fungsi *main* 107
4.4.2.2 Fungsi *sendSensorData* 108
4.5 Implementasi *Node* Sensor 109
4.5.1 Implementasi Rangkaian Utama 109
4.5.2 Implementasi Fungsi (*Node* Sensor) 111
4.5.2.1 Fungsi *setup* 111
4.5.2.2 Fungsi *loop* 112

4.5.2.3 Fungsi *updateSensor* 113
BAB V HASIL UJI COBA DAN EVALUASI 115
5.1 Lingkungan Uji Coba 115
5.2 Skenario Uji Coba Fungsionalitas 116
5.2.1 Skenario Uji Coba (UJ-F01) - Menampilkan Sensor Berdasarkan Kelompok 116
5.2.2 Skenario Uji Coba (UJ-F02) - Mendaftarkan Kelompok Baru 117
5.2.3 Skenario Uji Coba (UJ-F03) - Mendaftarkan Sensor Baru 118
5.2.4 Skenario Uji Coba (UJ-F04) - Menampilkan Peta Lokasi Keseluruhan 119
5.2.5 Skenario Uji Coba (UJ-F05) - Menampilkan Peta Lokasi Khusus 120
5.2.6 Skenario Uji Coba (UJ-F06) - Menampilkan Detail Sensor 122
5.2.7 Skenario Uji Coba (UJ-F07) - Menerima SMS dan Mengirimakan Data ke Server 123
5.2.8 Skenario Uji Coba (UJ-F08) - Mendapatkan Nilai Sensor dan Mengirimkan Data ke Android *Gateway* 124
5.2.9 Skenario Uji Coba (UJ-F09) - Koneksi dengan Satu Hingga Lebih dari Satu *Node* Sensor Sekaligus ... 125
5.2.10 Skenario Uji Coba (UJ-F10) - Menjalankan Android *Gateway* di Beberapa Lokasi Berbeda 126
5.2.11 Skenario Uji Coba (UJ-F11) - Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dinyalakan 128
5.2.12 Skenario Uji Coba (UJ-F12) - Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dimatikan 129
5.3 Hasil Uji Coba Fungsionalitas 131
5.3.1 Hasil Uji Coba (UJ-F01) - Menampilkan Sensor Berdasarkan Kelompok 131

5.3.2 Hasil Uji Coba (UJ-F02) - Mendaftarkan Kelompok Baru.....131
5.3.3 Hasil Uji Coba (UJ-F03) - Mendaftarkan Sensor Baru 133
5.3.4 Hasil Uji Coba (UJ-F04) - Menampilkan Peta Lokasi Keseluruhan.....134
5.3.5 Hasil Uji Coba (UJ-F05) - Menampilkan Peta Lokasi Khusus.....135
5.3.6 Hasil Uji Coba (UJ-F06) - Menampilkan Detail Sensor.....135
5.3.7 Hasil Uji Coba (UJ-F07) - Menerima SMS dan Mengirimakan Data ke Server.....137
5.3.8 Hasil Uji Coba (UJ-F08) - Mendapatkan Nilai Sensor dan Mengirimkan Data ke Android *Gateway*.....138
5.3.9 Hasil Uji Coba (UJ-F09) - Koneksi dengan Satu Hingga Lebih dari Satu *Node* Sensor Sekaligus...140
5.3.10 Hasil Uji Coba (UJ-F10) - Menjalankan Android *Gateway* di Beberapa Lokasi Berbeda.....142
5.3.11 Hasil Uji Coba (UJ-F11) - Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dinyalakan.....145
5.3.12 Hasil Uji Coba (UJ-F12) - Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dimatikan 146
5.4 Skenario Uji Coba Performa.....148
5.4.1 Skenario Uji Coba (UJ-P01) - Lama Waktu Bluetooth untuk Terhubung.....148
5.4.2 Skenario Uji Coba (UJ-P02) - *Delay* Pengiriman Data ke Server.....149
5.4.3 Skenario Uji Coba (UJ-P03) - Akurasi Pengiriman Data Bluetooth.....150
5.4.4 Skenario Uji Coba (UJ-P04) – Penggunaan Daya pada *Node* Sensor.....151
5.4.5 Skenario Uji Coba (UJ-P05) – Penggunaan Daya pada Android *Gateway*.....152

5.5 Hasil Uji Coba Performa 153
5.5.1 Hasil Uji Coba (UJ-P01) - Lama Waktu Bluetooth untuk Terhubung 153
5.5.2 Hasil Uji Coba (UJ-P02) - *Delay* Pengiriman Data ke Server 155
5.5.3 Hasil Uji Coba (UJ-P03) - Akurasi Pengiriman Data Bluetooth 157
5.5.4 Hasil Uji Coba (UJ-P04) - Penggunaan Daya pada *Node* Sensor 158
5.5.5 Hasil Uji Coba (UJ-P05) - Penggunaan Daya pada Android *Gateway* 160
5.6 Evaluasi Hasil Uji Coba 162
BAB VI KESIMPULAN DAN SARAN 165
6.1 Kesimpulan 165
6.2 Saran 166
DAFTAR PUSTAKA 167
LAMPIRAN 173
BIODATA PENULIS 179

DAFTAR GAMBAR

Gambar 2.1 Contoh Ilustrasi Penerapan WSN [6] 9
Gambar 2.2 Contoh Visualisasi Data 11
Gambar 2.3 Arduino UNO R3 [10] 13
Gambar 2.4 Ilustrasi Piconet [16] 15
Gambar 2.5 HC-06 [19] 16
Gambar 2.6 LM35 [21] 17
Gambar 2.7 DHT11 18
Gambar 2.8 Sensor Cahaya 19
Gambar 2.9 Sensor Suara 19
Gambar 3.1 Arsitektur Umum Sistem 27
Gambar 3.2 Diagram Kasus Penggunaan 29
Gambar 3.3 Skema Basis Data (*Sensor Cluster Management*) ... 32
Gambar 3.4 Skema Tabel *users* 33
Gambar 3.5 Skema Tabel *sensor_group* 34
Gambar 3.6 Skema Tabel *sensor* 35
Gambar 3.7 Skema Tabel *sensor_data* 38
Gambar 3.8 Rancangan Antarmuka Halaman Utama 41
Gambar 3.9 Rancangan Antarmuka Halaman *Form New Group* 42
Gambar 3.10 Rancangan Antarmuka Halaman *Form New Sensor* 43
Gambar 3.11 Rancangan Antarmuka Halaman Lokasi Keseluruhan 44
Gambar 3.12 Rancangan Antarmuka Halaman Lokasi Khusus .. 45
Gambar 3.13 Rancangan Antarmuka Halaman Detail Sensor 46
Gambar 3.14 Diagram Alir Sistem (Android *Gateway*) 48
Gambar 3.15 Diagram Alir Subproses *handleMessage* 49
Gambar 3.16 Ilustrasi Komunikasi antara *Node* Sensor dengan Android *Gateway* 49
Gambar 3.17 Diagram Alir Subproses *NormalTask* 50
Gambar 3.18 Ilustrasi Pengiriman Data Sensor (Internet) 51
Gambar 3.19 Ilustrasi Pengiriman Data Sensor (SMS) 52
Gambar 3.20 Diagram Alir Subproses *StrictTask* 53
Gambar 3.21 Diagram Alir Subproses *OnLocationChanged* 54

Gambar 3.22 Diagram Alir Subproses *InternetAvailability* 55
Gambar 3.23 Rancangan Antarmuka Pengguna (Android *Gateway*) 57
Gambar 3.24 Diagram Alir Sistem (SMS *Receiver*) 58
Gambar 3.25 Diagram Alir Subproses *sendSensorData* 59
Gambar 3.26 Ilustrasi SMS *Receiver* Bekerja 60
Gambar 3.27 Skema Tabel *inbox* 61
Gambar 3.28 *Format* Umum Penulisan SMS 62
Gambar 3.29 *Format* Rinci Penulisan SMS 62
Gambar 3.30 Rancangan Rangkaian *Node* Sensor 66
Gambar 3.31 Diagram Alir Sistem (Node Sensor) 67
Gambar 3.32 Ilustrasi Cara Kerja *Node* Sensor 68
Gambar 3.33 Diagram Alir Subproses *setup* 68
Gambar 3.34 Diagram Alir Subproses *updateSensor* 69
Gambar 3.35 Ilustrasi Pengambilan Data dengan Sensor Suhu .. 70
Gambar 4.1 Implementasi Halaman Utama 88
Gambar 4.2 Implementasi Halaman *Form New Group* 88
Gambar 4.3 Implementasi Halaman *Form New Sensor* 89
Gambar 4.4 Implementasi Halaman Peta Lokasi Keseluruhan ... 90
Gambar 4.5 Implementasi Halaman Peta Lokasi Khusus 91
Gambar 4.6 Implementasi Halaman Detail Sensor 92
Gambar 4.7 Implementasi Antarmuka Pengguna (Android *Gateway*) 104
Gambar 4.8 Implementasi Rangkaian Utama *Node* Sensor 110
Gambar 5.1 Menu *New Group* 117
Gambar 5.2 Menu *New Sensor* 118
Gambar 5.3 Menu *Maps* 120
Gambar 5.4 Ikon *Location Marker* 121
Gambar 5.5 Tombol *Detail* 122
Gambar 5.6 SMS yang dikirimkan untuk uji coba 123
Gambar 5.7 Lokasi Pertama 127
Gambar 5.8 Lokasi Kedua 128
Gambar 5.9 Lokasi Ketiga 128
Gambar 5.10 Hasil Uji Coba UJ-F01 (Halaman Utama) 131
Gambar 5.11 Isian Data pada *Form New Group* 132

Gambar 5.12 Hasil Uji Coba UJ-F02 Berupa Tampilan Kelompok Baru 132
Gambar 5.13 Isian Data pada *Form New Sensor* 133
Gambar 5.14 Hasil Uji Coba UJ-F03 Berupa Tampilan Sensor Baru 134
Gambar 5.15 Hasil Uji Coba UJ-F04 Berupa Tampilan Peta Lokasi Keseluruhan 134
Gambar 5.16 Hasil Uji Coba UJ-F05 Berupa Tampilan Peta Lokasi Khusus 135
Gambar 5.17 Hasil Uji Coba UJ-F06 Berupa Tampilan Halaman Detail Sensor 136
Gambar 5.18 SMS yang diterima dalam basis data gammu 137
Gambar 5.19 Status pengiriman data ke server 137
Gambar 5.20 Hasil Uji Coba UJ-F07 Berupa Tampilan Website yang Menunjukkan Data SMS Masuk 138
Gambar 5.21 *Serial monitor* pada *Node* Sensor 1 138
Gambar 5.22 Hasil Uji Coba UJ-F08 berupa data yang diterima Android *Gateway* dari *Node* Sensor 1 138
Gambar 5.23 *Serial monitor* pada *Node* Sensor 2 139
Gambar 5.24 Hasil Uji Coba UJ-F08 berupa data yang diterima Android *Gateway* dari *Node* Sensor 2 139
Gambar 5.25 *Serial monitor* pada *Node* Sensor 3 140
Gambar 5.26 Hasil Uji Coba UJ-F08 berupa data yang diterima Android *Gateway* dari *Node* Sensor 3 140
Gambar 5.27 Hasil Uji Coba UJ-F09 mengkoneksikan Android *Gateway* dengan satu *node* sensor 141
Gambar 5.28 Hasil Uji Coba UJ-F09 mengkoneksikan Android *Gateway* dengan dua *node sensor* sekaligus 141
Gambar 5.29 Hasil Uji Coba UJ-F09 mengkoneksikan Android *Gateway* dengan tiga *node* sensor sekaligus 142
Gambar 5.30 Status pengiriman di lokasi 1 143
Gambar 5.31 Hasil Uji Coba F10 berupa tampilan peta yang menunjukkan lokasi 1 143
Gambar 5.32 Status pengiriman di lokasi 2 144

Gambar 5.33 Hasil Uji Coba F10 berupa tampilan peta yang menunjukkan lokasi 2 144
Gambar 5.34 Status pengiriman di lokasi 3 144
Gambar 5.35 Hasil Uji Coba F10 berupa tampilan peta yang menunjukkan lokasi 3 145
Gambar 5.36 Status pengiriman data via internet (a) *node* sensor 1, (b) *node* sensor 2, (c) *node* sensor 3 146
Gambar 5.37 Hasil Uji Coba UJ-F11 yaitu data yang masuk di basis data server 146
Gambar 5.38 Status pengiriman data via sms (a) *node* sensor 1, (b) *node* sensor 2, (c) *node* sensor 3 147
Gambar 5.39 Status pengiriman data ke server dari 3 *node* sensor 147
Gambar 5.40 Hasil Uji Coba UJ-F12 yaitu data yang masuk di basis data server 148
Gambar 5.41 Grafik Perbandingan *Delay* (*Realtime*) Pengiriman Data Melalui Internet dan SMS 156
Gambar 5.42 Grafik Perbandingan Akurasi Pengiriman Data Bluetooth 158
Gambar 5.43 Grafik Perbandingan Penggunaan Daya Berdasarkan Jeda Waktu Pengiriman Data 159
Gambar 5.44 Grafik Hasil Uji Coba UJ-P05 – Penggunaan Daya pada Pengiriman Data Melalui Internet 160
Gambar 5.45 Grafik Hasil Uji Coba UJ-P05 – Penggunaan Daya pada Pengiriman Data Melalui SMS 161
Gambar 5.46 Grafik Hasil Perbandingan Penggunaan Daya Antara Pengiriman Data Internet dan SMS 162

DAFTAR TABEL

Tabel 3.1 Detail Tabel *users*.....33
Tabel 3.2 Detail Tabel *sensor_group*.....34
Tabel 3.3 Detail Tabel *sensor*.....36
Tabel 3.4 Detail Tabel *sensor_data*.....39
Tabel 3.5 Jenis Koneksi Internet.....55
Tabel 3.6 Kecepatan Koneksi Internet.....56
Tabel 3.7 Detail Tabel *inbox*.....61
Tabel 3.8 Penjelasan Kode Sensor.....63
Tabel 3.9 Komponen Rangkaian.....64
Tabel 3.10 Koneksi *PIN* Arduino dengan Komponen Lain.....65
Tabel 4.1 Lingkungan Implementasi Perangkat Keras.....71
Tabel 4.2 Lingkungan Implementasi Perangkat Lunak.....73
Tabel 4.3 Komponen Rangkaian Implementasi *Node* Sensor...109
Tabel 5.1 Skenario Uji Coba Menampilkan Sensor Berdasarkan Kelompok.....116
Tabel 5.2 Skenario Uji Coba Mendaftarkan Kelompok Baru...117
Tabel 5.3 Skenario Uji Coba Mendaftarkan Sensor Baru.....119
Tabel 5.4 Skenario Uji Coba Menampilkan Peta Lokasi Keseluruhan.....120
Tabel 5.5 Skenario Uji Coba Menampilkan Peta Lokasi Khusus 121
Tabel 5.6 Skenario Uji Coba Menampilkan Detail Sensor.....122
Tabel 5.7 Skenario Uji Coba Menerima SMS dan Mengirimkan Data ke Server.....123
Tabel 5.8 Skenario Uji Coba Mendapatkan Nilai Sensor dan Mengirimkan Data ke Android *Gateway*.....124
Tabel 5.9 Skenario Uji Coba Koneksi dengan Satu Hingga Lebih dari Satu *Node* Sensor Sekaligus.....125
Tabel 5.10 Skenario Uji Coba Menjalankan Android *Gateway* di Beberapa Lokasi Berbeda.....126
Tabel 5.11 Skenario Uji Coba Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dinyalakan.....129

Tabel 5.12 Skenario Uji Coba Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dimatikan.....130
Tabel 5.13 Skenario Uji Coba Lama Waktu Bluetooth untuk Terhubung 148
Tabel 5.14 Skenario Uji Coba *Delay* Pengiriman Data ke Server 149
Tabel 5.15 Skenario Uji Coba Akurasi Pengiriman Data Bluetooth 150
Tabel 5.16 Skenario Uji Coba Penggunaan Daya pada *Node* Sensor 151
Tabel 5.17 Skenario Uji Coba Penggunaan Daya pada Android *Gateway*.....152
Tabel 5.18 Hasil Uji Coba UJ-P01 pada Satu *Node* Sensor.....154
Tabel 5.19 Hasil Uji Coba UJ-P01 pada Dua *Node* Sensor Sekaligus 154
Tabel 5.20 Hasil Uji Coba UJ-P01 pada Tiga *Node* Sensor Sekaligus 155
Tabel 5.21 Hasil Akhir Uji Coba UJ-P01.....155
Tabel 5.22 Hasil Uji Coba UJ-P02 *Delay* Pengiriman Data.....155
Tabel 5.23 Hasil Uji Coba UJ-P03 Akurasi Pengiriman Bluetooth 157
Tabel 5.24 Hasil Uji Coba UJ-P04 Penggunaan Daya *Node* Sensor 159
Tabel 5.25 Evaluasi Hasil Uji Coba Fungsionalitas.....162
Tabel 5.26 Evaluasi Hasil Uji Coba Performa 163
Tabel 7.1 Rincian Hasil Uji Coba UJ-P01 pada Satu *Node* Sensor 173
Tabel 7.2 Rincian Hasil Uji Coba UJ-P01 pada Dua *Node* Sensor Sekaligus 173
Tabel 7.3 Rincian Hasil Uji Coba UJ-P01 pada Tiga *Node* Sensor Sekaligus 174
Tabel 7.4 Rincian Hasil Uji Coba UJ-P02 (*Delay* Internet).....175
Tabel 7.5 Rincian Hasil Uji Coba UJ-P02 (*Delay* SMS) 175
Tabel 7.6 Rincian Hasil Uji Coba UJ-P02 (*Delay* Bluetooth)...175

Tabel 7.7 Rincian Hasil Uji Coba UJ-P03 pada Jarak ≤1 Meter 176
Tabel 7.8 Rincian Hasil Uji Coba UJ-P03 pada Jarak ±5 Meter 176
Tabel 7.9 Rincian Hasil Uji Coba UJ-P03 pada Jarak ±10 Meter 176
Tabel 7.10 Rincian Hasil Uji Coba UJ-P04 Penggunaan Daya *Node* Sensor.....176

DAFTAR KODE SUMBER

Kode Sumber 4.1 *Pseudocode* Fungsi *index* 75
Kode Sumber 4.2 *Pseudocode* Fungsi *getGroupList* 76
Kode Sumber 4.3 *Pseudocode* Fungsi *getLastSensorList* 76
Kode Sumber 4.4 *Pseudocode* Fungsi *createGroup* 77
Kode Sumber 4.5 *Pseudocode* Fungsi *createNewGroup* 78
Kode Sumber 4.6 *Pseudocode* Fungsi *createSensor* 79
Kode Sumber 4.7 *Pseudocode* Fungsi *createNewSensor* 79
Kode Sumber 4.8 *Pseudocode* Fungsi *viewMaps* 80
Kode Sumber 4.9 *Pseudocode* Fungsi *gatewayLocation* 80
Kode Sumber 4.10 *Pseudocode* Fungsi *getGroupData* 81
Kode Sumber 4.11 *Pseudocode* Fungsi *getLastSensorData* 81
Kode Sumber 4.12 *Pseudocode* Fungsi *viewSensor* 82
Kode Sumber 4.13 *Pseudocode* Fungsi *getSensorData* 83
Kode Sumber 4.14 *Pseudocode* Fungsi *getSensorReading* 83
Kode Sumber 4.15 *Pseudocode* Fungsi *retrieveDataSensor* 85
Kode Sumber 4.16 *Pseudocode* Fungsi *checkSensorKey* 86
Kode Sumber 4.17 *Pseudocode* Fungsi *insertSensorReadingData* 86
Kode Sumber 4.18 *Pseudocode* Fungsi *updateLastUpdatedSensor* 86
Kode Sumber 4.19 *Pseudocode* Fungsi *updateLocation* 87
Kode Sumber 4.20 *Pseudocode* Fungsi *updateGatewayLocation* 87
Kode Sumber 4.21 *Pseudocode* Fungsi *onCreate* 95
Kode Sumber 4.22 *Pseudocode* Fungsi *onResume* 96
Kode Sumber 4.23 *Pseudocode* Fungsi *handleMessage* 98
Kode Sumber 4.24 *Pseudocode* Kelas *NormalTask* 100
Kode Sumber 4.25 *Pseudocode* Kelas *StrictTask* 102
Kode Sumber 4.26 *Pseudocode* Fungsi *onLocationChanged* ... 102
Kode Sumber 4.27 *Pseudocode* Fungsi *InternetAvailabilty* 103
Kode Sumber 4.28 *Pseudocode* Fungsi *onSensorChanged* 104
Kode Sumber 4.29 Konfigurasi *File gammurc* 106
Kode Sumber 4.30 Konfigurasi *File smsdrc* 106

Kode Sumber 4.31 *Pseudocode* Fungsi *main*.....108
Kode Sumber 4.32 *Pseudocode* Fungsi *sendSensorData*.....109
Kode Sumber 4.33 *Pseudocode* Fungsi *setup* dan Inisialisasi Variabel Global 112
Kode Sumber 4.34 *Pseudocode* Fungsi *loop*.....113
Kode Sumber 4.35 *Pseudocode* Fungsi *updateSensor* 114

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di zaman yang semakin maju seperti saat ini, perkembangan teknologi di bidang komunikasi berlangsung relatif sangat pesat. Setiap waktunya, beragam metode baru mengenai teknologi komunikasi diterapkan untuk menunjang berbagai aspek kehidupan manusia. Salah satu bidang teknologinya adalah *Wireless Sensor Network* (WSN). WSN merupakan salah satu jenis jaringan nirkabel yang bekerja dengan memanfaatkan *node-node* sensor dengan tujuan melakukan proses sensor, *monitoring*, pengiriman data dan menyajikan informasi kepada pengguna melalui komunikasi internet [1]. WSN merupakan teknologi yang terbilang populer untuk diterapkan.

Salah satu modul yang cocok digunakan adalah Bluetooth. Penjelasan tentang modul Bluetooth diatur dalam IEEE 802.15.1. Modul Bluetooth adalah modul komunikasi nirkabel untuk melakukan pertukaran data jarak pendek [2]. Alasan utama penggunaan modul ini karena penggunaannya yang terbilang hemat daya.

Virtualisasi sensor [3] dapat diterapkan untuk memberikan kemudahan dalam pengelolaan data-data sensor. Virtualisasi sensor dapat dilakukan untuk *monitoring* dan menampilkan visualisasi data dari sensor fisik. Melalui sebuah sistem *Sensor Cluster Management*, pengguna dapat memantau data sensor secara visual. Istilah ini sering dikenal dengan nama *sensor cloud* [4].

Berdasarkan hal di atas, akan dibuat suatu jaringan sensor yang nantinya semua *node* sensor diharapkan dapat mengirimkan data ke sebuah server. Namun, jika semua *node* diharuskan melakukan pengiriman data secara langsung ke server, maka artinya setiap *node* harus terhubung dengan jaringan internet secara

langsung pula. Hal ini akan menjadi masalah jika jumlah *node* relatif banyak, karena setiap *node* akan membutuhkan modem untuk dapat terhubung dengan internet secara langsung. Sehingga perlu adanya sebuah *node* yang menjadi perantara antar *node-node* sensor dengan server. *Node* ini biasa disebut *node gateway*, *node* inilah yang akan menjembatani komunikasi antara *node-node* yang ada dengan server.

Node gateway adalah *gateway* adaptif yang dimungkinkan adalah sebuah perangkat ponsel pintar (*smartphone*) berbasis Android, sehingga bisa juga disebut Android *gateway*. Peningkatan popularitas ponsel pintar 10 tahun belakangan ini menarik minat berbagai pihak untuk menggunakan ponsel pintar. Mobilitasnya yang tinggi, antarmuka yang mudah dipelajari dan kehandalan kinerja komputasi menjadi alasan utama meningkatnya penggunaan ponsel pintar [5].

Proses awal yang dilakukan adalah sensor akan mendeteksi keadaan lingkungan sekitarnya. Setelah itu, data yang berhasil direkam oleh sensor akan dikirimkan pada Android *gateway* melalui komunikasi via Bluetooth. Komunikasi antara *node-node* sensor dengan Android *gateway* dilakukan secara bergiliran. Android *gateway* juga melakukan pemantauan ketersediaan koneksi internet, jika koneksi internet terganggu atau sedang tidak dapat digunakan, maka pengiriman data ke server akan dilakukan dengan menggunakan sms. Di sini lah Android *gateway* diharapkan dapat berperan adaptif terhadap ketersediaan koneksi internet.

Pengerjaan tugas akhir ini diharapkan akan menghasilkan sebuah jaringan sensor nirkabel dimana data yang didapat dari *node* sensor akan dikirim ke Android *gateway* melalui Bluetooth, kemudian Android *gateway* akan berperan adaptif dalam menentukan proses pengiriman data ke sebuah server melalui internet atau melalui sms. Data yang berhasil direkam di server diharapkan dapat disajikan dalam sebuah halaman website.

1.2 Rumusan Masalah

Rumusan masalah yang berusaha diselesaikan dalam tugas akhir ini adalah :

1. Bagaimana membuat rancangan sistem *Sensor Cluster Management* untuk menyediakan layanan virtualisasi sensor?
2. Bagaimana menentukan cara pengiriman data secara adaptif (via internet atau sms) dari *gateway* ke server?
3. Bagaimana rancangan komunikasi nirkabel berbasis Bluetooth (koneksi antara *node* sensor dengan *gateway*)?
4. Performa dari keseluruhan sistem yang meliputi:
    a. Berapa lama waktu yang dibutuhkan antar komunikasi Bluetooth untuk dapat saling terhubung satu sama lain?
    b. Bagaimana *delay* pengiriman data melalui sms jika dibandingkan dengan *delay* pengiriman data melalui internet?
    c. Bagaimana perbandingan akurasi pengiriman data menggunakan Bluetooth jika melibatkan jarak sebagai acuan perbandingan?
    d. Bagaimana perbandingan konsumsi daya dari *node* sensor jika dilakukan variasi terhadap jeda waktu pengiriman data sensor?
    e. Bagaimana perbandingan konsumsi daya dari Android *gateway* antara pengiriman data melalui internet dengan pengiriman data melalui sms?

1.3 Batasan Permasalahan

Pada tugas akhir ini ada beberapa batasan yang menjadi pertimbangan, berikut ini batasan-batasan yang menjadi pertimbangan :

1. Modul *wireless* yang digunakan untuk komunikasi dari *node* sensor ke *node gateway* adalah modul Bluetooth.
2. Jumlah *node* yang digunakan sebanyak 5 dengan 1 sebagai server, 1 *node gateway* (Android) dan 3 sebagai *end device* atau *node* sensor yang mengirimkan data secara bergantian.
3. *Node* tambahan sejumlah 1 buah yang berupa modul sms *gateway* (*receiver* sms yang dapat dipasang di satu tempat dengan server ataupun dipasang terpisah dari server).
4. *Microcontroller* yang digunakan adalah Arduino.
5. *Node gateway* berupa ponsel pintar (*smartphone*) berbasis Android.
6. Biaya pengiriman sms tidak menjadi bahasan utama.

1.4 Tujuan

Tujuan dari pengerjaan Tugas Akhir ini adalah :

1. Mengembangkan sistem *Sensor Cluster Management* untuk melakukan pengelompokkan (*cluster*) dan pengelolaan data pada sensor – sensor yang tersebar.
2. Membangun *gateway* adaptif yang mampu menentukan cara pengiriman data ke server berdasarkan ketersediaan koneksi internet.
3. Mengembangkan sistem komunikasi *node* sensor menggunakan Bluetooth.

1.5 Manfaat

Pengerjaan tugas akhir ini memiliki manfaat untuk menghasilkan sebuah jaringan sensor nirkabel yang dapat menentukan proses pengiriman data melalui koneksi internet atau melalui sms sehingga data dapat dipantau secara langsung melalui sebuah website.

Pengerjaan tugas akhir ini juga memiliki manfaat tersendiri bagi penulis. Manfaat bagi penulis yaitu sebagai sarana untuk mengimplementasikan telah dipelajari selama kuliah agar berguna bagi masyarakat.

1.6 Metodologi

Berikut ini metodologi yang diterapkan dalam pengerjaan tugas akhir :

1.6.1 Penyusunan Proposal

Tahap awal pengerjaan tugas akhir ini dimulai dengan penyusunan Proposal Tugas Akhir. Proposal Tugas Akhir ini berisi tentang perencanaan perancangan jaringan sensor nirkabel menggunakan modul Bluetooth sebagai media pertukaran data dari *node* sensor ke *node gateway*, dalam hal ini Android *gateway* juga berperan dalam menentukan proses pengiriman data ke server menggunakan koneksi internet atau menggunakan sms berdasarkan ketersediaan koneksi internet.

Proposal Tugas Akhir ini terdiri dari deskripsi pendahuluan yang berisi penjabaran latar belakang dan rumusan masalah yang menjadi dasar pengerjaan tugas akhir, batasan masalah dalam perancangan jaringan, serta tujuan dan manfaat yang diharapkan dapat tercapai dengan perancangan jaringan nirkabel ini. Pada proposal Tugas Akhir ini juga menyertakan tinjauan pustaka yang menjelaskan berbagai teori yang menjadi dasar pembuatan tugas akhir ini, yaitu modul Bluetooth dan penggunaan variabel ketersediaan koneksi internet dalam penentuan proses pengiriman data ke server.

1.6.2 Studi Literatur

Studi literatur yang dilakukan dalam pengerjaan Tugas Akhir ini dilakukan dengan mencari informasi dari sumber-sumber

terkait yang membahas mengenai perancangan jaringan sensor nirkabel dengan menggunakan modul Bluetooth dan Arduino untuk melakukan pengiriman data antara *node* sensor dengan Android *gateway*. Selain itu, juga dilakukan studi literatur mengenai penentuan proses pengiriman data menggunakan koneksi internet atau sms antara Android *gateway* dengan server berdasarkan ketersediaan koneksi internet.

Hal-hal lain yang dipelajari dalam studi literatur meliputi mikrokontroler Arduino, pencarian modul Bluetooth yang cocok, pemrograman Arduino, pemrograman Android, pemrograman Python, modul sms *gateway*, basis data, pemrograman website dan lain sebagainya.

1.6.3 Analisis dan Desain Perangkat Lunak

Tahap ini meliputi perancangan dan analisis sistem berdasarkan studi literatur. Berdasarkan konsep teknologi dari perangkat lunak yang ada saat ini, dilakukan perancangan sistem yang akan dibangun. Langkah-langkah pengerjaan juga didefinisikan pada tahap ini.

Pada tahapan ini dibuat prototipe dari sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Selain itu, dalam tahap ini juga dilakukan desain sistem dan proses-proses yang ada.

1.6.4 Implementasi Perangkat Lunak

Implementasi Perangkat Lunak merupakan tahapan untuk membangun rancangan program yang telah dibuat. Pada tahapan ini merealisasikan apa yang terdapat pada tahapan sebelumnya, sehingga menjadi sebuah program yang sesuai dengan apa yang telah direncanakan.

1.6.5 Pengujian dan Evaluasi

Pada tahapan ini dilakukan uji coba pada alat yang telah dibuat. Tahapan ini dimaksudkan untuk mengevaluasi tingkat akurasi dan performa dari alat tersebut serta mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

1.6.6 Penyusunan Buku

Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan tugas akhir yang mencakup seluruh konsep, dasar teori, implementasi, serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan

Penulisam buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir secara menyeluruh. Selain itu, diharapkan dapat bermanfaat bagi pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan tugas akhir.

2. Bab II. Tinjauan Pustaka
   Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

3. Bab III. Perancangan Perangkat Lunak

Bab ini berisi implementasi dari perancangan perangkat lunak yang telah dibuat pada bab sebelumnya. Implementasi berupa *pseudocode* dari fungsi utama dan *screenshot* perangkat lunak.

4. Bab IV. Implementasi
   Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

5. Bab V. Hasil Uji Coba 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 solusi ke depannya.

7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.

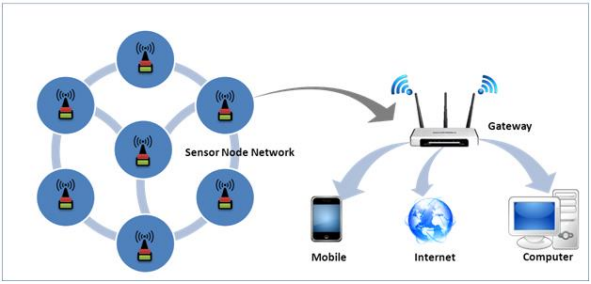
8. Lampiran
   Lampiran yang ada berisi kelengkapan – kelengkapan yang diperlukan dalam menyusun buku tugas akhir.

BAB II
TINJAUAN PUSTAKA

Pada bab ini, dijabarkan tentang penjelasan teori-teori yang berkaitan dengan pokok bahasan tugas akhir. Bab ini juga menjelaskan modul dan alat yang nantinya akan digunakan pada tahap implementasi program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap alat yang digunakan dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Jaringan Sensor Nirkabel

Jaringan sensor nirkabel sering dikenal dengan nama *wireless sensor network* (WSN). WSN merupakan salah satu jenis jaringan nirkabel yang bekerja dengan memanfaatkan *node-node* sensor dengan tujuan melakukan proses sensor, *monitoring*, pengiriman data dan menyajikan informasi kepada pengguna melalui komunikasi internet. Jaringan sensor nirkabel biasanya memanfaatkan teknologi *Embedded System* (sistem benam) pada *node-node* sensor yang ada. Sensor-sensor yang umumnya digunakan dalam jaringan ini ada bermacam-macam, contohnya sensor suhu, kelembaban, intensitas cahaya, suara, getaran dan sebagainya [1]. Gambar 2.1 merupakan ilustrasi atau gambaran mengenai penerapan WSN.



Gambar 2.1 Contoh Ilustrasi Penerapan WSN [6]

Cikal bakal teknologi *wireless sensor network* (WSN) dimulai pada tahin 1950 oleh Amerika Serikat. Enam komponen utama dalam *node* sensor adalah sensor, *transceiver*, memori eksternal, kontroler, *power source*, *Analog to Digital Converter* (ADC).

Pada tugas akhir akan dibangun sebuah jaringan nirkabel yang di dalamnya terdiri dari beberapa *node* sensor.

2.2 Sensor Cloud

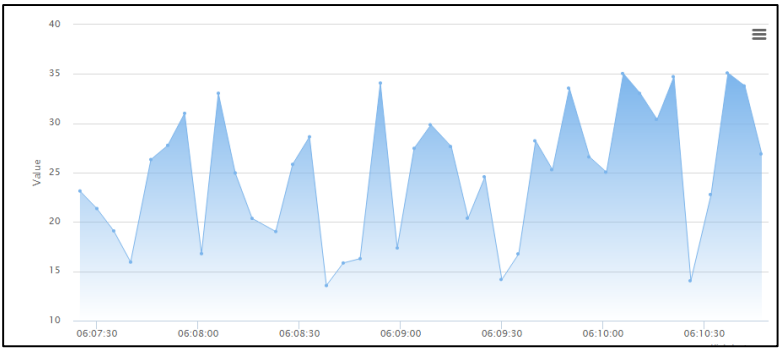
Sensor Cloud [4] merupakan istilah yang sering digunakan dengan menerapkan konsep virtualisasi sensor. Virtualisasi sensor [3] merupakan metode untuk menyediakan layanan sensor secara virtual. Sensor virtual yang dibuat juga telah dikelompokkan (*cluster*) sedemikian rupa sehingga pengguna dapat lebih mudah dalam melakukan pengelolaan data. Perancangan sensor dan kelompok sensor virtual berguna agar pengguna dapat melakukan pemantauan tanpa perlu mengkhawatirkan tentang faktor lokasi dan spesifikasi sensor fisik.

Berdasarkan konsep tersebut, pada Tugas Akhir ini muncul ide untuk mengembangkan sistem virtualisasi sensor yang dinamakan *Sensor Cluster Management*. Sistem ini digunakan untuk mengelola dan mengelompokkan sensor - sensor yang berada di lokasi tersebar.

2.3 Visualisasi Data

Visualisasi data pada banyak bidang ilmu pengetahuan dianggap sebagai bentuk komunikasi visual gaya baru (*modern*). Visualisasi data itu sendiri tidak termasuk dalam bidang keilmuan manapun, melainkan merupakan cara yang digunakan dalam menyuguhkan data dalam bentuk visual. Pada penggunaannya, data yang ada diolah sedemikian rupa sehingga menghasilkan informasi yang telah disajikan dalam bentuk skematis, termasuk

variabel atau atribut dari unit informasi. Gambar 2.2 merupakan data yang telah divisualisasikan.



Gambar 2.2 Contoh Visualisasi Data

Tujuan utama dari visualisasi data adalah untuk memudahkan penyampaian informasi secara jelas dan efisien kepada pengguna melalui media grafik informasi tertentu, seperti tabel dan grafik. Visualisasi yang efektif dapat membantu seseorang dalam menganalisis dan memahami data yang ada dengan lebih efektif dan efisien. Proses visualisasi membuat data yang kompleks bisa lebih mudah untuk ditelaah, dipahami dan lebih memiliki banyak kegunaan. Berdasarkan jenis datanya, visualisasi data dibedakan menjadi dua, yaitu *scientific visualization* dan *information visualization* [7].

Penggunaan visualisasi data pada tugas akhir ini adalah untuk menampilkan data sensor dalam bentuk visual.

2.4 *Gateway* Adaptif

Gateway merupakan merupakan jembatan yang menjadi perantara untuk komunikasi dan pertukaran data antara dua perangkat (komputer, ponsel dan sebagainya) atau lebih. *Gateway* ini mengatur arus data dan mekanisme pertukaran data tersebut, sehingga perangkat yang lain dapat saling berhubungan.

Gateway adaptif [8] artinya adalah sebuah *gateway* yang dapat melakukan suatu tindakan tertentu (sesuai apa yang telah di programkan) berdasarkan keadaan-keadaan yang saat itu sedang terjadi di sekitarnya. Jika dilihat lebih spesifik, keadaan yang dimaksud di sini berhubungan dengan *resources* atau sumber daya yang mempengaruhi. Sebagai contohnya, dalam kaitannya dengan WSN, sumber daya yang ada meliputi kapasitas memori, jumlah energi yang dimiliki (baterai), ketersediaan koneksi internet dan sebagainya. Hal-hal tersebut menjadi pertimbangan *gateway* untuk dapat beradaptasi terhadap sumber daya yang dimiliki.

Gateway adaptif dalam kaitannya dengan Tugas Akhir ini, yaitu *gateway* yang mampu beradaptasi terhadap ketersediaan koneksi internet. Pengiriman data yang dilakukan *gateway* ke server bergantung pada ketersediaan koneksi internet. Saat koneksi internet tidak tersedia, maka pengiriman data dialihkan dengan menggunakan sebuah pesan berupa sms. Hal ini dilakukan agar data-data yang ada setiap waktunya tetap dapat direkam dalam komputer server.

2.5 Mikrokontroler Arduino UNO

Arduino Uno dikenal sebagai salah satu mikrokontroler yang diproduksi oleh Atmel AVR Atmega328. Alat ini memiliki 14 *input* atau *output* digital, dengan 6 *input* analog (10 *bits* ADC) dan memiliki 32 KB *flash memory*. Sumber energinya dapat dihubungkan langsung melalui sebuah port USB atau bisa juga melalui sumber energi eksternal. Sumber energi ini dapat terdeteksi secara otomatis. AVR Atmega328 merupakan mikrokontroler (8 bit) bertenaga rendah yang berjalan pada 16 MHz [9]. Gambar 2.3 merupakan contoh fisik dari mikrokontroler Arduino UNO R3.

Gambar 2.3 Arduino UNO R3 [10]

Spesifikasi lain dari Arduino UNO adalah memiliki tegangan operasi sebesar 5 Volt. *Input* tegangan yang direkomendasikan melalui *jack* DC adalah sebesar 7V – 12V. Harus diperhatikan juga bahwa batas *input* tegangan melalui *jack* DC yaitu antara 6V – 20V. Ukuran dari Arduino UNO ini sekitar 68,6 mm x 53,4 mm dan memiliki berat sekitar 25 gram. Program dapat dimasukkan ke dalam Arduino UNO dengan cara diunggah menggunakan koneksi USB *type A to type B* [11].

Selain berat dan ukurannya yang relatif kecil, Arduino UNO juga memiliki beberapa keunggulan lain. Berikut ini beberapa keunggulan lain dari Arduino UNO :

- Harga relatif terjangkau (untuk saat ini sekitar $10 - $15 US Dollar).
- Dapat dijalankan pada berbagai sistem operasi seperti Windows, Linux, Mac dan sebagainya.
- Bahasa pemrograman relatif sederhana sehingga mudah dipelajari.

- Perangkat keras maupun perangkat lunak merupakan produk *open-source*.

Arduino UNO R3 akan digunakan sebagai mikrokontroler untuk membangun *node* sensor pada tugas akhir.

2.6 Bluetooth

Bluetooth merupakan sarana bertukar informasi nirkabel yang termasuk pada kawasan pribadi (*personal*). Hal ini juga sering disebut dengan WPAN (*Wireless Personal Area Network*). Bluetooth memiliki standar khusus yang diatur dalam jurnal IEEE 802.15.1. Ranah frekuensi beroprasinya Bluetooth adalah pada pita frekuensi 2,4 GHz. Penggunaan *frequency hopping transceiver* mampu memberikan layanan komunikasi data dalam jarak yang terbatas. Kelebihan Bluetooth terletak pada transmisinya yang mampu menembus dinding meskipun hanya dalam jarak yang terbatas [12], [13].

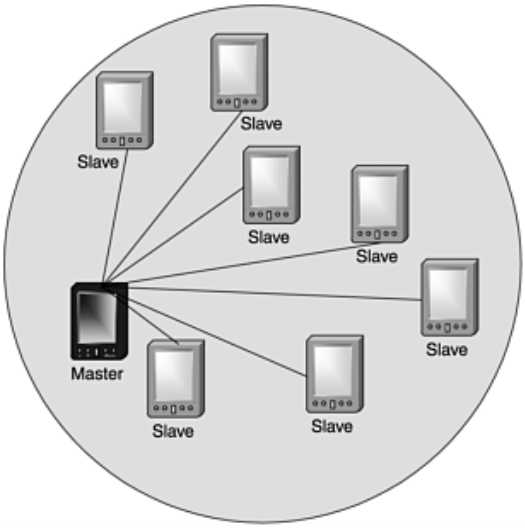
Standarisasi Bluetooth diatur dalam IEEE 802.15.1. Bluetooth versi 4.0 atau lebih tinggi, memungkinkan beberapa *device* untuk terkoneksi dalam waktu yang bersamaan [14].

Pada tugas akhir ini, Blutooth digunakan sebagai sarana pertukaran data dari *node* sensor ke Android *gateway*.

2.7 Piconet

Piconet [15] merupakan suatu jaringan antar perangkat-perangkat yang saling terhubung menggunakan Bluetooth. Jaringan ini terdiri dari minimal dua perangkat hingga delapan perangkat, dengan satu perangkat bertindak sebagai *master* dan sisanya bertindak sebagai *slave*.

Piconet biasanya diterapkan pada jaringan rumahan yang jangkauan jaringannya hanya untuk perangkat yang berada di dalam rumah (ruangan). Gambar 2.4 merupakan ilustrasi hubungan antar perangkat – perangkat dalam sebuah Piconet.



Gambar 2.4 Ilustrasi Piconet [16]

Sebuah Piconet dapat berkomunikasi dengan Piconet lain untuk membentuk jaringan yang lebih besar. Komunikasi dari beberapa Piconet membentuk jaringan yang disebut Scatternet.

Tugas akhir ini menjadikan Piconet sebagai salah satu dasar untuk merancang arsitektur sistem.

2.8 Variabel Ketersediaan Koneksi Internet

Variabel ketersediaan koneksi internet digunakan untuk menentukan proses pengiriman data dari Android *gateway* ke server. Jika koneksi internet sedang dalam kondisi baik maka pengiriman data dapat dilakukan melalui internet. Namun, jika koneksi internet sedang buruk (*not available*) maka pengiriman data dilakukan via sms.

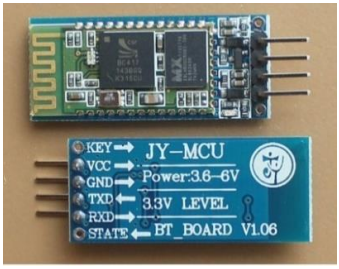
Ketersediaan koneksi internet dapat dilihat dari terhubung atau tidaknya dengan *wifi* atau paket data. Selain itu juga dapat dilihat dari jenis koneksi yang terhubung. Koneksi dikatakan baik jika jenis koneksi yang terhubung adalah *WIFI, EVDO_0,*

EVDO_A, EVDO_B, HSDPA, HSPA, HSUPA, UMTS, EHRDP, HSPAP dan *LTE*. Sedangkan untuk koneksi yang dikatakan tidak terlalu baik yaitu jika jenis koneksinya adalah *1xRTT, CDMA, EDGE, GPRS* dan *IDEN* [17].

Perbandingan lain yang digunakan untuk menentukan ketersediaan internet adalah melalui kecepatan koneksi. Kecepatan internet dibagi menjadi 5 yaitu *UNKNOWN* (kurang dari 0 kbps), *POOR* (1 hingga 150 kbps), *MODERATE* (150 hingga 550 kbps), *GOOD* (551 hingga 2000 kbps) dan *EXCELLENT* (lebih dari 2000 kbps) [18]. Penggabungan dari dua kondisi di atas akan digunakan untuk menentukan ketersediaan koneksi internet pada tugas akhir.

2.9 Modul HC-06

Modul Bluetooth HC-06 adalah modul komunikasi nirkabel jarak pendek untuk melakukan pertukaran data. Modul ini bekerja menggunakan gelombang radio UHF (*Ultra High Frequency*) dengan rentang ISM (*Industrial, Scientific and Medical*) 2,4 – 2,485 GHz. Jarak komunikasi yang dapat diakomodasi modul ini mencapai radius 10 meter. Modul ini berbasis pada Cambridge Silicon Radio BC417 2,4 GHz (*BlueTooth Radio Chip*) yang merupakan perangkat rumit dengan menggunakan 8 Mbit memori *flash* eksternal [2]. Modul ini cocok digunakan dengan menggunakan *Microcontroller* Arduino. Gambar 2.5 adalah contoh fisik modul HC-06.



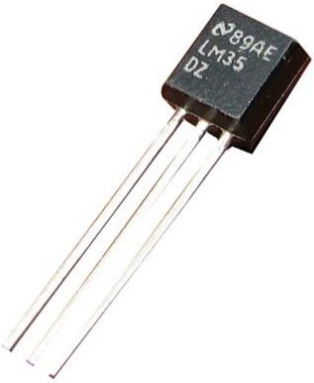
Gambar 2.5 HC-06 [19]

Modul HC-06 dapat disambungkan dengan Arduino melalui komunikasi serial. Modul ini memiliki 4 buah *pin*. Satu *pin* untuk VCC yang dapat disambungkan dengan 5V pada Arduino. Satu *pin* untuk GND (*ground*). Dua *pin* yang lainnya adalah RX (disambungkan dengan TX pada mikrokontroler) dan TX (disambungkan dengan TX pada mikrokontroler) [20].

Modul HC-06 adalah modul yang akan digunakan dalam tugas akhir ini untuk menerapkan pertukaran data melalui Bluetooth.

2.10 Modul LM35

Modul LM35 merupakan komponen elektronik untuk mendeteksi suhu (*temperature*). Komponen ini adalah salah satu jenis sensor suhu yang banyak digunakan di pasaran. Penggunaannya dapat dengan mudah diatur dengan menggunakan Arduino. LM35 memiliki 3 *pin* yang masing-masingnya adalah VCC, V_{OUT} dan GND. Gambar 2.6 merupakan contoh modul LM35.



Gambar 2.6 LM35 [21]

LM35 ini memiliki sensitivitas suhu dengan faktor skala *linier* 10 mVolt/^{O}C sehingga membuatnya dapat dikalibrasi secara

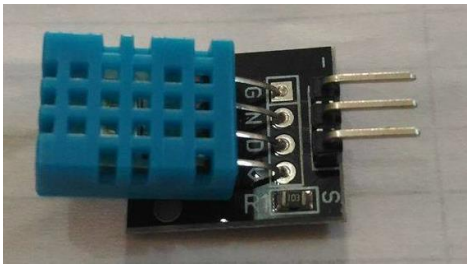
langsung dalam derajat celcius. Rentang operasi suhunya adalah -55 sampai 150 ^{O}C. LM35 bekerja pada tegangan 4 – 30 volt. Pada suhu ruangan, akurasi kalibrasi yaitu 0,5 ^{O}C [22]. [23], [24].

Sensor suhu LM35 merupakan sensor yang akan digunakan untuk mengecek suhu udara pada *node* sensor yang akan dibangun.

2.11 Modul DHT11

Modul DHT11 merupakan modul sensor yang dapat digunakan untuk mendeteksi kelembaban udara. Meskipun dapat juga digunakan untuk mendeteksi suhu udara, namun data yang didapat tidak terlalu detail [25]. Gambar 2.7 merupakan bentuk fisik dari DHT11.

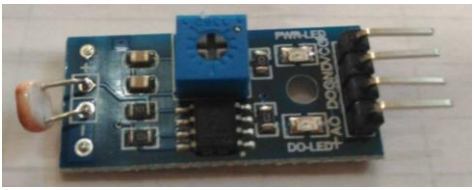
Pada tugas akhir ini DHT11 digunakan sebagai sensor untuk mengukur nilai kelembaban udara. Alasan pemilihan sensor ini adalah penggunaannya yang relatif mudah dan harganya yang relatif murah.



Gambar 2.7 DHT11

2.12 Modul Sensor Cahaya (LM393)

Modul sensor LM393 ini merupakan modul sensor cahaya yang sederhana. Modul ini memungkinkan *output* data berupa data analog maupun data digital [26]. Gambar 2.1 bentuk fisik dari sensor cahaya.

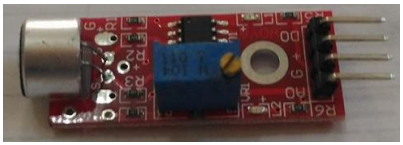


Gambar 2.8 Sensor Cahaya

Sensor ini digunakan pada tugas akhir untuk mengukur intensitas cahaya. Penggunaannya yang fleksibel menjadi pilihan utama pemilihan modul ini.

2.13 Modul Sensor Suara (KY-038)

Modul sensor KY-038 adalah modul sensor untuk mendeteksi keberadaan suara. Modul ini dapat menghasilkan data dalam bentuk data analog maupun digital [27]. Gambar 2.9 merupakan bentuk fisik dari modul sensor suara.



Gambar 2.9 Sensor Suara

Sensor ini digunakan untuk mendeteksi suara pada *node* sensor yang akan dibuat. Alasan pemilihan sensor ini adalah penggunaannya yang relatif fleksibel.

2.14 Android

Android merupakan sistem operasi berbasis Linux yang perancangannya digunakan untuk perangkat bergerak dengan layar sentuh. Perangkat-perangkat yang menggunakan sistem operasi Android biasanya seperti komputer tablet dan ponsel pintar (*smartphone*).

Hingga saat ini telah ada beberapa versi dari Android. Beberapa diantaranya adalah Android versi 1.0, Android Cupcake,

Android Donut, Android Eclair, Android Froyo, Android Gingerbread, Android Honeycomb, Android Ice Cream Sandwich, Android Jelly Bean, Android KitKat, Android Lollipop, Android Marshmallow dan yang terbaru adalah Android Nougat [28].

Android merupakan sistem operasi tempat berjalannya gateway adaptif yang disebut juga dengan Android *Gateway*. Versi yang digunakan adalah Android Marshmallow.

2.15 CodeIgniter

CodeIgniter (CI) merupakan aplikasi *open-source* yang bekerja berdasarkan teori MVC (*Model*, *View*, *Controller*). MVC adalah konsep yang cukup popular dalam pembangunan aplikasi berbasis website.

Secara sederhana konsep MVC terdiri dari 3 bagian yaitu *Model*, *View* serta *Controller*. Sebuah website dinamis setidaknya memiliki 3 elemen utama sebagai penyusunnya, yaitu basis data, logika aplikasi dan cara menampilkan halaman website. Hal itu direpresentasikan menggunakan MVC dimana basis data diakses lewat *Model*, logika aplikasi diatur dalam *Controller* dan *View* digunakan untuk menampilkan halaman website [29].

Ada beberapa kelebihan CodeIgniter (CI) dibandingkan dengan *Framework* PHP lain, diantaranya :

1. Performa sangat cepat.
2. Konfigurasi yang sangat minim (*nearly zero configuration*).
3. Banyak sekali komunitas.
4. Dokumentasi yang sangat lengkap.

Code Igniter ini akan digunakan untuk membangun *back-end* yang ada pada website *Sensor Cluster Management*.

2.16 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*database management system*). MySQL

merupakan turunan salah satu konsep utama dalam basis data yang sudah ada sejak lama, yaitu SQL (*Structured Query Language*). MySQL memungkinkan pengoperasian data dapat dikerjakan dengan mudah secara otomatis. MySQL memiliki beberapa kelebihan, antara lain portabilitas, *open-source*, skalabilitas, *multi-user*, keamanan cukup baik dan lain sebagainya [30].

Basis data MySQL ini adalah basis data yang akan digunakan pada server.

2.17 Bootstrap

Bootstrap [31] adalah *framework* yang dikembangkan dengan menggunakan HTML, CSS dan JS. Bootstrap biasa digunakan untuk memudahkan *programmer* dalam membuat sebuah tampilan website yang responsif.

Bootstrap versi 3 yang ditulis ulang memanfaatkan fungsional responsif dengan pendekatan pertama *mobile*. Bootstrap ini akan digunakan untuk membuat tampilan sistem *Sensor Cluster Management*.

2.18 SMS *Gateway*

SMS *Gateway* [32] merupakan perangkat yang memiliki fungsi untuk penghubung lalu lintas pesan (sms). Perangkat ini dapat digunakan untuk mengirim maupun menerima sms, untuk kemudian diolah sesuai kebutuhan.

Pada penerapannya, SMS *Gateway* digunakan untuk menjembatani komunikasi yang berupa sms antara ponsel dengan perangkat komputer. Sebuah perangkat SMS *Gateway* umumnya terdiri dari tiga komponen penting yaitu *hardware* (komputer dan ponsel / modem), *software* (aplikasi pengolah pesan) dan *database* (tempat penyimpanan data) [33].

SMS *Gateway* adalah dasar teori dibangunnya sistem untuk pengelolaan sms.

2.19 Gammu

Gammu [34] merupakan nama sebuah proyek *open-source*. Gammu juga merupakan nama dari fungsi pada *command line* yang dapat digunakan untuk mengendalikan aktivitas ponsel. Proyek ini dibuat dalam bahasa pemrograman C dan dibangun di atas pustaka libGammu.

Penggunaan Gammu dapat berfungsi untuk mengakses fitur-fitur yang dimiliki suatu ponsel. Fitur yang dapat diakses menggunakan Gammu berbeda untuk setiap jenis ponsel dengan merk dan jenis yang berbeda. Fitur-fitur yang dapat diakomodasi oleh Gammu mencakup pemanggilan telepon, pengelolaan sms maupun mms, pengelolaan buku telepon di ponsel, pengelolaan kalender, pembacaan informasi ponsel / jaringan dan akses ke dalam berkas sistem ponsel.

Gammu merupakan alat bantu untuk membangun sistem penerima sms (SMS *Receiver*).

2.20 Python

Python merupakan salah satu bahasa pemrograman tingkat tinggi yang bersifat *interpreter*, interaktif serta *object oriented*. Python dapat beroperasi pada banyak *platform* berbeda seperti UNIX, Mac, Windows ataupun yang lainnya. Karena sintaksnya yang lebih sederhana dibandingkan bahasa pemrograman lain seperti C atau C++, bahasa pemrograman Python menjadi salah satu bahasa pemrograman tingkat tinggi yang relatif mudah dipelajari [35], [36].

Pesan yang diterima oleh SMS *Receiver* akan diolah menggunakan bahasa Pemrograman Python sebelum diteruskan ke server.

2.21 Highcharts

Highcharts [37] merupakan sebuah pustaka (*library*) pengelolaan grafik yang ditulis dalam bahasa pemrograman

Javascript murni. Pustaka ini relatif mudah untuk digunakan dalam sebuah website atau aplikasi website. Pustaka ini menyediakan tampilan grafik garis, area, kolom, lingkaran dan sebagainya.

Highcharts adalah pustaka yang digunakan untuk membuat grafik dari data sensor yang ada pada website.

2.22 Volley *Library* (Android)

Volley [38] merupakan pustaka HTTP yang digunakan untuk memudahkan pemrograman antar jaringan pada aplikasi Android. Pustaka ini diperkenalkan dengan tujuan utama untuk menanggulangi tidak tersedianya *networking class* pada Android SDK. Sebelum adanya Volley, *tools* yang tersedia untuk komunikasi antara *client* dan *remote backend* hanya menggunakan Http Connection (Java *class*) dan Apache Http Client.

Beberapa keunggulan adalah komputasinya yang lebih cepat, relatif lebih efisien digunakan, lebih cocok digunakan untuk pengiriman data-data kecil (tidak terlalu baik jika digunakan untuk *streaming* atau unduh berkas berukuran besar) [39].

Pustaka Volley digunakan untuk mengirimkan data dari aplikasi Android ke server.

2.23 SmsManager *Library* (Android)

SmsManager merupakan sebuah pustaka (*library*) yang ada pada Android. Fungsi utama SmsManager adalah untuk mengatur operasi sms. Hal-hal yang diakomodasi seperti mengirim sms, data dan juga membaca pesan teks pada Android. Pustaka ini memungkinkan pengguna untuk mengirimkan sms melalui sebuah perintah yang ada pada aplikasi yang dibuat [40].

Pustaka ini merupakan pustaka pada aplikasi Android yang digunakan untuk pengelolaan fungsi – fungsi sms.

2.24 Network Connection Class *Library* (Android)

Network Connection Class merupakan sebuah pustaka yang dapat digunakan untuk menentukan kecepatan koneksi internet. Pustaka ini bekerja dengan cara melakukan beberapa *request* untuk mengunduh berkas. Kecepatan internet dihitung berdasarkan rata – rata kecepatan data yang masuk. Pustaka ini dikembangkan oleh Facebook secara *open source* [18].

Pustaka ini merupakan pustaka untuk mengecek kecepatan koneksi internet di Android.

2.25 MySQLdb *Library* (Python)

MySQLdb merupakan sebuah pustaka tambahan yang dimiliki bahasa pemrograman Python. Pustaka ini tidak tersedia saat melakukan instalasi Python, sehingga dibutuhkan instalasi tersendiri sebelum dapat menggunakan pustaka MySQLdb. Pustaka ini berisi fungsi-fungsi yang dapat digunakan Python untuk berkomunikasi dengan basis data MySQL [41], [42].

Pustaka ini digunakan pada bahasa pemrograman Python untuk mengakses basis data MySQL.

2.26 Urllib2 *Library* (Python)

Urllib2 merupakan sebuah pustaka yang ada pada bahasa pemrograman Python. Pustaka ini dapat digunakan untuk mengakses sebuah URL. Penggunaan pustaka ini berbasis pada HTTP *request* dan *response*. Urllib2 menyediakan fungsi-fungsi dan kelas-kelas untuk membantu URL *action*, seperti *basic and digest*, *authentication*, *redirection*, *cookies* dan sebagainya [43], [44].

Pustaka ini digunakan untuk melakukan *request* http pada bahasa Python.

BAB III
PERANCANGAN PERANGKAT LUNAK

Bab ini membahas khusus mengenai analisis dan perancangan perangkat lunak yang akan dikembangkan. Secara teknis, aktivitas perancangan merupakan salah satu aktivitas yang sangat penting dilakukan dalam rangka pengembangan perangkat lunak. Beberapa hal yang secara umum dibahas dalam bab ini adalah deskripsi umum sistem, arsitektur umum sistem, diagram kasus penggunaan, perancangan basis data, diagram alur dan desain antarmuka perangkat lunak.

3.1 Deskripsi Umum Sistem

Pada Tugas Akhir ini akan dibangun suatu sistem *monitoring* server dan *gateway* adaptif yang berguna untuk memantau keadaan sensor-sensor yang berada pada lokasi yang berbeda - beda (tersebar). *Node-node* sensor ini dalam sistem akan ditampilkan secara berkelompok sesuai dengan lokasi *gateway* masing-masing. Pada penerapannya, ada 4 bagian (*node*) utama yang akan dibangun, yaitu server (*Sensor Cluster Management*) berbentuk website untuk *monitoring* sensor-sensor tersebar), *node gateway* (*gateway* adaptif dalam bentuk Android *smartphone*), *node* sms *receiver* (modem yang akan dijalankan dalam sebuah komputer untuk menerima dan mengolah sms) dan *node* sensor (menggunakan mikrokontroler Arduino).

Sensor Cluster Management yang ada pada server memiliki beberapa fungsi yang dapat digunakan untuk memantau sensor. Fungsi - fungsi yang ada pada website meliputi menampilkan sensor berdasarkan kelompok, mendaftarkan kelompok baru, mendaftarkan sensor baru, menampilkan lokasi peta lokasi keseluruhan sensor, menampilkan peta lokasi khusus (lokasi spesifik dari sebuah kelompok sensor) serta dapat menampilkan detail sensor.

Node gateway atau bisa disebut Android *gateway* (mengacu pada sistem operasi yang digunakan) merupakan *gateway* yang adaptif. Maksud adaptif di sini ialah, Android *gateway* dalam kaitannya dengan pengiriman data ke server dapat menggunakan dua cara yaitu melalui koneksi internet atau dengan menggunakan sms (pesan singkat). Penentuan cara pengiriman data ke server ditentukan dengan melihat kondisi ketersediaan koneksi internet. Saat koneksi internet tersedia maka pengiriman data sensor dari Android *gateway* ke server dilakukan dengan menggunakan koneksi internet. Sedangkan sebaliknya saat koneksi internet tidak tersedia, maka pengiriman data sensor dari Android *gateway* ke server dilakukan dengan mengirimkan sms.

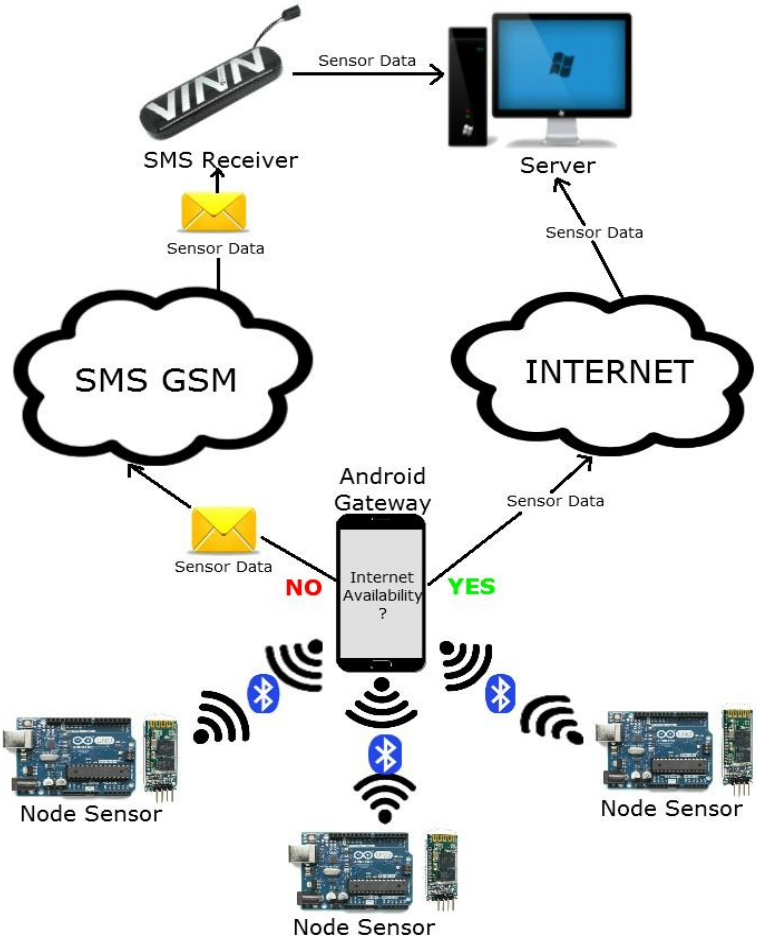
Jika melihat dari cara komunikasi atau pengiriman data yang menggunakan sms, maka diperlukan pula sebuah *node* tambahan untuk menerima dan mengelola sms. *Node* ini dapat disebut sms *receiver*. *Node* sms *receiver* berfungsi untuk menerima sms dari Android *gateway* untuk kemudian diteruskan ke server. *Node* ini dimungkinkan berupa sebuah modem yang dapat langsung dipasang pada komputer server atau bisa pula dipasang pada komputer yang berlainan.

Node terakhir adalah *node* sensor. *Node* ini akan dibangun menggunakan mikrokontroler Arduino UNO. *Node* sensor dapat berkomunikasi untuk melakukan pertukaran data dengan Android *gateway*. Komunikasi ini dilakukan menggunakan jaringan nirkabel. Sarana komunikasi nirkabel yang digunakan adalah Bluetooth. Sehingga pada *node* sensor ini akan digunakan sebuah modul HC-06 yang merupakan salah satu varian dari beberapa jenis modul Bluetooth yang tersedia di pasaran.

3.2 Arsitektur Umum Sistem

Alasan perancangan sistem seperti Gambar 3.1 adalah untuk menanggulangi data yang tidak terkirim ke server saat tidak ada koneksi internet. Berdasarkan perancangan arsitektur sistem pada

Gambar 3.1, ada dua sampai tiga *node* sensor yang mengirimkan data kepada Android *gateway*. *Node – node* sensor dan Android *gateway* ini membentuk sebuah jaringan yang disebut Piconet. *Node* sensor di sini bertindak sebagai *slave*. Sedangkan Android *gateway* bertindak sebagai *master*.



Gambar 3.1 Arsitektur Umum Sistem

Node sensor akan mengambil data berdasarkan lingkungan di sekitarnya. Sensor yang digunakan adalah sensor suhu, kelembaban, cahaya dan suara. Setelah data dari sensor berhasil didapatkan, kemudian data sensor akan dikirimkan ke Android *gateway* melalui komunikasi Bluetooth. Android *gateway* kemudian mengirimkan data tersebut ke server. Di sinilah Android *gateway* berperan adaptif dalam penentuan cara pengiriman data sensor ke server. Apabila koneksi internet tersedia, maka setiap sepuluh menit Android *gateway* akan mengirimkan data sensor terakhir yang diterima dari *node* sensor untuk kemudian diteruskan ke server melalui koneksi internet. Jika koneksi internet tidak tersedia, maka pengiriman data akan ditunda hingga tiga kali sepuluh menit (akumulasi 30 menit). Jika pada sepuluh menit ketiga koneksi internet masih tidak tersedia, maka pengiriman data ke server akan dilakukan melalui sms.

Data yang dikirim melalui sms akan diterima oleh sms *receiver*. Setelah data diterima, sms *receiver* akan mengolah data sesuai format pesan yang telah dibuat. Jika sms yang dikirimkan sesuai dengan format, maka data akan diteruskan ke server.

Server akan menampung data dan memasukkannya pada basis data yang ada. Berdasarkan pada basis data inilah, server akan mengolah data yang ada untuk kemudian ditampilkan dalam bentuk tampilan sebuah website agar lebih mudah dipahami.

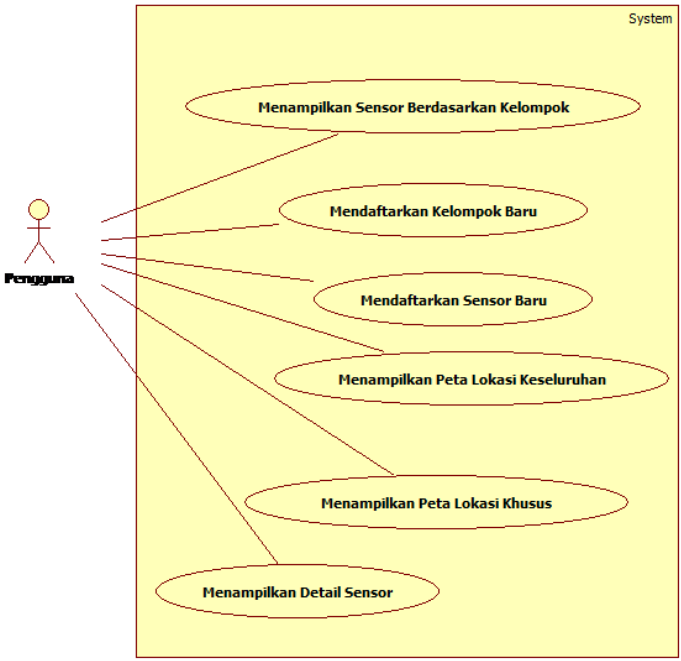
3.3 Perancangan *Sensor Cluster Management* (Server)

Sensor Cluster Management yang akan dibangun merupakan sebuah website untuk memonitor atau memantau keadaan lingkungan melalui data – data yang dikirimkan oleh *node* sensor. *Node – node* sensor yang telah didaftarkan akan ditampilkan pada website dalam kelompok – kelompok tertentu sesuai dengan *node gateway* yang terhubung dengan *node* sensor.

Kegunaan lain yang dimiliki website ada beberapa hal. Beberapa hal tersebut meliputi, menampilkan sensor berdasarkan

kelompok, mendaftarkan kelompok baru, mendaftarkan sensor baru, menampilkan lokasi peta lokasi keseluruhan sensor, menampilkan peta lokasi khusus (lokasi spesifik dari sebuah kelompok sensor) serta dapat menampilkan detail sensor.

3.3.1 Perancangan Diagram Kasus Penggunaan



Gambar 3.2 Diagram Kasus Penggunaan

Pada bagian ini akan dijelaskan rincian kasus penggunaan (*use case*) yang akan dibangun dalam website. Kasus penggunaan yang akan dibuat dalam website terdiri dari beberapa hal yaitu menampilkan sensor berdasarkan kelompok, mendaftarkan kelompok baru, mendaftarkan sensor baru, menampilkan peta lokasi keseluruhan, menampilkan peta lokasi khusus dan

menampilkan detail sensor. Gambar 3.2 merupakan gambar diagram kasus penggunaan (*use case*). Berdasarkan diagram kasus penggunaan yang telah dibuat, berikut ini rincian dari setiap kasus penggunaan yang ada :

3.3.1.1 Menampilkan Sensor Berdasarkan Kelompok

Kasus penggunaan ini merupakan halaman utama (*dashboard*) dari Pengguna. Pengguna dapat menampilkan seluruh sensor yang telah didaftarkan. Sensor – sensor ini dikelompokkan (*cluster*) sesuai dengan kelompok (*group*) yang telah didaftarkan sebelumnya. Pada kasus penggunaan ini juga ditampilkan beberapa hal lain seperti status keaktifan sensor, data – data sensor (suhu, kelembaban udara, intensitas cahaya, intensitas suara), waktu *update* terakhir, koneksi yang digunakan. Selain itu juga ditampilkan lokasi (dalam bentuk *latitude* dan *longitude*) *gateway*.

3.3.1.2 Mendaftarkan Kelompok Baru

Pada kasus penggunaan yang satu ini, Pengguna dapat mendaftarkan kelompok baru dengan cara memilih menu yang tersedia. Agar Pengguna dapat mendaftarkan kelompok baru, mula – mula yang harus dilakukan adalah mengisi *form* yang tersedia dengan memasukkan detail dari kelompok yang akan dibuat. *Form* yang harus diisi adalah nama kelompok (*group name*) dan deskripsi kelompok (*group description*). Kelompok baru yang didaftarkan akan muncul pada halaman utama dan juga pada pilihan saat mendaftarkan sensor baru.

3.3.1.3 Mendaftarkan Sensor Baru

Setelah Pengguna mendaftarkan kelompok baru (memiliki kelompok untuk tempat mendaftarkan sensor), di sini Pengguna dapat mendaftarkan sensor baru dengan cara memilih menu yang untuk mendaftarkan sensor baru. Pengguna yang akan mendaftarkan sensor baru, yang harus dilakukan pertama adalah

mengisi *form* yang tersedia dengan memasukkan detail dari sensor yang akan didaftarkan. Rincian *form* yang harus diisi antara lain adalah nama sensor (*sensor name*), deskripsi sensor (*sensor description*), pilihan kelompok dan tipe – tipe sensor yang akan dipantau (suhu, kelembaban udara, intensitas cahaya, intensitas suara). Sensor baru yang didaftarkan akan muncul pada halaman utama sesuai dengan kelompoknya.

3.3.1.4 Menampilkan Peta Lokasi Keseluruhan

Kasus penggunaan ini merupakan halaman untuk menampilkan peta lokasi dari keseluruhan kelompok-kelompok sensor yang dimiliki. Pengguna dapat menampilkan seluruh kelompok-kelompok sensor yang dimiliki sesuai lokasinya pada sebuah peta. Saat *marker* lokasi ditekan, maka akan menampilkan beberapa data terkait dengan nama kelompok sensor, *latitude*, *longitude*, *node – node* sensor yang tersambung beserta sedikit detail tentang *node* sensor tersebut).

3.3.1.5 Menampilkan Peta Lokasi Khusus

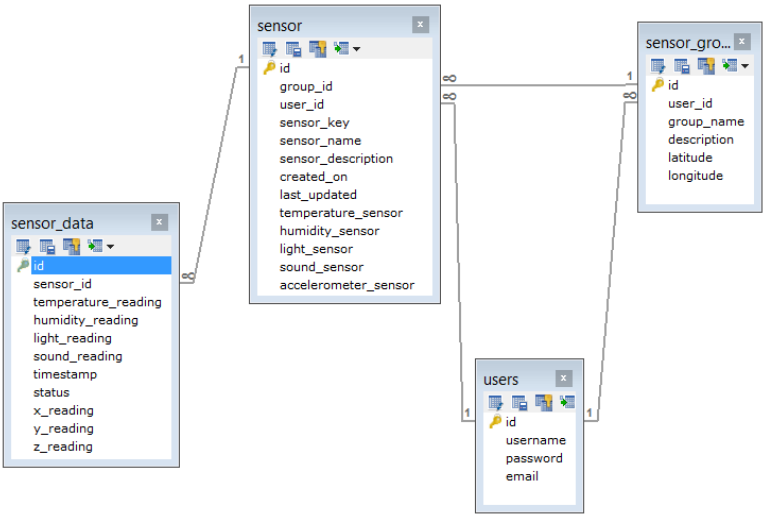
Sedikit berbeda dengan kasus penggunaan sebelumnya yang menampilkan keseluruhan lokasi. Kasus penggunaan ini adalah sebuah halaman untuk menampilkan peta lokasi spesifik dari satu kelompok sensor.

Pengguna dapat menampilkan khusus lokasi spesifik dari satu kelompok sensor yang dipilih. Sehingga menampilkan peta lokasi yang lebih dekat (*zoom-in*) pada lokasi kelompok sensor tertentu. Sama dengan sebelumnya, saat *marker* lokasi ditekan, maka akan menampilkan beberapa data terkait dengan nama kelompok sensor, *latitude*, *longitude*, *node – node* sensor yang tersambung beserta sedikit detail tentang *node* sensor tersebut).

3.3.1.6 Menampilkan Detail Sensor

Kasus penggunaan ini merupakan sebuah halaman yang menampilkan detail dari sebuah sensor. Melalui sebuah tombol ‘detail’ yang ada pada halaman utama, Pengguna dapat menampilkan detail dari sensor yang dipilih. Rincian data yang ditampilkan dalam kasus penggunaan ini meliputi informasi sensor (nama, deskripsi, *id*, *key*, nama *group*), grafik sensor data secara *realtime* (suhu, kelembaban udara, intensitas cahaya, intensitas suara) dan riwayat seluruh data sensor yang pernah dikirimkan.

3.3.2 Perancangan Basis Data (*Sensor Cluster Management*)

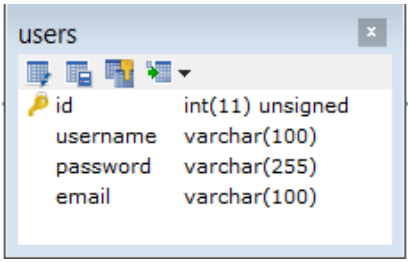


Gambar 3.3 Skema Basis Data (*Sensor Cluster Management*)

Perancangan basis data merupakan satu tahap untuk merancang basis data yang akan digunakan untuk menampung data – data yang diperlukan. Gambar 3.3 merupakan rancangan skema dari basis data. Basis data inilah yang nantinya akan diolah dan ditampilkan dalam website. Sistem manajemen basis data yang

akan digunakan ialah MySQL. Beberapa hal yang akan disimpan dalam basis data adalah data – data mengenai Pengguna dan data – data sensor (detail sensor, kelompok sensor maupun data hasil pembacaan sensor). Berikut ini penjelasan lebih rinci mengenai setiap entitas dan atribut dalam basis data.

3.3.2.1 Tabel *users*



Gambar 3.4 Skema Tabel *users*

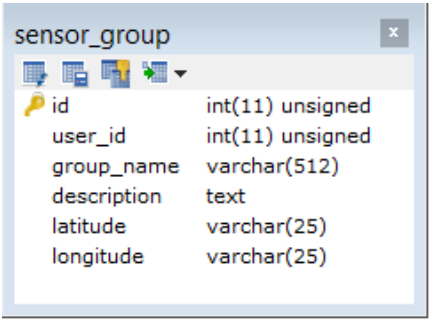
Gambar 3.4 adalah skema dari tabel *users*. Fungsi dari tabel (*entity*) ini adalah untuk menyimpan data – data dari Pengguna. Data Pengguna yang disimpan di dalamnya meliputi *id*, *username*, *password* dan *email*. Tabel 3.1 merupakan tabel yang menjelaskan detail dari tabel *users*.

Tabel 3.1 Detail Tabel *users*

No	Nama Atribut	Tipe Data	Keterangan
1.	*id*	*integer*	Merupakan *primary key* dari tabel *users*. Diatur sebagai *auto increment*
2.	*username*	*varchar(100)*	*username* dari Pengguna yang digunakan sebagai pengenal untuk masuk (*login*) ke dalam website
3.	*password*	*varchar(255)*	*password* dari Pengguna yang digunakan sebagai kode pengaman untuk masuk (*login*) ke dalam website

No	Nama Atribut	Tipe Data	Keterangan
4.	*email*	*varchar(100)*	Merupakan *email* didaftarkan saat proses registrasi

3.3.2.2 Tabel *sensor_group*



Gambar 3.5 Skema Tabel *sensor_group*

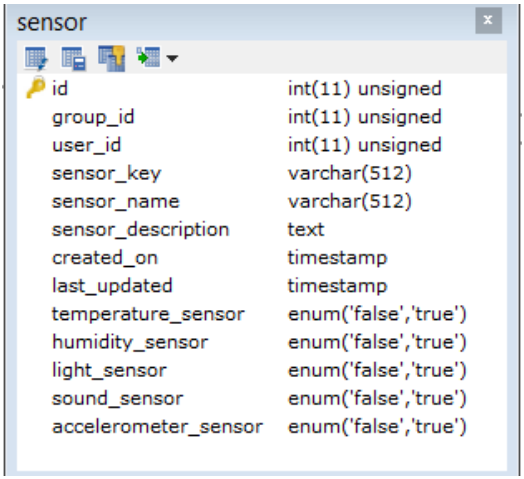
Gambar 3.5 adalah skema dari tabel *sensor_group*. Fungsi dari tabel (*entity*) ini adalah untuk menyimpan data – data mengenai kelompok sensor. Data yang disimpan di dalamnya meliputi *id*, *user_id*, *group_name*, *description*, *latitude* serta *longitude*. Data ini didapatkan ketika proses pendaftaran kelompok sensor baru. Sedangkan data lokasi (*latitude*, *longitude*) diperbarui secara berkala untuk merekam lokasi keberadaan kelompok sensor. Tabel 3.2 merupakan tabel yang menjelaskan detail dari tabel *sensor_group*.

Tabel 3.2 Detail Tabel *sensor_group*

No	Nama Atribut	Tipe Data	Keterangan
1.	*id*	*integer*	Merupakan *primary key* dari tabel *sensor_group*. Diatur sebagai *auto increment*
2.	*user_id*	*integer*	Merupakan *foreign key* yang terhubung dengan tabel *users*. Satu *users* dapat

No	Nama Atribut	Tipe Data	Keterangan
			memiliki banyak *sensor_group* (*one to many*).
3.	*group_name*	*varchar(512)*	Nama dari kelompok sensor yang didaftarkan
4.	*description*	*text*	Deskripsi dari kelompok sensor yang didaftarkan
5.	*latitude*	*varchar(25)*	Lokasi dari kelompok sensor yang menyimpan nilai *latitude*
6.	*longitude*	*varchar(25)*	Lokasi dari kelompok sensor yang menyimpan nilai *longitude*

3.3.2.3 Tabel *sensor*



Gambar 3.6 Skema Tabel *sensor*

Gambar 3.6 merupakan skema dari tabel *sensor*. Fungsi dari tabel (*entity*) ini adalah untuk menyimpan data – data mengenai detail sensor. Data sensor yang disimpan tersebut meliputi *id*, *group_id*, *user_id*, *sensor_key*, *sensor_name*, *sensor_decription*, *created_on*, *last_updated*, *temperature_sensor*, *humidity_sensor*,

light_sensor, *sound_sensor* dan *accelerometer_sensor*. Data sensor didapatkan saat proses pendaftaran sensor baru. Tabel 3.3 merupakan tabel yang menjelaskan detail dari tabel *sensor*.

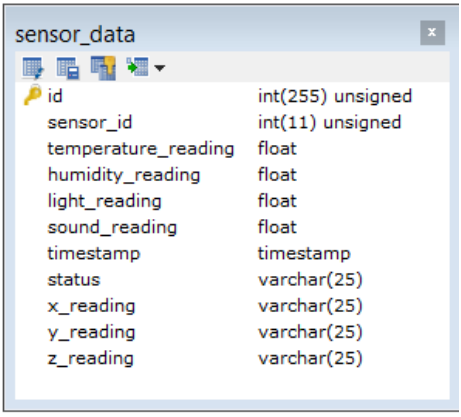
Tabel 3.3 Detail Tabel *sensor*

No	Nama Atribut	Tipe Data	Keterangan
1	*id*	*integer*	Merupakan *primary key* dari tabel *sensor*. Diatur sebagai *auto increment*
2.	*group_id*	*integer*	Merupakan *foreign key* yang terhubung dengan tabel *sensor_group*. Satu *sensor_group* dapat memiliki banyak *sensor* (*one to many*).
3.	*user_id*	*integer*	Merupakan *foreign key* yang terhubung dengan tabel *users*. Satu *users* dapat memiliki banyak *sensor* (*one to many*).
4	*sensor_key*	*varchar(512)*	*sensor_key* merupakan atribut yang berfungsi untuk menyimpan sebuah *string* unik pada setiap sensor yang ada. Nilai dari *sensor_key* dibuat dengan menggabungkan nama sensor, *current_timestamp* dan *user_id*, untuk kemudian di enkripsi dengan *hash function* SHA1.
5.	*sensor_name*	*varchar(512)*	Nama dari sensor yang didaftarkan

No	Nama Atribut	Tipe Data	Keterangan
6.	*sensor_description*	*text*	Deskripsi dari sensor yang didaftarkan
7.	*created_on*	*timestamp*	Merupakan atribut yang menyimpan waktu dan tanggal pembuatan sensor. Memiliki nilai *default* yaitu sama dengan waktu saat itu (*current_timestamp*)
8.	*last_updated*	*timestamp*	Merupakan atribut yang menyimpan waktu dan tanggal terakhir sensor menerima data. Memiliki nilai *default* yaitu sama dengan waktu saat itu (*current_timestamp*). Nilai ini akan diperbarui seiring dengan masuknya data sensor
9.	*temperature_sensor*	*enum ('true', 'false')*	Merupakan penanda (*flag*) untuk menandakan apakah sensor suhu dimiliki oleh suatu *node* sensor atau tidak
10.	*humidity_sensor*	*enum ('true', 'false')*	Merupakan penanda (*flag*) untuk menandakan apakah sensor kelembaban udara dimiliki oleh suatu *node* sensor atau tidak
11.	*light_sensor*	*enum ('true', 'false')*	Merupakan penanda (*flag*) untuk

No	Nama Atribut	Tipe Data	Keterangan
			menandakan apakah sensor cahaya dimiliki oleh suatu *node* sensor atau tidak
12.	*sound_sensor*	*enum ('true', 'false')*	Merupakan penanda (*flag*) untuk menandakan apakah sensor suara dimiliki oleh suatu *node* sensor atau tidak
13.	*accelerometer_sensor*	*enum ('true', 'false')*	Merupakan penanda (*flag*) untuk menandakan apakah sensor akselerometer dimiliki oleh suatu *node* sensor atau tidak

3.3.2.4 Tabel *sensor_data*



Gambar 3.7 Skema Tabel *sensor_data*

Gambar 3.7 merupakan skema dari tabel *sensor_data*. Fungsi dari tabel (*entity*) ini adalah untuk menyimpan data – data mengenai yang didapat dari sensor. Data dari sensor yang disimpan

tersebut meliputi *id*, *sensor_id*, *temperature_reading*, *humidity_reading*, *light_reading*, *sound_reading*, *timestamp*, *status*, *x_reading*, *y_reading* dan *z_reading*. Data ini diperoleh secara berkala dari pengambilan data pada *node* sensor. Tabel 3.4 merupakan tabel yang menjelaskan detail dari tabel *sensor_data*.

Tabel 3.4 Detail Tabel *sensor_data*

No	Nama Atribut	Tipe Data	Keterangan
1	*id*	*integer*	Merupakan *primary key* dari tabel *sensor_data* Diatur sebagai *auto increment*
2.	*sensor_id*	*integer*	Merupakan *foreign key* yang terhubung dengan tabel *sensor*. Satu *sensor* dapat memiliki banyak *sensor_data* (-*one to many*).
3.	*temperature_reading*	*float*	Berisi data tentang suhu yang didapatkan dari *node* sensor
4.	*humidity_reading*	*float*	Berisi data tentang kelembaban udara yang didapatkan dari *node* sensor
5.	*light_reading*	*float*	Berisi data tentang intensitas cahaya yang didapatkan dari *node* sensor
6.	*sound_reading*	*float*	Berisi data tentang intensitas suara yang didapatkan dari *node* sensor
7.	*timestamp*	*timestamp*	Merupakan atribut yang menyimpan waktu dan tanggal data sensor masuk. Memiliki nilai *default* yaitu sama

No	Nama Atribut	Tipe Data	Keterangan
			dengan waktu saat itu (*current_timestamp*).
8.	*status*	*varchar(25)*	Atribut ini berisi data yang menunjukkan jenis jenis koneksi yang digunakan untuk mengirimkan data ke server, yaitu melalui sms atau internet
9.	*x_reading*	*varchar(25)*	Berisi data tentang data x sensor akselerometer yang didapatkan Android *gateway*
10.	*y_reading*	*varchar(25)*	Berisi data tentang data y sensor akselerometer yang didapatkan dari Android *gateway*
11.	*z_reading*	*varchar(25)*	Berisi data tentang data z sensor akselerometer yang didapatkan dari Android *gateway*

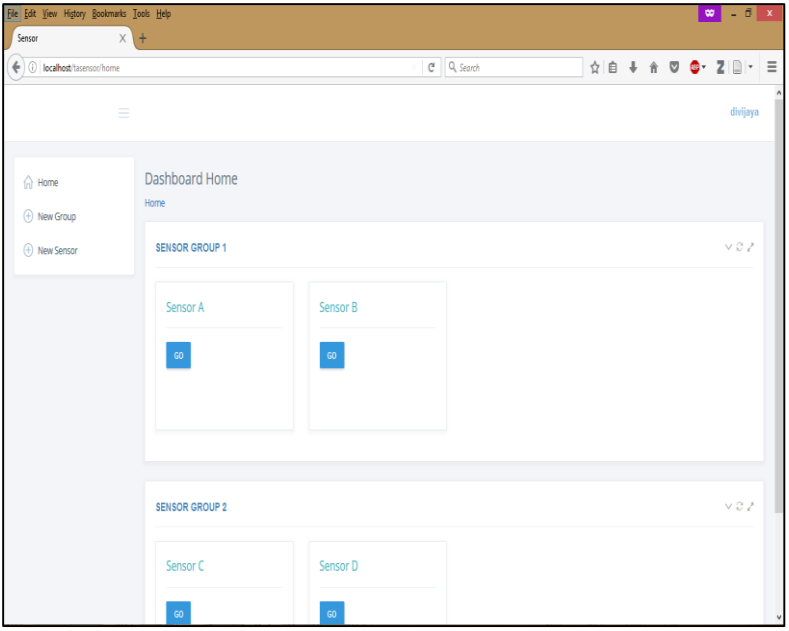
3.3.3 Perancangan Antarmuka Pengguna (*Sensor Cluster Management*)

Perancangan antarmuka pengguna merupakan satu tahap yang dilakukan untuk persiapan dalam pembuatan antarmuka website agar lebih mudah dalam pelaksanaan proses implementasi. Antarmuka website akan dibuat dengan memanfaatkan Bootstrap.

Website yang akan dibangun memiliki beberapa halaman penting. Antarmuka halaman yang akan dibuat meliputi halaman utama (menampilkan sensor dalam kelompok – kelompok), halaman *form* pendaftaran kelompok baru, halaman *form* pendaftaran sensor baru, halaman peta lokasi (keseluruhan maupun khusus) dan halaman detail sensor.

3.3.3.1 Halaman Utama

Halaman ini merupakan halaman utama yang akan ditampilkan saat Pengguna masuk (*login*). Mengacu pada bab 3.3.1.1, pada halaman ini akan ditampilkan sensor – sensor berdasarkan kelompoknya masing – masing. Gambar 3.8 adalah rancangan antarmuka dari halaman utama.

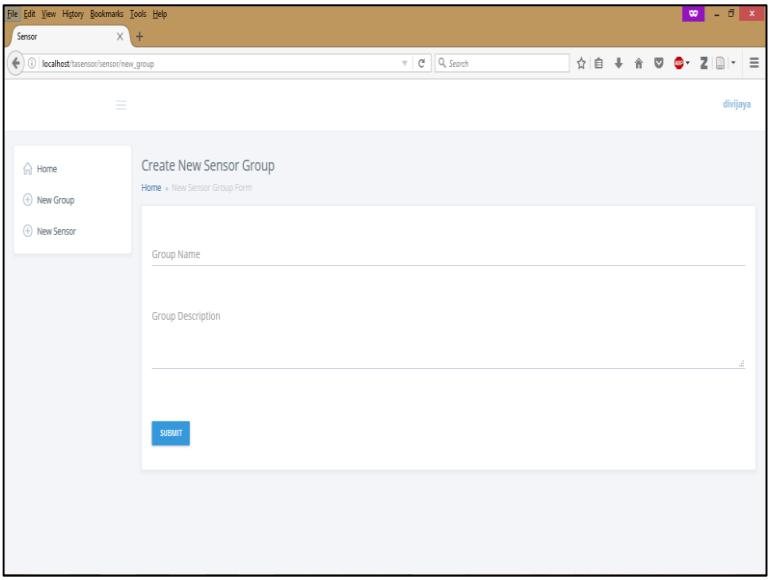


Gambar 3.8 Rancangan Antarmuka Halaman Utama

Data sensor yang ditampilkan pada halaman utama adalah data – data seperti data sensor terakhir yang masuk, koneksi yang digunakan (sms atau internet), waktu *update* terakhir serta status aktif tidaknya sensor. Pada halaman ini ditampilkan juga lokasi (*latitude* dan *longitude*) dari kelompok sensor.

3.3.3.2 Halaman *Form New Group*

Halaman ini berhubungan dengan kasus kegunaan pada bab 3.3.1.2, yang akan menampilkan sebuah isian (*form*) untuk mendaftarkan kelompok baru. Melalui menu '*new group*', Pengguna akan diarahkan pada halaman pendaftaran kelompok sensor baru. Data yang harus diisikan untuk mendaftarkan kelompok sensor baru adalah nama kelompok (*group name*) dan deskripsi kelompok (*group description*). Gambar 3.9 merupakan rancangan antarmuka halaman *form new group*.

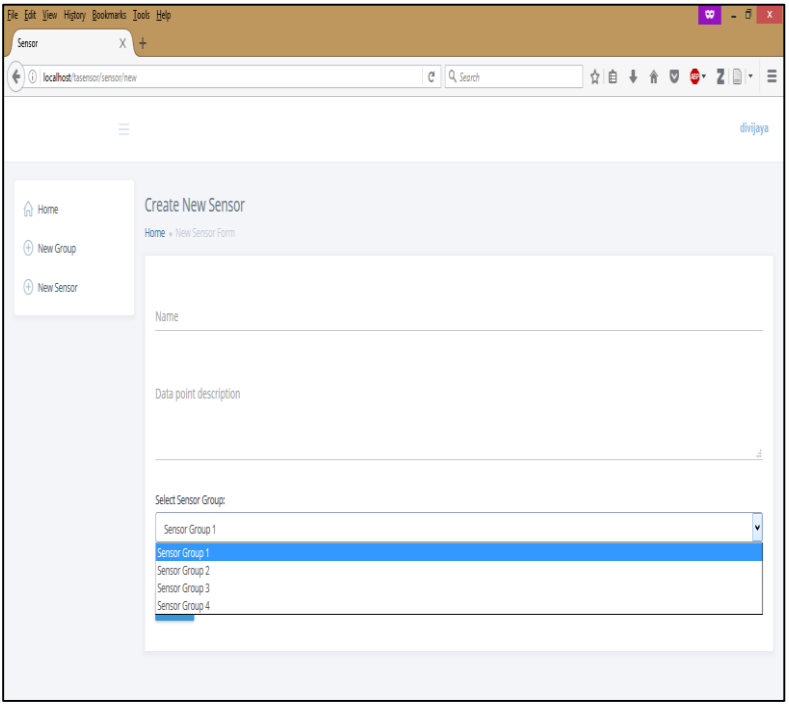


Gambar 3.9 Rancangan Antarmuka Halaman *Form New Group*

3.3.3.3 Halaman *Form New Sensor*

Mengacu pada kasus kegunaan di bab 3.3.1.3, pada halaman ini akan ditampilkan sebuah isian (*form*) untuk mendaftarkan sensor baru. Melalui menu '*new sensor*', Pengguna akan diarahkan pada halaman pendaftaran sensor baru. Data yang harus diisikan

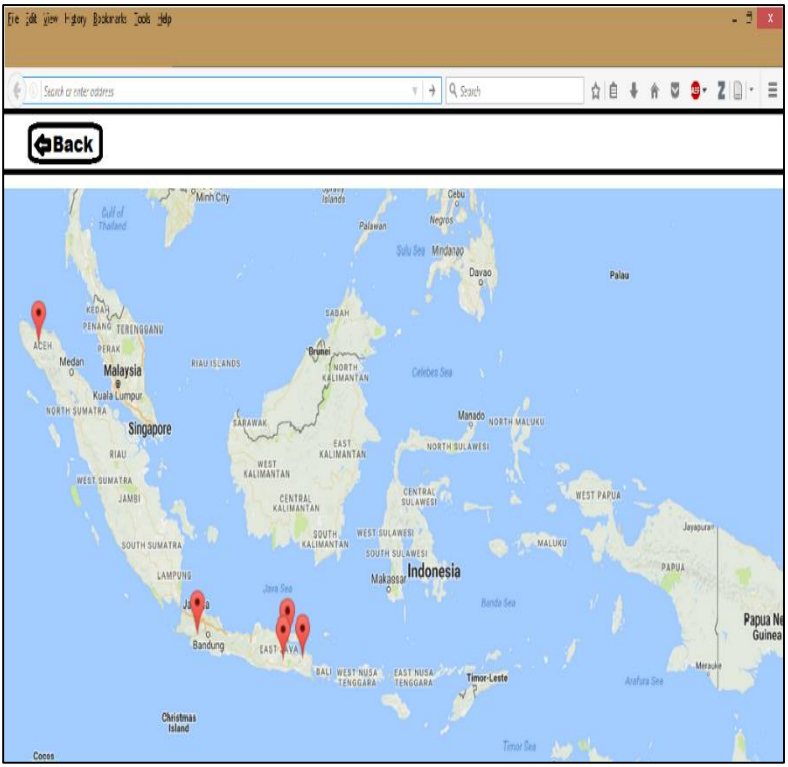
untuk mendaftarkan sensor baru adalah nama sensor (*sensor name*), deskripsi sensor (*sensor description*), pilihan kelompok dan tipe-tipe sensor. Gambar 3.10 merupakan rancangan antarmuka halaman *form new sensor*.



Gambar 3.10 Rancangan Antarmuka Halaman *Form New Sensor*

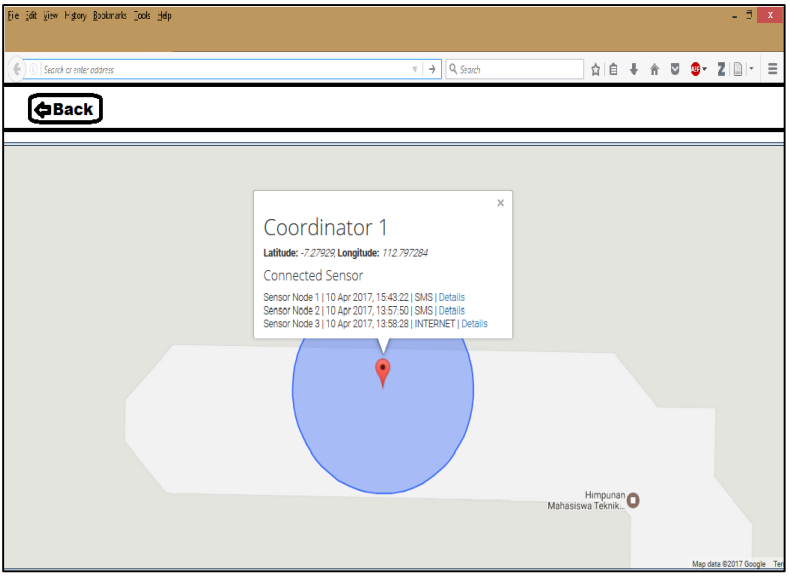
3.3.3.4 Halaman Peta Lokasi (*Maps*)

Berdasarkan kasus kegunaan pada bab 3.3.1.4, di halaman peta lokasi yang pertama akan ditampilkan sebuah peta yang menunjukkan lokasi keberadaan dari kelompok – kelompok sensor. Lokasi dari kelompok sensor ditunjukkan dengan sebuah (*marker*). Gambar 3.11 merupakan rancangan halaman peta lokasi keseluruhan kelompok sensor.



Gambar 3.11 Rancangan Antarmuka Halaman Lokasi Keseluruhan

Peta lokasi kedua (merujuk pada kasus kegunaan di bab 3.3.1.5) merupakan peta lokasi khusus yang menampilkan lokasi spesifik dari suatu kelompok sensor. Lokasi dari kelompok sensor ditunjukkan dengan sebuah (*marker*). Saat *marker* ditekan, maka akan menampilkan sedikit detail tentang lokasi dan keadaan sensor. Gambar 3.12 merupakan rancangan antarmuka halaman lokasi kelompok sensor secara spesifik.

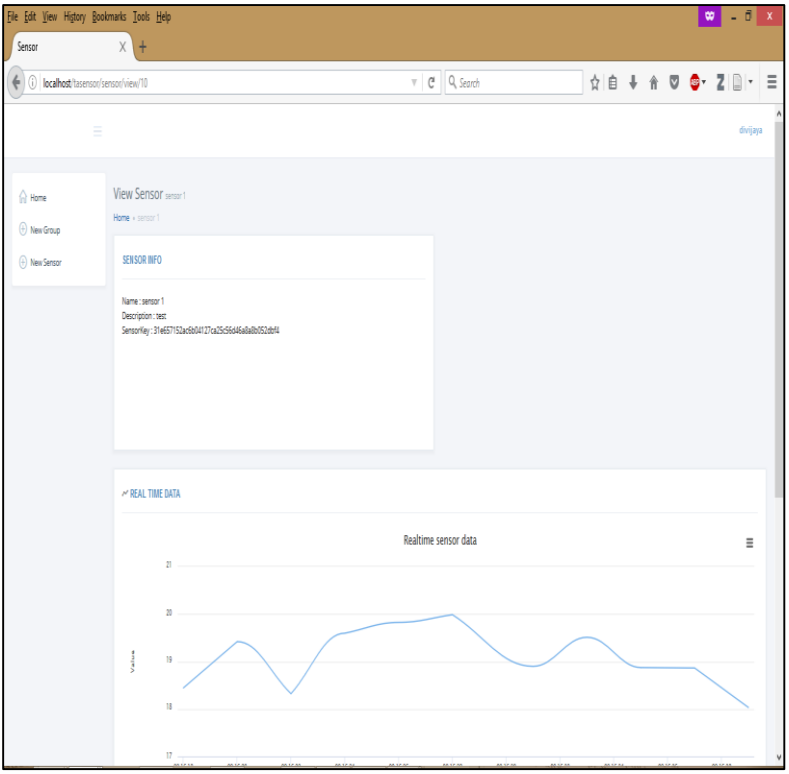


Gambar 3.12 Rancangan Antarmuka Halaman Lokasi Khusus

3.3.3.5 Halaman Detail Sensor

Jika dilihat dari kasus kegunaan di bab 3.3.1.6, pada halaman ini akan ditampilkan seluruh data sensor yang ada. Bagian teratas dari halaman ini menampilkan informasi sensor yang meliputi nama sensor (*sensor name*), sensor *id*, deskripsi sensor (*sensor description*), *key* dan nama kelompok *(group name*).

Terletak di bawahnya adalah tampilan dari grafik data sensor yang ditampilkan secara *realtime*. Grafik ini akan menampilkan data sensor baru yang masuk ke server. Bagian terakhir yang ditampilkan pada halaman ini adalah riwayat pengiriman data (*data history*). Riwayat yang ditampilkan merupakan data – data seperti nilai sensor data (*suhu, humidity, light, sound*), waktu dan jenis koneksi yang digunakan (internet atau sms). Gambar 3.13 merupakan rancangan antarmuka halaman detail sensor.



Gambar 3.13 Rancangan Antarmuka Halaman Detail Sensor

3.4 Perancangan Android *Gateway*

Node gateway atau bisa disebut Android (mengacu pada sistem operasi yang digunakan) *gateway* merupakan *gateway* yang adaptif. Maksud adaptif di sini ialah, Android *gateway* dalam kaitannya dengan pengiriman data ke server dapat menggunakan dua cara yaitu melalui koneksi internet atau dengan menggunakan sms (pesan singkat). Penentuan cara pengiriman data ke server ditentukan dengan melihat kondisi ketersediaan koneksi internet. Saat koneksi internet tersedia maka pengiriman data sensor dari Android *gateway* ke server dilakukan dengan menggunakan

koneksi internet. Sedangkan sebaliknya saat koneksi internet tidak tersedia, maka pengiriman data sensor dari Android *gateway* ke server dilakukan dengan mengirimkan sms.

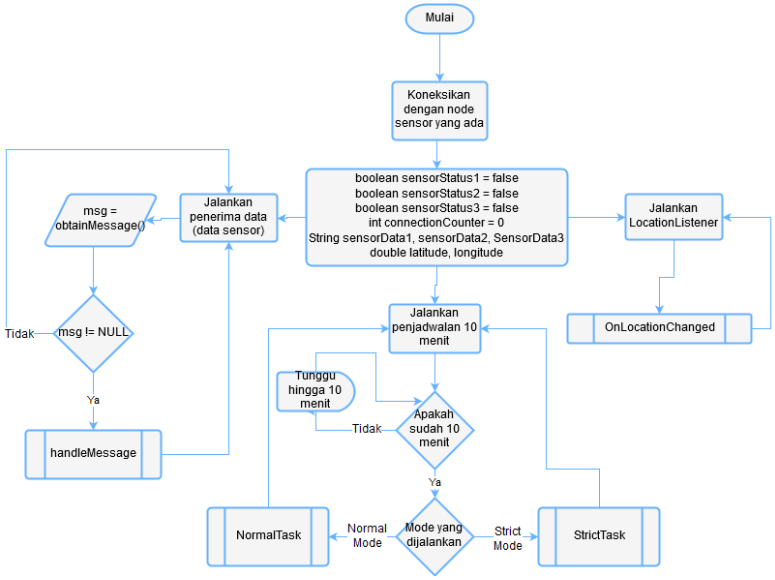
Di sinilah Android *gateway* berperan adaptif dalam penentuan cara pengiriman data sensor ke server. Ada dua mode yang dapat dipilih untuk berjalannya penjadwalan yaitu *normal mode* dan *strict mode*. Apabila koneksi internet tersedia, maka setiap sepuluh menit Android *gateway* akan mengirimkan data sensor terakhir yang diterima dari *node* sensor untuk kemudian diteruskan ke server melalui koneksi internet. Jika koneksi internet tidak tersedia, maka untuk pemilihan *normal mode* pengiriman data akan ditunda hingga dua kali sepuluh menit berikutnya (akumulasi 30 menit). Jika pada sepuluh menit ketiga koneksi internet masih tidak tersedia, maka pengiriman data ke server akan dilakukan melalui sms. Sedangkan untuk pemilihan *strict mode*, jika tidak ada koneksi internet maka data akan langsung dikirim menggunakan sms tanpa menunggu 30 menit.

3.4.1 Perancangan Diagram Alir (Android *Gateway*)

Pada subbab ini, akan dibahas alur dari sistem yang akan berjalan pada Android *gateway*. Perancangan aliran data sistem pada *gateway* secara keseluruhan dilakukan agar dapat lebih mudah memahami jalannya *gateway* secara menyeluruh. Selain itu digunakan juga untuk menunjukkan fungsi – fungsi utama atau aliran logika jalannya *gateway*.

Secara garis besar, sistem yang berjalan dalam Android *gateway* terdiri dari beberapa subproses penting yaitu *handleMessage*, *NormalTask*, *StrictMode* dan *OnLocationChanged*. Sebelum proses berjalan atau saat aplikasi baru dijalankan, mula – mula *gateway* akan melakukan pencarian jaringan Bluetooth yang ada di sekitarnya. Koneksi dilakukan sesuai dengan alamat (*hardware address*) dari masing perangkat Bluetooth yang ada pada *node* sensor. Setelah terhubung, barulah

ketiga proses lainnya akan berjalan. Gambar 3.14 merupakan gambaran umum jalannya proses secara keseluruhan dalam bentuk sebuah diagram alir.

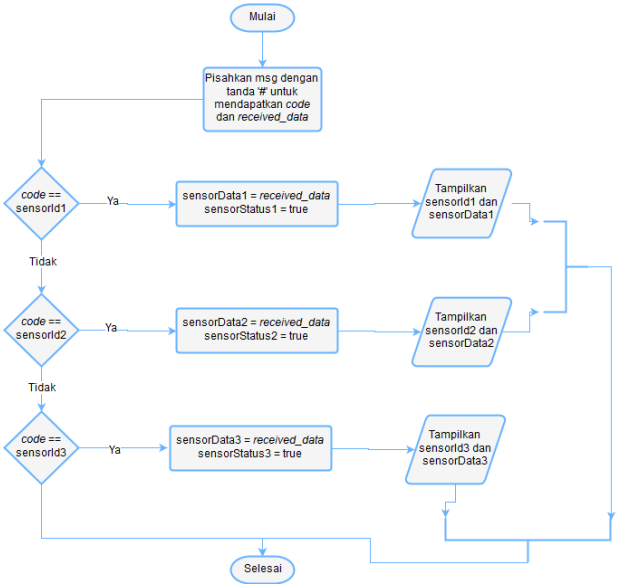


Gambar 3.14 Diagram Alir Sistem (Android *Gateway*)

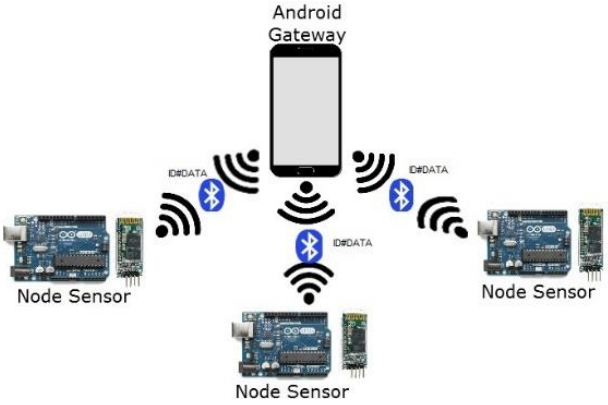
3.4.1.1 Diagram Alir Subproses *handleMessage*

Proses yang terjadi pada subproses *handleMessage* ini dimulai dengan melakukan inisialisasi data yang didapatkan dari *node* sensor. Data tersebut disimpan dalam sebuah *variable* bernama *msg*. Susunan data pada variabel *msg* adalah '*ID#DATA*'. '*ID*' merupakan representasi dari *sensor_id* yang memiliki nilai sesuai dengan *sensor_id* dari masing – masing sensor. Sedangkan '*DATA*' merupakan nilai dari sensor yang telah didapatkan berdasarkan keadaan lingkungan (rincian susunan '*DATA*' akan dijelaskan pada pembahasan selanjutnya). Antara '*ID*' dan '*DATA*' dipisahkan dengan tanda pagar ('#') untuk lebih mudah dalam proses pengolahan selanjutnya. Gambar 3.15 merupakan diagram

alir subproses *handleMessage* dan Gambar 3.16 adalah ilustrasi komunikasi antara *node* sensor dengan Android *gateway*.



Gambar 3.15 Diagram Alir Subproses *handleMessage*



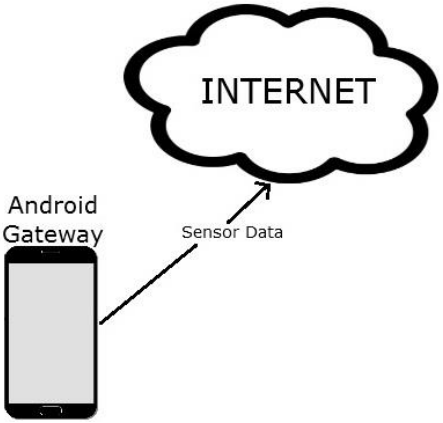
Gambar 3.16 Ilustrasi Komunikasi antara *Node* Sensor dengan Android *Gateway*

3.4.1.2 Diagram Alir Subproses *NormalTask*

Gambar 3.17 Diagram Alir Subproses *NormalTask*

Pada subproses *NormalTask* diatur mengenai mekanisme pengiriman data sensor ke server. Gambar 3.17 menunjukkan diagram alir subproses *NormalTask*. Penentuan cara pengiriman data ke server ditentukan dengan melihat kondisi ketersediaan koneksi internet. Saat koneksi internet tersedia maka pengiriman data sensor dari Android *gateway* ke server dilakukan dengan menggunakan koneksi internet. Sedangkan sebaliknya saat koneksi internet tidak tersedia, maka pengiriman data sensor dari Android *gateway* ke server dilakukan dengan mengirimkan sms.

Variabel *connectionCounter* digunakan sebagai penanda yang menentukan waktu. Setiap pertambahan satu nilai dari *connectionCounter* maka sama artinya dengan 10 menit. Nilai dari *connectionCounter* ini akan bertambah satu per satu setiap 10 menit jika tidak ada koneksi internet. Jika nilainya mencapai lebih dari 2 maka artinya tidak ada koneksi internet dalam waktu 30 menit, sehingga dilakukan pengiriman data melalui sms. Gambar 3.18 merupakan ilustrasi pengiriman data sensor dari Android *gateway* melalui internet.



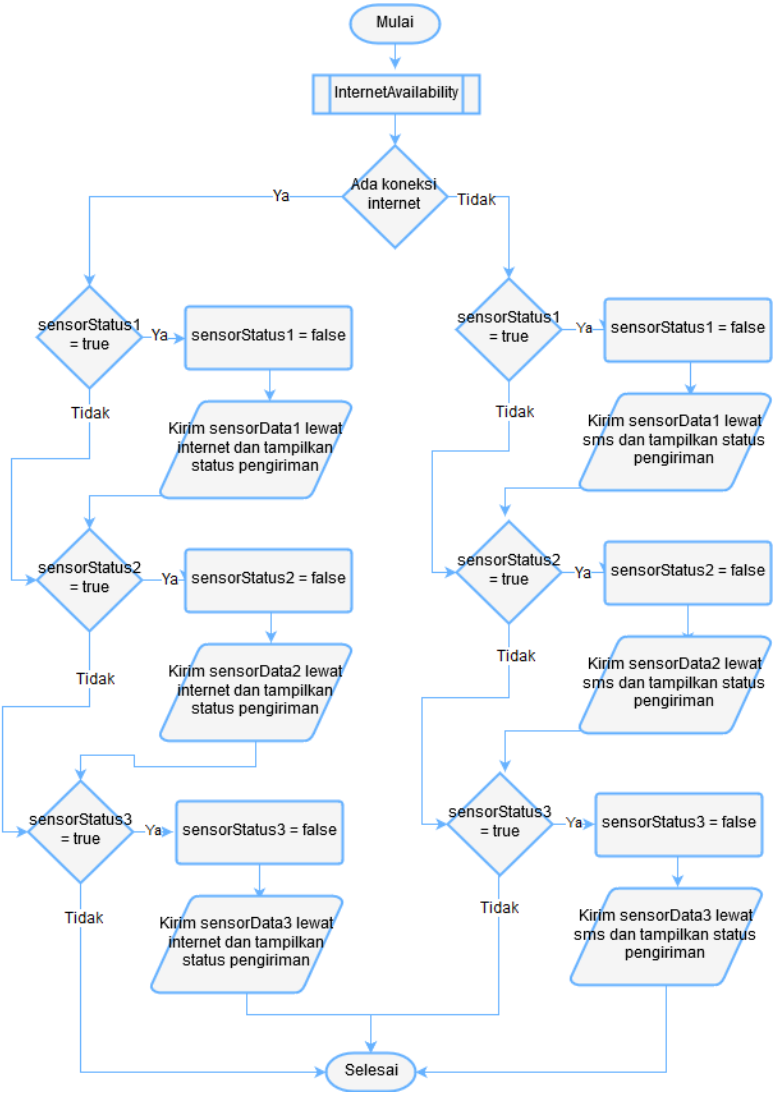
Gambar 3.18 Ilustrasi Pengiriman Data Sensor (Internet)

Sebelum melakukan pengiriman data, dilakukan pengecekan *sensorStatus* untuk menentukan apakah *node* sensor aktif atau tidak. Jika *node* sensor tidak aktif, maka tidak dilakukan pengiriman data dari sensor tersebut. Hal ini dilakukan untuk menghindari pengiriman data dari *node* sensor yang tidak aktif. Setiap akan melakukan pengiriman data baik melalui internet maupun sms, tidak lupa pula untuk mengembalikan nilai dari *connectionCounter* menjadi 0.

Pengiriman sms memiliki susunan (*format*) tersendiri. Susunan pengiriman sms adalah '*#KEY#DATA*'. '*KEY*' merupakan representasi dari *sensor_key*. Sedangkan '*DATA*' merupakan nilai dari sensor yang telah didapatkan berdasarkan keadaan lingkungan (rincian susunan '*DATA*' akan dijelaskan pada pembahasan selanjutnya). Sama dengan sebelumnya, keduanya dipisahkan dengan tanda pagar ('#'). Tanda pagar di bagian awal digunakan sebagai penanda untuk menunjukkan bahwa sms yang dikirimkan dari Android *gateway* dapat dikenali oleh sms *receiver*. Gambar 3.19 merupakan ilustrasi pengiriman data sensor melalui sms.

Gambar 3.19 Ilustrasi Pengiriman Data Sensor (SMS)

3.4.1.3 Diagram Alir Subproses *StrictTask*

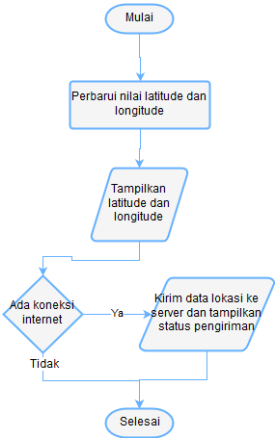


Gambar 3.20 Diagram Alir Subproses *StrictTask*

Gambar 3.20 merupakan diagram alir dari proses *StrictTask* yang merupakan proses yang dijalankan jika memilih menu *strict mode* (mode ketat). Hampir sama dengan *NormalTask*, yang akan dijalankan untuk melakukan pengecekan koneksi internet setiap sepuluh menit sebelum mengirimkan data. Hanya saja yang membedakan *strict mode* dengan *normal mode* ada pada pengiriman smsnya. Jika tidak ada koneksi internet, maka data akan langsung dikirimkan melalui sms tanpa menunggu hingga 30 menit.

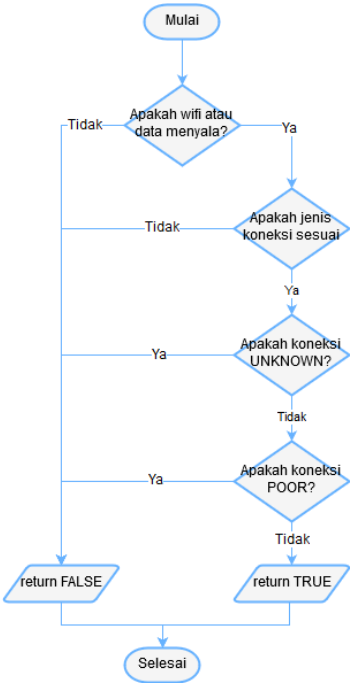
3.4.1.4 Diagram Alir Subproses *OnLocationChanged*

Pada subproses *OnLocationChange* dilakukan mekanisme untuk memperbarui lokasi dari kelompok sensor (Android *gateway*). Saat ada perubahan lokasi dari maka nilai dari *latitude* dan *longitude* akan diperbarui juga. Setelah mendapatkan nilai *latitude* dan *longitude* terbaru, dilakukan pengecekan koneksi internet. Jika terdapat koneksi internet, maka akan dilakukan pengiriman data lokasi terbaru ke server melalui internet. Gambar 3.21 menunjukkan diagram alir subproses *OnLocationChanged.*



Gambar 3.21 Diagram Alir Subproses *OnLocationChanged*

3.4.1.5 Diagram Alir Subproses *InternetAvailability*



Gambar 3.22 Diagram Alir Subproses *InternetAvailability*

Subproses yang dijabarkan pada Gambar 3.22, digunakan untuk menentukan ketersediaan koneksi internet. Penentuan ketersediaan koneksi internet adalah penggabungan dari tiga aspek yaitu salah satu dari *wifi* atau paket data aktif atau tidak, jenis koneksi dan kecepatan koneksi internet. Jenis koneksi internet dijabarkan pada Tabel 3.5.

Tabel 3.5 Jenis Koneksi Internet

No	Jenis Koneksi	Status
1.	WIFI	*TRUE*
2.	1xRTT	*FALSE*
3.	CDMA	*FALSE*

No	Jenis Koneksi	Status
4.	EDGE	*FALSE*
5.	EVDO 0	*TRUE*
6.	EVDO A	*TRUE*
7.	GPRS	*FALSE*
8.	HSDPA	*TRUE*
9.	HSPA	*TRUE*
10.	HSUPA	*TRUE*
11.	UMTS	*TRUE*
12.	EHRPD	*TRUE*
13.	EVDO B	*TRUE*
14.	HSPAP	*TRUE*
15.	IDEN	*FALSE*
16.	LTE	*TRUE*

Sedangkan kecepatan internet dibagi menjadi 5. Kecepatan koneksi internet dijabarkan pada Tabel 3.6.

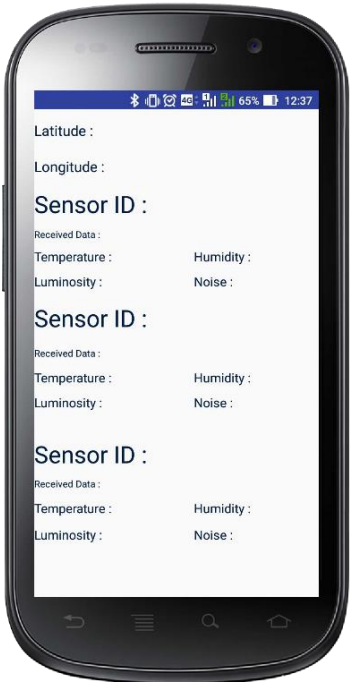
Tabel 3.6 Kecepatan Koneksi Internet

No	Keadaan Koneksi	Kecepatan	Status
1.	UNKNOWN	Kurang dari 0 kbps	FALSE
2.	POOR	1 – 150 kbps	FALSE
3.	MODERATE	151 – 550 kbps	TRUE
4.	GOOD	551 – 2000 kbps	TRUE
5.	EXCELLENT	Lebih dari 2000 kbps	TRUE

3.4.2 Perancangan Antarmuka Pengguna (Android *Gateway*)

Tidak terlalu banyak hal yang dibahas dalam bagian ini. Pada bagian ini hanya akan dijelaskan mengenai rancangan antarmuka pengguna dari aplikasi Android *gateway*. Data – data penting yang ditampilkan pada antarmuka aplikasi adalah lokasi (ditunjukkan dengan *Latitude*, *Longitude*), *Sensor ID, Received Data* yaitu data sensor yang diperoleh dari data yang dikirimkan *node* sensor, *Temperature, Humidity, Luminousity* (sensor cahaya) dan *Noise* (sensor suara). Nilai dari setiap data akan senantiasa

berubah seiring dengan data yang diterima dari *node* sensor. Gambar 3.23 merupakan rancangan antarmuka pengguna.



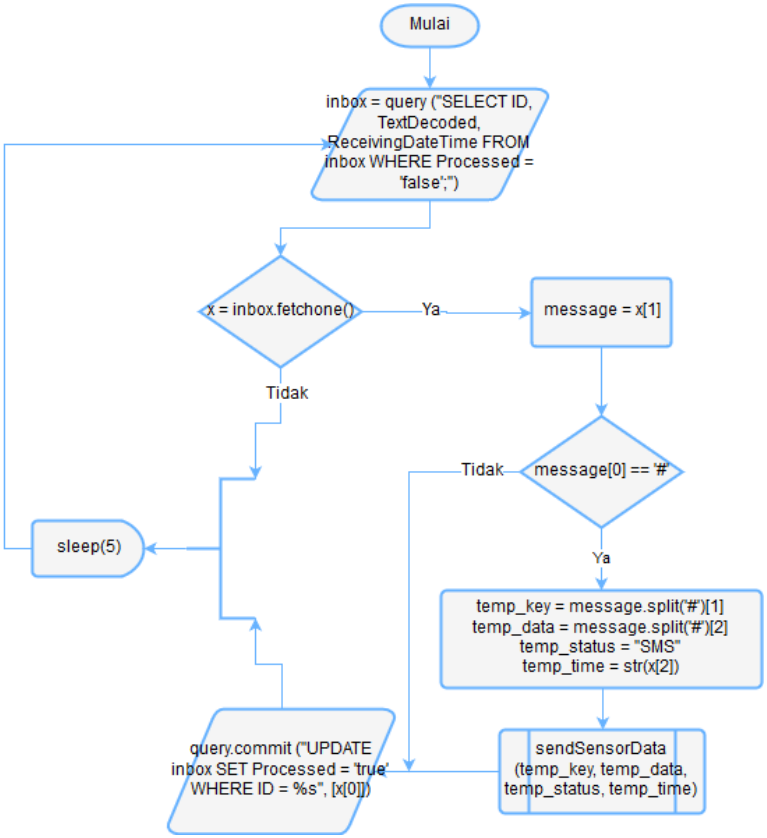
Gambar 3.23 Rancangan Antarmuka Pengguna (Android *Gateway*)

3.5 Perancangan SMS *Receiver*

Pada bagian ini akan dilakukan perancangan dari *node* sms *receiver*. *Node* ini akan dibuat dengan bantuan sebuah alat bantu sms *gateway* yaitu Gammu.

Node sms *receiver* bertugas untuk menerima sms dari Android *gateway* untuk kemudian diteruskan ke server. *Node* ini dimungkinkan berupa sebuah modem yang dapat langsung dipasang pada komputer server atau bisa pula dipasang pada komputer yang berlainan.

3.5.1 Perancangan Diagram Alir (SMS *Receiver*)



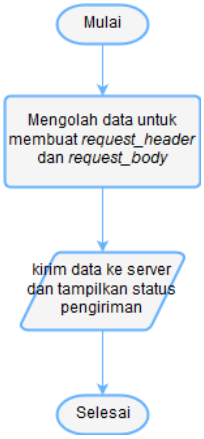
Gambar 3.24 Diagram Alir Sistem (SMS *Receiver*)

Pada bagian ini akan dijelaskan mengenai mekanisme kerja dari *node* sms *receiver*. *Node* ini berperan untuk mengolah sms yang masuk. Sms yang masuk tersebut jika sesuai dengan susunan penulisan, akan langsung dikirimkan ke server. Gambar 3.24 merupakan gambaran umum jalannya proses sms *receiver* secara keseluruhan dalam bentuk diagram alir

Hal pertama yang dilakukan adalah menjalankan sebuah *query* untuk mendapatkan data yang ada pada basis data sms *receiver*. Tiga data yang diambil adalah *ID* (*primary key* dari sms), *TextDecoded* (isi pesan pendek) dan *ReceivingDateTime* (waktu masuknya sms). Semua data yang berhasil diperoleh akan diolah satu per satu dan dimasukkan ke dalam variabel *x*. Jika tidak ada satu pun data yang masuk, maka proses akan dihentikan selama 5 detik sebelum kemudian melakukan pengecekan kembali.

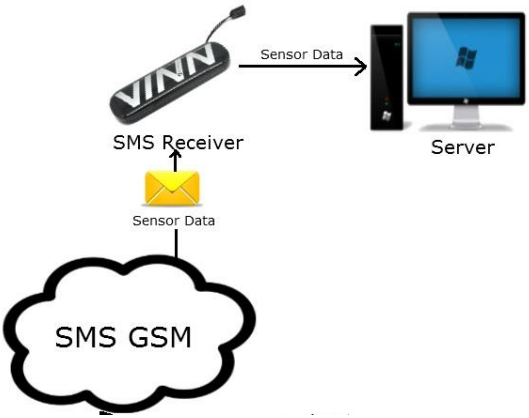
Setelah didapatkan adanya sms, maka diambil nilai dari *x[1]* yang merupakan isi dari sms dan dimasukkan dalam variabel *message*. Kemudian dilakukan pengecekan karakter pertama yang ada pada isi pesan, jika karakter pertama tidak sama dengan karakter pagar (‘#’) maka status *Processed* data akan langsung dijadikan *true*.

Jika diperoleh susunan yang sesuai, maka pesan akan dipecah sesuai dengan susunan ‘*#KEY#DATA*’. ‘*KEY*’ di sini merupakan nilai dari *sensor_key* dan ‘*DATA*’ merupakan nilai dari data yang didapatkan pada *node* sensor. Setelah itu berjalanlah subproses *sendSensorData*. Gambar 3.25 merupakan diagram alir subproses *sendSensorData*.



Gambar 3.25 Diagram Alir Subproses *sendSensorData*

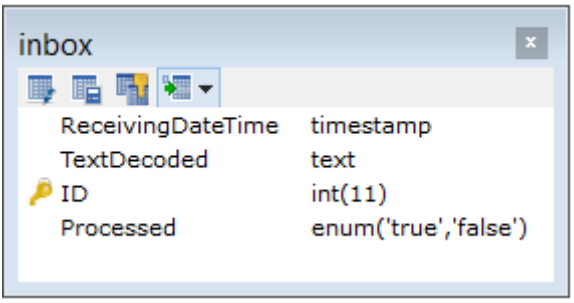
Pada bagian subproses *sendSensorData*, data yang telah diperoleh sebelumnya akan diolah dan dikirimkan ke server. Pengiriman data dilakukan dengan pustaka *urllib2*. Sebelum dikirimkan, data di-*encode* dalam mode *utf-8*. Proses pengiriman dilakukan dengan membuat *request_header* dan *request_body*. Tipe konten (*Content-Type*) yang digunakan adalah *'application/x-www-form-urlencoded*'. Data tersebut ditampilkan pada layar untuk mempermudah proses *debug*. Gambar 3.26 merupakan ilustrasi sms *receiver* bekerja.



Gambar 3.26 Ilustrasi SMS *Receiver* Bekerja

3.5.2 Perancangan Basis Data (SMS *Receiver*)

Pada bagian ini akan dibahas mengenai perancangan basis data pada *node* sms *receiver*. Basis data yang digunakan di sini adalah basis data yang ada pada Gammu. Basis data yang ada pada Gammu tersebut memiliki 9 entitas. Namun tidak semuanya akan digunakan untuk *node* sms *receiver* ini. Hanya satu tabel saja yang akan digunakan yaitu tabel *inbox*.



Gambar 3.27 Skema Tabel *inbox*

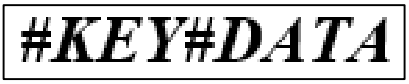
Tabel *inbox* ini sejatinya memiliki lebih dari 10 atribut. Semuanya ini memiliki fungsi masing – masing dalam sistem sms *gateway*. Namun, di dalam *node* sms *receiver* ini hanya akan digunakan 4 atribut sesuai kebutuhan. Gambar 3.27 merupakan skema tabel *inbox* beserta atribut – atribut yang diperlukan. Keempat atribut tersebut meliputi *ReceivingDateTime*, *TextDecoded*, *ID* dan *Processed*. Tabel 3.7 merupakan detail tabel *inbox*.

Tabel 3.7 Detail Tabel *inbox*

No	Nama Atribut	Tipe Data	Keterangan
1.	*ReceivingDateTime*	*timestamp*	Atribut ini menunjukkan waktu masuknya sebuah sms
2.	*TextDecoded*	*text*	Merupakan isi pesan dari sms yang diterima
3.	*ID*	*integer*	Merupakan *primary key* dari tabel *users*. Diatur sebagai *auto increment*
4.	*Processed*	*enum('true', 'false')*	Merupakan penanda (*flag*) untuk menandakan apakah sudah pernah dibaca atau belum

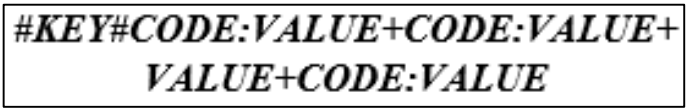
3.5.3 Perancangan Susunan (*Format*) SMS

Pada bagian ini akan dilakukan perancangan susunan atau *format* dari sms yang akan dijadikan patokan. Berikut ini *format* umum penulisan sms :



Gambar 3.28 *Format* Umum Penulisan SMS

Berdasarkan Gambar 3.28, *format* penulisan sms diawali dengan sebuah tanda pagar (‘#’) sebagai penanda. Selanjutnya ada ‘*KEY*’ yang merupakan representasi dari *sensor_key*. Agar dapat mengetahui *key* dari suatu sensor, dapat dilihat pada detail sensor pada website. Kemudian dipisahkan lagi dengan tanda pagar sekali lagi. Baru setelah itu ada ‘*DATA*’ yang memiliki rincian *format* penulisan tersendiri. Berikut ini *format* rinci penulisan sms :



Gambar 3.29 *Format* Rinci Penulisan SMS

Berdasarkan *format* penulisan rinci sms pada Gambar 3.29, ‘*DATA*’ itu sendiri dijabarkan lagi menjadi dua yaitu ‘*CODE*’ dan ‘*VALUE*’. ‘*CODE*’ merupakan suatu kode tersendiri yang akan dijelaskan pada Tabel 3.8. Sedangkan ‘*VALUE*’ merupakan nilai dari sebuah pembacaan sensor. ‘*CODE*’ dan ‘*VALUE*’ dipisahkan dengan tanda titik dua (‘:’). Setiap gabungan dari ‘*CODE*’ dan ‘*VALUE*’ dipisahkan dengan tanda tambah (‘+’) untuk mengirimkan lebih dari satu nilai pada jenis sensor yang berbeda. Berikut ini Tabel 3.8 yang menjelaskan mengenai setiap ‘*CODE*’ dan maknanya.

Tabel 3.8 Penjelasan Kode Sensor

No	CODE	Keterangan
1.	TEM	Merupakan kode yang menunjukkan bahwa nilai yang menyertai kode ini adalah nilai yang didapat dari sensor suhu (*temperature*)
2.	HUM	Merupakan kode yang menunjukkan bahwa nilai yang menyertai kode ini adalah nilai yang didapat dari sensor kelembaban udara (*humidity*)
3.	LIG	Merupakan kode yang menunjukkan bahwa nilai yang menyertai kode ini adalah nilai yang didapat dari sensor cahaya (*light intensity / luminosity*)
4.	SOU	Merupakan kode yang menunjukkan bahwa nilai yang menyertai kode ini adalah nilai yang didapat dari sensor suara (*sound intensity / noise*)
5.	X	Merupakan kode yang menunjukkan bahwa nilai yang menyertai kode ini adalah nilai X yang didapat dari sensor akselerometer (*accelerometer*)
6.	Y	Merupakan kode yang menunjukkan bahwa nilai yang menyertai kode ini adalah nilai Y yang didapat dari sensor akselerometer (*accelerometer*)
7.	Z	Merupakan kode yang menunjukkan bahwa nilai yang menyertai kode ini adalah nilai Z yang didapat dari sensor akselerometer (*accelerometer*)

Setelah melihat Tabel 3.8 yang menjelaskan rincian kode sensor untuk *format* sms, dapat diketahui bahwa pengiriman nilai sensor dalam satu waktu bisa lebih dari satu. Namun yang harus diperhatikan adalah, dalam satu pengiriman tidak boleh ada lebih dari satu jenis sensor yang sama. Berikut ini contoh susunan penulisan sms yang **kurang sesuai** :

#16c571188108b0a2c0c133bb7a7a6d81c2961e6e#TEM:25.64+HUMID:45.25+TEM:31.53SOU:32.67.

Ada beberapa kesalahan yang ada pada contoh pengiriman sms di atas. Pertama kode '*HUMID*' seharusnya '*HUM*' saja. Kedua terdapat dua data dengan kode yang sama yaitu '*TEM*'. Dan yang terakhir setelah angka '*31.53*' seharusnya dipisahkan dengan sebuah tanda '+'. Berikut ini contoh penulisan sms yang **sesuai** : *#16c571188108b0a2c0c133bb7a7a6d81c2961e6e#TEM:25.64+HUM:45.25+LIG:31.53+SOU:32.67*.

3.6 Perancangan *Node* Sensor

Pada bagian ini, akan dijelaskan mengenai rancangan dari *node* sensor. *Node* sensor ini akan dibangun menggunakan mikrokontroler Arduino UNO R3. Sensor yang digunakan adalah sensor suhu LM35, kelembaban udara DHT11, cahaya LM393 dan suara KY-038. Sedangkan untuk modul komunikasi yang digunakan adalah modul HC-06 yang merupakan modul komunikasi nirkabel yaitu Bluetooth.

Pembahasan lain yang akan dijelaskan dalam bagian ini adalah mengenai mekanisme kerja dari sensor. Mekanisme kerja akan digambarkan dalam bentuk diagram alir.

3.6.1 Perancangan Rangakaian Utama

Rangkaian dari *node* sensor membutuhkan beberapa komponen yang harus dipersiapkan. Berikut ini Tabel 3.9 menjabarkan komponen yang dibutuhkan beserta penjelasannya :

Tabel 3.9 Komponen Rangkaian

No	Nama Komponen	Deskripsi	Jumlah
1.	*Arduino UNO R3*	Merupakan sebuah mikrokontroler yang akan menjadi otak sekaligus mengatur berjalannya komponen lain. Tipe Arduino	1 buah

No	Nama Komponen	Deskripsi	Jumlah
		yang digunakan ini adalah tipe *UNO* dan *R3* adalah kependekan dari *Revision 3*	
2.	*HC-06*	Merupakan modul komunikasi nirkabel menggunakan Bluetooth. Modul ini bertindak sebagai *slave*.	1 buah
3.	*LM35, DHT11, LM393* dan *KY-038*	Modul ini adalah modul - modul sensor yang secara berurutan adalah sensor suhu, kelembaban, cahaya dan suara	1 buah
4.	*Resistor 20K Ω*	Digunakan untuk membuat rangkaian pembagi tegangan	1 buah
5.	*Resistor 10K Ω*	Digunakan untuk membuat rangkaian pembagi tegangan	1 buah
6.	*Breadboard*	Sebuah papan yang digunakan untuk membuat *prototype* rangkaian elektronik	1 buah
7.	*Kabel Jumper*	Menghubungkan antara satu komponen dengan komponen lainnya	10 – 12 buah

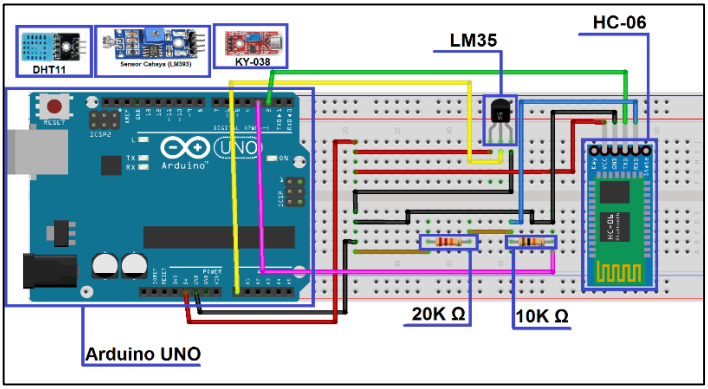
Semua komponen yang ada, kemudian dirangkai menjadi satu. Gambar 3.30 merupakan gambaran rangkaian yang akan dibuat. Pada gambar tersebut, setiap komponen mulai dari mikrokontroler Arduino UNO, HC-06, LM35, DHT11, LM393, KY-038 dan resistor dirangkai menjadi satu menggunakan kabel *jumper* dalam satu *breadboard*. Penjelasan mengenai *PIN* yang terhubung dengan Arduino akan dijelaskan pada Tabel 3.10.

Tabel 3.10 Koneksi *PIN* Arduino dengan Komponen Lain

PIN	Modul / Komponen	Deskripsi
POWER		
5V	*Breadboard (VCC)*	Sebagai *VCC*
GND	*Breadboard (GND)*	Sebagai *GROUND*
ANALOG IN		

PIN	Modul / Komponen	Deskripsi
A0	*LM35*	Terhubung dengan *PIN* kedua (tengah) dari sensor suhu LM35.
A3	*LM393*	Terhubung dengan *PIN* pada sensor cahaya (LM393)
A5	*KY-038*	Terhubung dengan *PIN* pada sensor suara KY-038
DIGITAL (PWM~)		
D2	*HC-06*	Berhubungan dengan *PIN* TX pada modul HC-06
D3	*Breadboard (HC-06)*	Berhubungan dengan *breadboard* untuk kemudian diteruskan agar tersambung dengan *PIN* RX pada modul HC-06
D7	*DHT11*	Terhubung dengan sensor DHT11

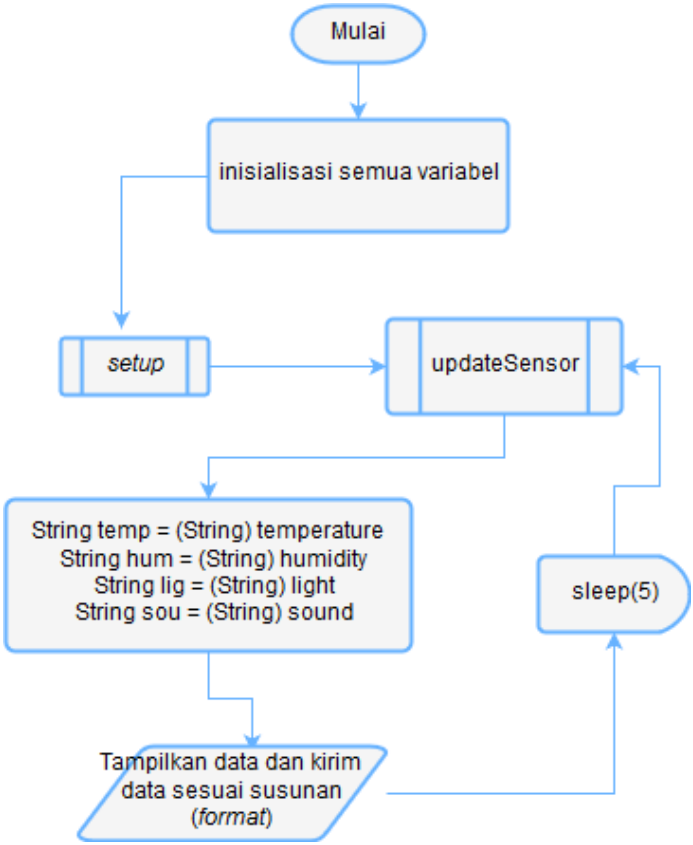
Rancangan rangakaian utama *node* sensor pada Gambar 3.30 adalah gambaran dari rangkaian yang akan diimplementasikan selanjutnya. Sensor LM35 akan digunakan untuk mendeteksi suhu udara, yang kemudian akan dikirimkan ke Android *gateway* melalui modul Bluetooth HC-06. Resistor 20K Ω dan 10K Ω digunakan sebagai pembagi tegangan agar modul HC-06 tidak mudah rusak saat digunakan.



Gambar 3.30 Rancangan Rangkaian *Node* Sensor

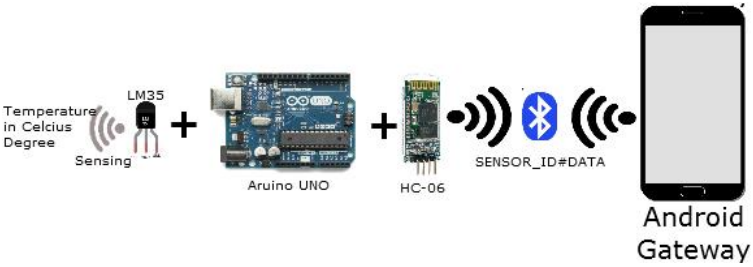
3.6.2 Perancangan Diagram Alir (*Node* Sensor)

Pada subbab ini, akan dibahas alur dari sistem yang akan berjalan pada *node* sensor. Perancangan aliran data sistem pada *node* sensor secara keseluruhan dilakukan agar dapat lebih mudah memahami jalannya *node* sensor secara menyeluruh. Selain itu digunakan juga untuk menunjukkan fungsi – fungsi utama atau aliran logika jalannya *node* sensor. Gambar 3.31 sebuah diagram alir jalannya proses.



Gambar 3.31 Diagram Alir Sistem (Node Sensor)

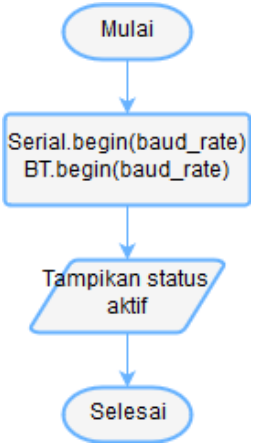
Secara garis besar, sistem yang berjalan dalam *node* sensor terdiri dari beberapa subproses penting yaitu *setup* dan *updateSensor*. Subproses *setup* digunakan untuk menginisialisasi berjalannya mikrokontroler. Sedangkan subrposes *updateSensor* digunakan untuk mendapatkan nilai sensor terbaru dari lingkungan. Subproses *updateSensor* sebenarnya ada di dalam fungsi *loop* yang menjadi aturan bawaan pemrograman Arduino.



Gambar 3.32 Ilustrasi Cara Kerja *Node* Sensor

Gambar 3.32 merupakan ilustrasi cara kerja node sensor secara umum.

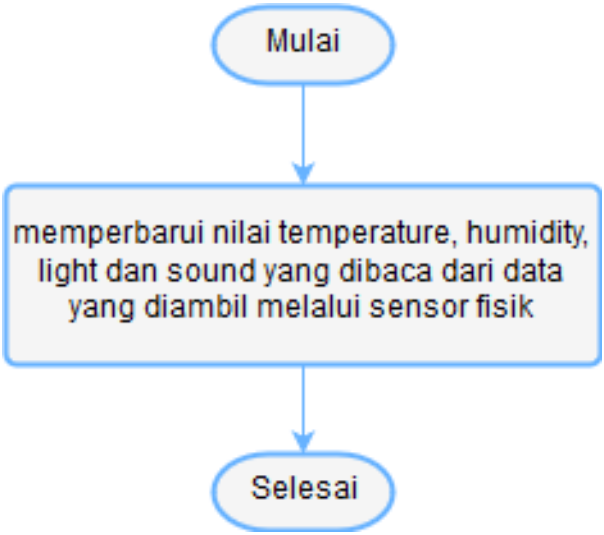
3.6.2.1 Diagram Alir Subproses *setup*



Gambar 3.33 Diagram Alir Subproses *setup*

Subproses *setup* memiliki peran sebagai insiator yang memulai jalannya *node* sensor. Pada subproses ini dijalankan sirkuit *Serial* dan Bluetooth. Perintah yang digunakan adalah *Serial.begin(baud_rate)* dan *BT.begin(baud_rate*). Nilai dari *baud_rate* adalah frekuensi yang dipakai untuk menjalankan sirkuit. Nilai - nilai yang biasa digunakan dalam *baud_rate* adalah 2400, 3600, 9600 dan lainnya. Kali ini nilai *baud_rate* yang akan digunakan adalah 9600. Gambar 3.33 merupakan diagram alir subproses *setup*.

3.6.2.2 Diagram Alir Subproses *updateSensor*



Gambar 3.34 Diagram Alir Subproses *updateSensor*

Gambar 3.34 merupakan diagram alir dari subproses *updateSensor*. Proses ini dijalankan untuk memperbarui nilai dari variabel *global* sensor – sensor yang ada. Bertujuan untuk mendapatkan nilai sensor terbaru pada lingkungan sekitar. Gambar 3.35 merupakan ilustrasi pengambilan data dengan sensor.

Gambar 3.35 Ilustrasi Pengambilan Data dengan Sensor Suhu

BAB IV
IMPLEMENTASI

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

4.1 Lingkungan Implementasi

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

4.1.1 Lingkungan Implementasi Perangkat Keras

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

Tabel 4.1 Lingkungan Implementasi Perangkat Keras

Perangkat	Detail Perangkat
Perangkat Komputer	**Model :** • ASUS ROG GL552JX **Manufaktur :** • ASUSTek Computer Inc. ***Processor* :** • Intel(R) Core(TM) i7-4720HQ @2.6 GHz 2.59 GHz

Perangkat	Detail Perangkat
	Memori : • 8 GB **Sistem Operasi :** • Microsoft Windows 8.1 Embedded 64-bit
Perangkat Mikrokontroler	**Mikrokontroler :** • Atmega328 **Model :** • Arduino UNO *Revision* 3 **Tegangan :** • 5 V **Memori *Flash* :** • 32 KB **SRAM :** • 2 KB **Sensor :** • Suhu, Kelembaban, Cahaya dan Suara ***Transceiver* :** • Model HC-06 (Bluetooth)
Perangkat Modem (SMS *Receiver*)	**Model :** • Turkcell 3G Vinn Modem – MF667 **Manufaktur :** • ZTE Corporation **Koneksi :** • 3G HSPA + 21.6 Mbps **Dimensi :** • 76.5 x 27.2 10.5 mm
Perangkat Mobile (*Smartphone*)	**Model :** • ASUS Zenfone Max ZC550KL **Manufaktur :** • ASUSTeK Computer Inc.

Perangkat	Detail Perangkat
	Processor : • Qualcomm Snapdragon 400/410 @1.21 GHz **Memori :** • 2 GB **Dimensi :** • 156 x 77.5 x 10.6 mm **Lebar Layar :** • 5.5” **Resolusi :** • 720 x 1280 pixels **Bluetooth :** • 4.0 + EDR + A2DP **Baterai :** • 5000 mAh **Sistem Operasi :** • Android 6.0.1 (Marsmallow)

4.1.2 Lingkungan Implementasi Perangkat Lunak

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

Tabel 4.2 Lingkungan Implementasi Perangkat Lunak

Perangkat	Detail Perangkat
Perangkat Komputer	**Sistem Operasi :** • Microsoft Windows 8.1 Embedded 64-bit

Perangkat	Detail Perangkat
	***Software* Arduino :** • Arduino IDE 1.6.11 ***Software* SMS *Gateway* :** • Gammu for Windows 1.31.0 • Python 2.7.10 **Webserver :** • XAMPP 1.8.2 • Apache 2.4.17 • PHP 5.6.14 **Basis Data :** • MySQL 5.0.11 **Web *Framework* :** • Bootstrap 3.3.6 (*Front-End*) • CodeIgniter 3.0.6 (*Back-End*) ***Text Editor* :** • Sublime Text 3 Build 3103 ***Software* Android :** • Android Studio 2.2

4.2 Implementasi *Sensor Cluster Management*

Pada ini akan dibahas mengenai bagaimana implementasi yang dilakukan untuk membangun *Sensor Cluster Management* yang mengacu pada bab 3.3. Sistem berupa website ini dibangun dengan menggunakan *framework* Bootstrap dan CodeIgniter. Sehingga kebanyakan *syntax* yang akan dijabarkan dalam sebuah *pseudocode* akan berhubungan erat dengan *syntax* pada *framework* tersebut khususnya bahasa pemrogramam PHP. Basis data yang digunakan adalah MySQL.

4.2.1 Implementasi Kasus Kegunaan

Berdasarkan kasus kegunaan yang dirancang pada bab 3.3.1, ada enam kasus kegunaan yang akan diimplementasikan. Kasus

kegunaan tersebut meliputi kasus kegunaan menampilkan sensor berdasarkan kelompok, mendaftarkan kelompok baru, mendaftarkan sensor baru, menampilkan peta lokasi keseluruhan, menampilkan peta lokasi khusus dan menampilkan detail sensor. Berikut ini penjelasan secara lebih menyeluruh.

4.2.1.1 Menampilkan Sensor Berdasarkan Kelompok

Kasus kegunaan ini akan menyajikan halaman utama Pengguna dan merupakan implementasi dari bab 3.3.1.1. Fungsi pada *controller* yang ada adalah fungsi *index*.

Pada kasus kegunaan ini yang akan dilakukan adalah mengolah seluruh data yang berhubungan dengan *sensor_group* dan juga *sensor* yang saling berhubungan dengannya. Lalu menampilkan data – data tersebut pada halaman utama (*home*).

Fungsi *index* pada Kode Sumber 4.1 berperan dalam mengelola data – data yang akan ditampilkan pada halaman utama. Hal pertama yang dilakukan adalah *load model* dan melakukan inisialisasi data. Lalu memanggil fungsi *getUserData* untuk mendapatkan data *users*. Daftar kelompok didapatkan dengan fungsi *getGroupList* dan memasukkan parameter *user_id*. Sedangkan untuk mendapatkan nilai sensor yang terakhir masuk dilakukan dengan pemanggilan fungsi *getLastSensorList* dan parameter yang sama.

```
1  FUNCTION index()
2    load model
3    initialize data
4    set data->userdata ← getUserData()
5    set data->grouplist ←
.      model->getGroupList(data->userdata)
6    set data->sensorlist ←
.      model->getLastSensorList(data->userdata-
.      >id)
7    call load->view->home(data)
8  ENDFUNCTION
```

Kode Sumber 4.1 *Pseudocode* Fungsi *index*

Setelah penjelasan mengenai fungsi *index*, sekarang akan dibahas mengenai fungsi untuk mengakses basis data. Fungsi pertama adalah *getGroupList*. Fungsi ini berguna untuk mendapatkan daftar kelompok sensor sesuai dengan parameter yang dikirim yaitu *user_id*. Berikut ini Kode Sumber 4.2.

```
1  FUNCTION getGroupList(user_id)
2    load model
3    initialize data
4    set data ← user_id
5    set db->from ← 'sensor_group'
6    set db->where ← "'sensor_group.user_id',
.      user_is"
7    set query ← db->get
8    return query->result
9  ENDFUNCTION
```

Kode Sumber 4.2 *Pseudocode* Fungsi *getGroupList*

Fungsi pertama adalah *getLastSensorList*. Fungsi ini berguna untuk mendapatkan nilai data sensor yang terakhir masuk dari semua sensor sesuai dengan parameter yang dikirim yaitu *user_id*. Berikut ini Kode Sumber 4.3.

```
1  FUNCTION getLastSensorList(user_id)
2    load model
3    initialize data
4    set data ← user_id
5    set db->query ← get 'sensor' and last
.      'sensor_data' where 'sensor.user_id' =
.      user_id
6    set query ← db->query
7    return query->result
8  ENDFUNCTION
```

Kode Sumber 4.3 *Pseudocode* Fungsi *getLastSensorList*

4.2.1.2 Mendaftarkan Kelompok Baru

Kasus kegunaan ini adalah implementasi dari bab 3.3.1.2. Pada bagian kasus ini, Pengguna dapat mendaftarkan kelompok sensor baru yaitu melalui fungsi *createGroup*.

Pada Kode Sumber 4.4 diperlihatkan mengenai jalannya proses pada sebuah *pseudocode*. Fungsi ini memiliki sebuah parameter yang mana jika nilainya sama dengan '*form_group'* maka akan membuka sebuah halaman isian data kelompok sensor. Namun jika nilainya sama dengan *'submit_group'* itu artinya isian yang ada sudah berhasil diisi dan akan diolah untuk dimasukkan basis data.

```
1  FUNCTION createGroup(param = NULL)
2    load model
3    initialize data
4    initialize newGroupData
5    set data->userdata ← getUserData()
6    IF param != NULL THEN
7      IF param = 'form_group' THEN
8        set load->view->new_sensor_group(data)
9      ELSE IF param = 'submit_group' THEN
10       set user_id ← getUserData()->id
11       set group_name ← input->post('name')
12       set description ←
.          input->post('description')
13       set newGroupData ← array_push(user_id,
.          group_name, description)
14       call model->createNewGroup(newGroupData)
15       load redidect('home')
16     ENDIF
17   ENDIF
18 ENDFUNCTION
```

Kode Sumber 4.4 *Pseudocode* Fungsi *createGroup*

Setelah itu adalah sebuah fungsi *createNewGroup* yaitu untuk memasukkan data baru ke basis data. Fungsi ini dipanggil dalam fungsi *createGroup*. Berikut ini secara Kode Sumber 4.5 lebih jelas.

```
1  FUNCTION createNewGroup(newGroupData)
2    load model
3    select database
4    insert newGroupData ← database
5  ENDFUNCTION
```

Kode Sumber 4.5 *Pseudocode* Fungsi *createNewGroup*

4.2.1.3 Mendaftarkan Sensor Baru

Kasus kegunaan ini adalah implementasi dari bab 3.3.1.3. Pada bagian kasus ini, Pengguna dapat mendaftarkan sensor baru yaitu melalui fungsi *createSensor*.

Pada Kode Sumber 4.6 diperlihatkan mengenai jalannya proses pada sebuah *pseudocode*. Fungsi ini memiliki sebuah parameter yang mana jika nilainya sama dengan '*form'* maka akan membuka sebuah halaman isian data sensor. Namun jika nilainya sama dengan *'submit'* itu artinya isian yang ada sudah berhasil diisi dan akan diolah untuk dimasukkan basis data. Sebagai tambahan, nilai dari *sensor_key* di peroleh dengan cara memasukkan nilai dari pertambahan antara nama sensor, waktu dan *user_id* ke dalam fungsi *hash* SHA1.

```
1   FUNCTION createSensor(param = NULL)
2     load model
3     initialize data
4     initialize newSensorData
5     set data->userdata ← getUserData()
6     set data->grouplist ←
.       model->getGroupList(data->userdata)
7
8     IF param != NULL THEN
9
10      IF param = 'form' THEN
11        set load->view->new_sensor_form(data)
12      ELSE IF param = 'submit' THEN
13        set user_id ← getUserData()->id;
14        set sensor_name ← input->post('name')
15        set sensor_description ←
.           input->post('description')
16        set group_id ← input->post('group_id')
17        set sensor_key ← sha1(sensor_name +
.           time() + user_id)
18        set temperature_sensor ←
.           input->post('temperature_sensor')
19        set humidity_sensor ←
.           input->post('humidity_sensor')
```

```
20        set light_sensor ←
.           input->post('light_sensor')
21        set sound_sensor ←
.           input->post('sound_sensor')
23        set accelerometer_sensor ←
.           input->post('accelerometer_sensor')
24        set newSensorData ← array_push(user_id,
.           sensor_name, sensor_description,
.           group_id, sensor_key,
.           temperature_sensor, humidity_sensor,
.           light_sensor, sound_sensor,
.           acceleromter_sensor)
25        call model->createNewSensor(
.           newSensorData)
26        load redidect('home')
27      ENDIF
28    ENDIF
29  ENDFUNCTION
```

Kode Sumber 4.6 *Pseudocode* Fungsi *createSensor*

Setelah itu adalah sebuah fungsi *createNewSensor* yaitu untuk memasukkan data baru ke dalam basis data. Fungsi ini dipanggil dalam fungsi *createSensor*. Berikut ini secara lebih jelas Kode Sumber 4.7.

```
1  FUNCTION createNewSensor(newSensorData)
2    load model
3    select database
4    insert newSensorData ← database
5  ENDFUNCTION
```

Kode Sumber 4.7 *Pseudocode* Fungsi *createNewSensor*

4.2.1.4 Menampilkan Peta Lokasi Keseluruhan

Bagian ini merupakan implementasi dari bab 3.3.1.4 yaitu untuk menampilkan lokasi dari keseluruhan kelompok sensor yang telah didaftarkan *users* tertentu. Fungsi yang diakses untuk menampilkan peta lokasi adalah fungsi *viewMaps*. Hampir sama dengan fungsi *index*, di dalam fungsi *viewMaps* ini akan mengakses fungsi yang sama dengan fungsi *index*.

Fungsi untuk mendapatkan data yang dimaksud adalah fungsi *getGroupList* dan fungsi *getLastSensorList*. Kedua fungsi ini dapat dilihat lebih jelas pada Kode Sumber 4.2 dan Kode Sumber 4.3. Berikut ini fungsi *viewMaps* lebih jelasnya pada Kode Sumber 4.8.

```
1  FUNCTION viewMaps()
2    load model
3    initialize data
4    set data->userdata ← getUserData()
5    set data->grouplist ← model->
.      getGroupList(data->userdata)
6    set data->sensorlist ← model->
.      getLastSensorList(data->userdata->id)
7    call load->view->maps(data)
8  ENDFUNCTION
```

Kode Sumber 4.8 *Pseudocode* Fungsi *viewMaps*

4.2.1.5 Menampilkan Peta Lokasi Khusus

Sedikit berbeda dengan kasus kegunaan sebelumnya, pada kasus kegunaan yang satu ini akan ditampilkan secara spesifik lokasi dari sebuah kelompok sensor pada peta (implementasi bab 3.3.1.5). Peta lokasi khusus ini dengan mengakses fungsi *gatewayLocation*. Di bawah ini merupakan Kode Sumber 4.9 yang merupakan *pseudocode* dari fungsi *gatewayLocation*.

```
1  FUNCTION gatewayLocation(group_id = NULL)
2    load model
3    initialize data
4    IF group_id != NULL THEN
5      set data->userdata ← getUserData()
6      set data->groupdata ←
.        model->getGroupData(group_id)
7      set data->sensordata ←
.        model->getLastSensorData(group_id)
8      call load->view->gateway_location(data)
9    ENDIF
10 ENDFUNCTION
```

Kode Sumber 4.9 *Pseudocode* Fungsi *gatewayLocation*

Fungsi *getGroupData* dipanggil pada fungsi di atas, untuk mengakses basis dara dan mendapatkan data *sensor_group*. Fungsi ini memiliki parameter yaitu *group_id*, yang berarti data *sensor_group* yang diambil berdasarkan pada *group_id*. Kode Sumber 4.10 merupakan *pseudocode* dari fungsi *getGroupData* untuk mengambil data *sensor_group* berdasarkan *group_id*.

```
1  FUNCTION getGroupData(group_id)
2    load model
3    initialize data
4    set data ← group_id
5    set db->query ← 'SELECT * FROM sensor_group
.      WHERE id = '.data.';'
6    set query ← db->query
7    return query->result
8  ENDFUNCTION
```

Kode Sumber 4.10 *Pseudocode* Fungsi *getGroupData*

Fungsi lain yang dipanggil dalam Kode Sumber 4.11 adalah fungsi *getLastSensorData*. Fungsi ini berguna untuk mendapatkan nilai terakhir dari data sensor berdasarkan *group_id*. Artinya setiap sensor yang ada pada kelompok sensor tertentu akan didapatkan data terakhir yang berhasil direkamnya. Berikut ini Kode Sumber 4.11 yang lebih jelas.

```
1  FUNCTION getLastSensorData(group_id)
2    load model
3    initialize data
4    set data ← group_id
5    set db->query ← get 'sensor' and last
.      'sensor_data' where 'sensor.group_id' =
.      group_id
6    set query ← db->query
7    return query->result
8  ENDFUNCTION
```

Kode Sumber 4.11 *Pseudocode* Fungsi *getLastSensorData*

4.2.1.6 Menampilkan Detail Sensor

Kasus kegunaan yang terakhir adalah implementasi dari bab 3.3.1.6. Pada halaman detail sensor ini akan ditampilkan mengenai informasi sensor, data *realtime* sensor dalam bentuk grafik dan riwayat data yang masuk.

Agar dapat menampilkan detail sensor, fungsi yang harus diakses adalah fungsi *viewSensor*. Fungsi ini akan mengelola semua data yang diperlukan untuk kemudian ditampilkan pada halaman *view_sensor*.

Fungsi *viewSensor* ini akan mengambil data berdasarkan pada *sensor_id*. Berikut ini Kode Sumber 4.12 untuk dapat memahami lebih jelas.

```
1  FUNCTION viewSensor(sensor_id = NULL)
2    load model
3    initialize data
4    IF sensor_id != NULL THEN
5      set data->userdata ← getUserData()
6      set data->sensordata ←
.        model->getSensorData(sensor_id)
7      set data->sensorreading ←
.        model->getSensorReading(sensor_id)
8      call load->view->view_sensor(data)
9    ENDIF
10 ENDFUNCTION
```

Kode Sumber 4.12 *Pseudocode* Fungsi *viewSensor*

Fungsi *viewSensor* di atas juga memanggil fungsi lain yaitu *getSensorData*. Fungsi ini digunakan untuk mendapatkan data informasi sensor berdasarkan pada *sensor_id*. Berikut ini Kode Sumber 4.13 menjabarkan secara lebih jelas.

```
1  FUNCTION getSensorData(sensor_id)
2    load model
3    initialize data
4    set data ← group_id
5    set db->select ← all
6    set db->from ← 'sensor'
```

```
7     set db->join ← "'sensor_group',
.       'sensor_group.id = sensor.group_id',
.       'left'"
8     set db->where ← 'sensor.id' = sensor_id
9     set query ← db->get
10    return query->result
11  ENDFUNCTION
```

Kode Sumber 4.13 *Pseudocode* Fungsi *getSensorData*

Fungsi lain yang juga dijalankan dalam fungsi *viewSensor* adalah fungsi *getSensorReading*. Fungsi berguna untuk mendapatkan data dari tabel *sensor_data* berdasarkan pada *sensor_id*. Data inilah yang akan ditampilkan menjadi riwayat data yang telah terekam. Berikut ini Kode Sumber 4.14 secara lebih jelas.

```
1   FUNCTION getSensorReading(sensor_id)
2     load model
3     set db->select ← all
4     set db->from ← 'sensor_data'
5     set db->where ← 'sensor_data.sensor_id' =
.       sensor_id
6     set db->order_by ← "'sensor_data.timestamp',
.       'ASC'"
7     set query ← db->get
8     return query->result
9   ENDFUNCTION
```

Kode Sumber 4.14 *Pseudocode* Fungsi *getSensorReading*

4.2.2 Implementasi Fungsi Lain (API)

Pada bagian ini akan dijelaskan sedikit mengenai fungsi – fungsi lain yang terkait dengan penerimaan data. Fungsi yang akan dibahas dalam bab ini merupakan fungsi yang cukup penting untuk berjalannya sistem. Beberapa fungsi inilah yang akan diakses *gateway* melalui HTTP *Request* untuk dapat berkomunikasi dengan server.

4.2.2.1 Fungsi Penerima Data Sensor

Fungsi yang akan dibahas pada subbab ini adalah fungsi yang berguna untuk menerima data sensor. Fungsi tersebut adalah fungsi *retrieveDataSensor*. Data yang dikirimkan melalui *gateway*, akan diterima melalui fungsi ini. Data diterima dengan *method post*. Data yang diterima meliputi *sensor_key*, *sensor_reading* (gabungan dari suhu, kelembaban, cahaya, suara), *status* dan *timestamp*.

Pada Kode Sumber 4.15 akan melakukan validasi *sensor_key* dengan fungsi *checkSensorKey*. Setelah itu, *sensor_reading* akan dipisah-pisahkan sesuai dengan jenis sensornya masing-masing. Data sensor yang telah dipisahkan akan dimasukkan dalam variabel *sensorReadingData* dan kemudian dimasukkan basis data melalui fungsi *insertSensorReadingData*. Setelah itu ada fungsi *updateLastUpdatedSensor* yang digunakan untuk memperbarui waktu terakhir data masuk pada tabel *sensor*.

```
1  FUNCTION retrieveDataSensor()
2    load model
3    initialize sensorReadingData
4    set sensor_key ← input->post('sensor_key')
5    set sensor_reading ←
.      input->post('sensor_reading')
7    set status ← input->post('status')
8    IF set tmp_time ← input->post('timestamp')
.      THEN
9      set timestamp ← tmp_time
10   ELSE
11     set timestamp ← date("Y-m-d H:i:s")
12   ENDIF
13   IF set sensor_id ←
.      model->checkSensorKey(sensor_key) THEN
14     set sensorReadingData->sensor_id ←
.        sensor_id
15     set sensorReadingData->timestamp ←
.        timestamp
16     set sensorReadingData->status ← status
17     set data_list ←
```

```
.       explode("+",sensor_reading)
18    FOR data in data_list
19      set data_temp ← explode(":", data)
20      IF data_temp[0] = "TEM" THEN
21        set sensorReadingData
.           ->temperature_reading ←
.           data_temp[1]
22      ELSE IF data_temp[0] = "HUM" THEN
23        set sensorReadingData
.           ->humidity_reading ← data_temp[1]
24      ELSE IF data_temp[0] = "LIG" THEN
25        set sensorReadingData->light_reading
.           ← data_temp[1]
26      ELSE IF data_temp[0] = "SOU" THEN
27        set sensorReadingData->sound_reading
.           ← data_temp[1]
28      ELSE IF data_temp[0] = "X" THEN
.         set sensorReadingData->x_reading
.           ← data_temp[1]
29      ELSE IF data_temp[0] = "Y" THEN
.         set sensorReadingData->y_reading
.           ← data_temp[1]
30      ELSE IF data_temp[0] = "Z" THEN
.         set sensorReadingData->z_reading
.           ← data_temp[1]
31      ENDIF
32    ENDFOR
33    call model->insertSensorReadingData(
.       sensorReadingData)
34    call model->updateLastUpdatedSensor(
.       sensor_id, timestamp)
35  ENDIF
36 ENDFUNCTION
```

Kode Sumber 4.15 *Pseudocode* Fungsi *retrieveDataSensor*

Kode Sumber 4.16 merupakan *pseudocode* dari fungsi yang digunakan untuk validasi *sensor_key*. Sedangkan secara berturut – turut Kode Sumber 4.17 dan Kode Sumber 4.18 adalah fungsi untuk *insert* data sensor baru dan fungsi untuk memperbarui waktu terakhir data sensor masuk.

```
1  FUNCTION checkSensorKey(sensor_key)
```

```
2     load model
3     set db->select ← 'sensor.id as SENSORID'
4     set db->from ← 'sensor'
5     set db->where ← "'sensor_key', sensor_key"
6     IF set query ← db->get THEN
7       return query->result
8     ELSE
9       return FALSE
10    ENDIF
11  ENDFUNCTION
```

Kode Sumber 4.16 *Pseudocode* Fungsi *checkSensorKey*

```
1  FUNCTION insertSensorReadingData(newData)
2    load model
3    select database
4    insert newData ← database
5  ENDFUNCTION
```

Kode Sumber 4.17 *Pseudocode* Fungsi *insertSensorReadingData*

```
1  FUNCTION updateLastUpdatedSensor(sensor_id,
2  newtime)
3    load model
4    set db->set ← "'last_updated', newtime"
5    set db->where ← "'id', sensor_id"
6    set db->update ← 'sensor'
7  ENDFUNCTION
```

Kode Sumber 4.18 *Pseudocode* Fungsi *updateLastUpdatedSensor*

4.2.2.2 Fungsi Memperbarui Lokasi

Fungsi pada subbab ini digunakan untuk memperbarui lokasi dari suatu kelompok sensor. Data lokasi yang dikirimkan berupa nilai *latitude* dan *longitude*. Nama fungsi ini adalah *updateLocation*. Berikut ini Kode Sumber 4.19 dari fungsi *updateLocation*.

```
1  FUNCTION updateLocation()
2    load model
3    initialize location
4    set group_id ← input->post('group_id')
5    set location->latitude ← input->
.      post('latitude')
7    set location->longitude ← input->
```

```
.       post('longitude')
8     call model-> updateGatewayLocation(group_id,
.       location)
9   ENDFUNCTION
```

Kode Sumber 4.19 *Pseudocode* Fungsi *updateLocation*

Adapun fungsi lain yang dijalankan untuk melakukan akses pada basis data adalah fungsi *updateGatewayLocation*. Berikut ini Kode Sumber 4.20 yang merupakan *pseudocode* fungsi *updateGatewayLocation*.

```
1   FUNCTION updateGatewayLocation(group_id,
2   location)
3     load model
4     set db->set ← location
5     set db->where ← "'id', group_id"
6     set db->update ← 'sensor_group'
7   ENDFUNCTION
```

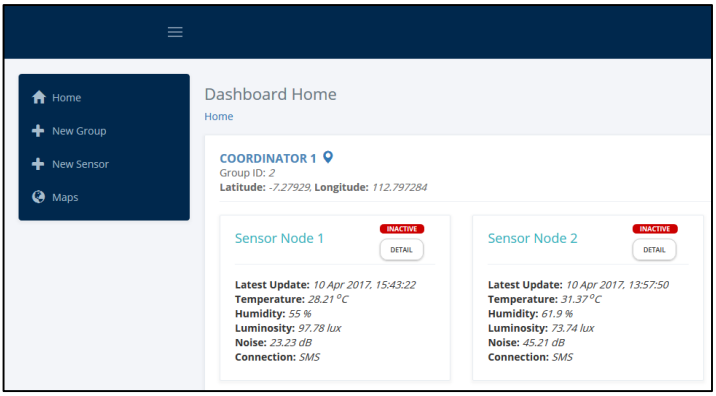
Kode Sumber 4.20 *Pseudocode* Fungsi *updateGatewayLocation*

4.2.3 Implementasi Antarmuka Pengguna *Sensor Cluster Management*

Berdasarkan tahap perancangan pada bab sebelumnya, disini akan dijelaskan mengenai implementasi dari perancangan antarmuka pengguna. Pada taha implementasi ini ada beberapa rancangan antarmuka yang akan diimplementasikan. Berikut ini antarmuka pengguna yang diimplementasikan.

4.2.3.1 Halaman Utama

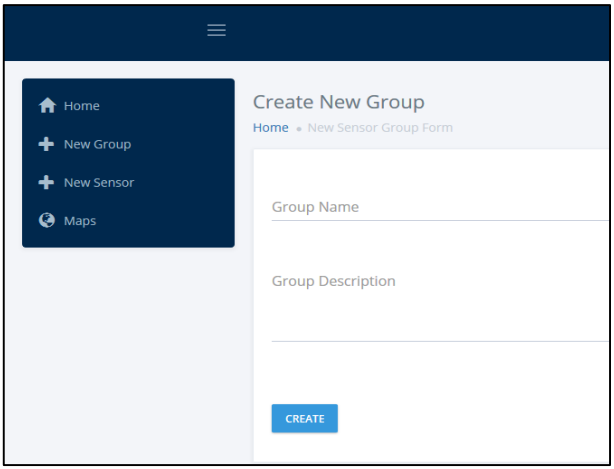
Halaman ini merupakan implementasi dari apa yang telah dirancang pada bab 3.3.3.1. Halaman ini menampilkan sensor – sensor berdasarkan kelompok sensor masing – masing. Gambar 4.1 merupakan implementasi halaman utama website.



Gambar 4.1 Implementasi Halaman Utama

4.2.3.2 Halaman *Form New Group*

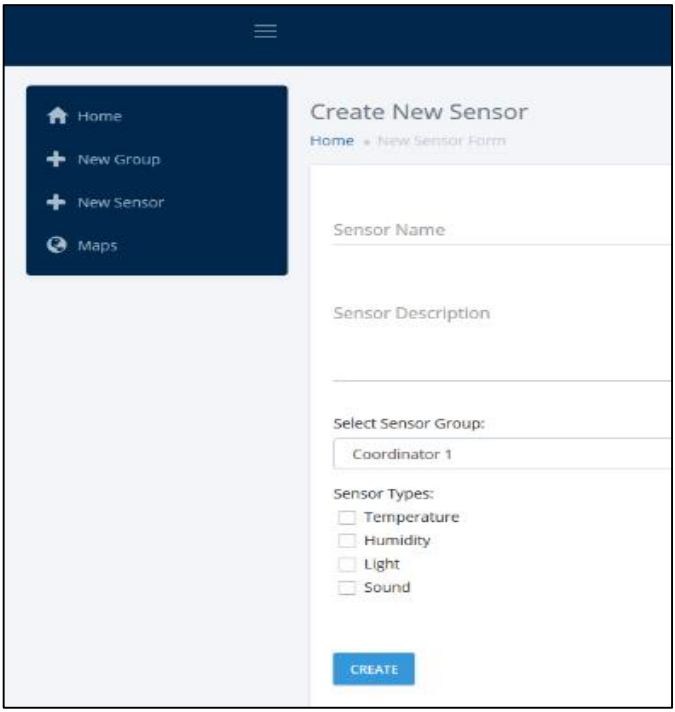
Halaman ini merupakan implementasi dari bab 3.3.3.2 yang digunakan untuk mendaftarkan kelompok sensor baru. Pendaftaran dilakukan dengan mengisi data – data yang diperlukan pada setiap *form* yang telah disediakan. Berikut ini tampilan antarmuka pada Gambar 4.2.



Gambar 4.2 Implementasi Halaman *Form New Group*

4.2.3.3 Halaman *Form New Group*

Halaman ini adalah imlementasi dari bab 3.3.1.3 yang berguna untuk mendaftarkan sensor baru. Pendaftaran dilakukan dengan mengisi data – data yang diperlukan pada setiap *form* yang telah disediakan. Tampilan antarmuka pada halaman ini dapat dilihat pada Gambar 4.3.

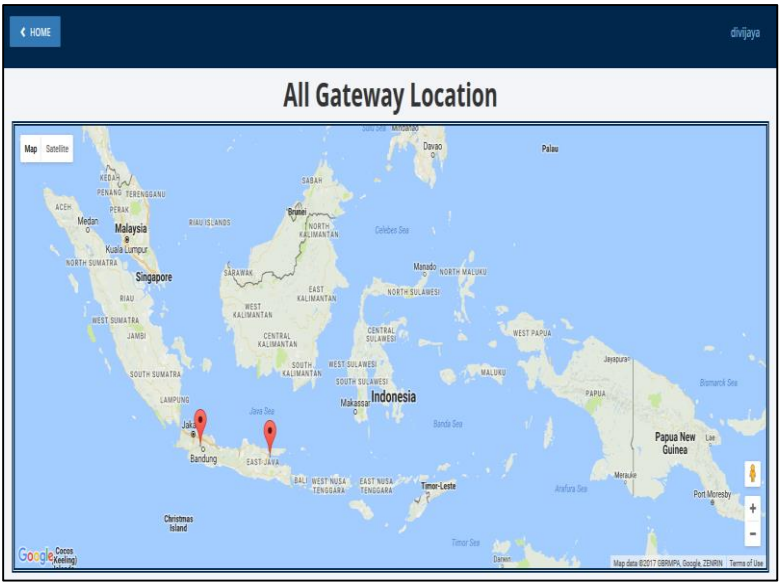


Gambar 4.3 Implementasi Halaman *Form New Sensor*

4.2.3.4 Halaman Peta Lokasi

Halaman peta lokasi terbagi menjadi dua (implementasi dari bab 3.3.3.4). Halaman peta lokasi pertama adalah halaman yang menampilkan peta lokasi secara menyeluruh. Peta lokasi ini akan

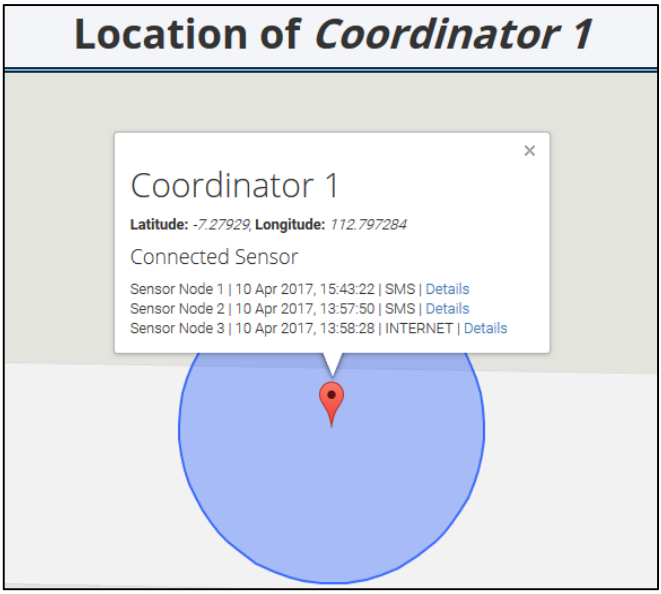
menampilkan keseluruhan kelompok sensor dalam satu peta yang sama. Peta lokasi keseluruhan ditunjukkan pada Gambar 4.4.



Gambar 4.4 Implementasi Halaman Peta Lokasi Keseluruhan

Sedangkan halaman peta lokasi yang kedua menampilkan peta lokasi yang lebih spesifik. Lokasi yang ditampilkan adalah lokasi spesifik dari satu kelompok sensor saja. Saat *marker* lokasi ditekan, maka akan menampilkan sedikit informasi mengenai *node* – *node* sensor yang terhubung.

Jika ingin melihat lebih rinci lagi mengenai sebuah sensor, dapat dipilih pada *link* yang tersedia pada tampilan peta lokasi. Gambar 4.5 merupakan tampilan dari peta lokasi khusus.



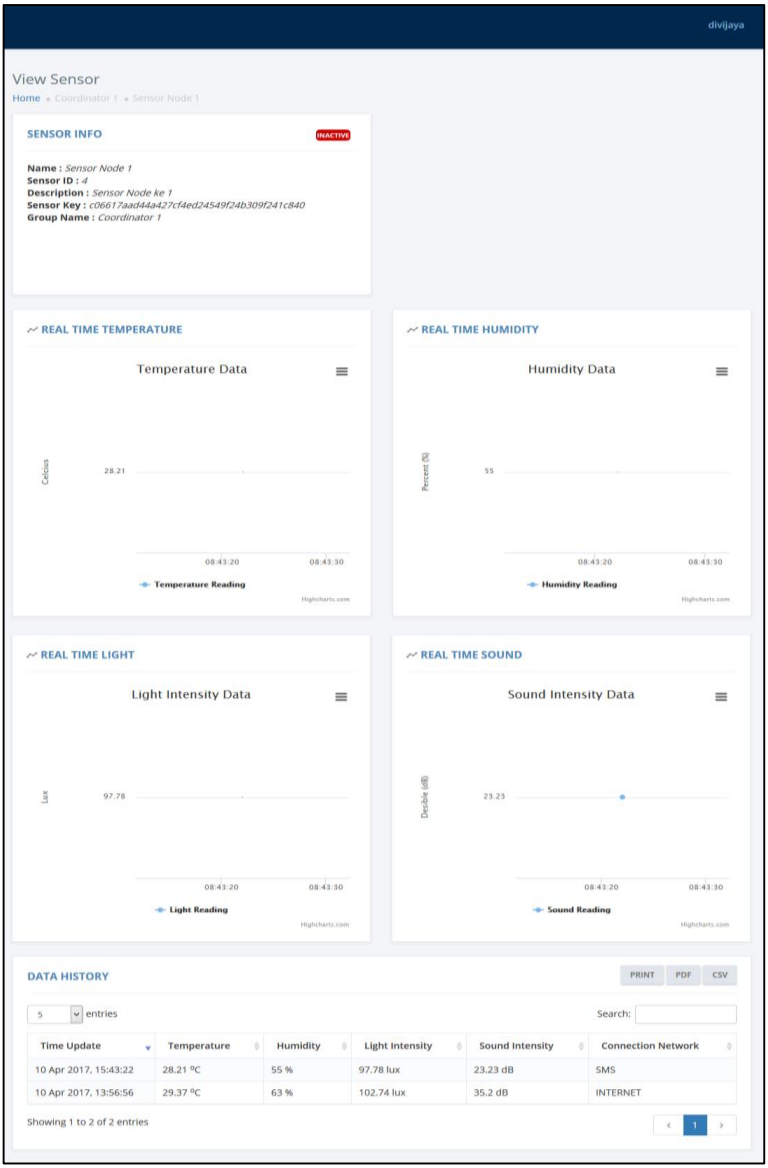
Gambar 4.5 Implementasi Halaman Peta Lokasi Khusus

4.2.3.5 Halaman Detail Sensor

Halaman ini adalah implementasi dari bab 3.3.1.6 yang menampilkan seluruh data sensor yang ada. Bagian teratas dari halaman ini menampilkan informasi sensor yang meliputi nama sensor (*sensor name*), sensor *id*, deskripsi sensor (*sensor description*), *key* dan nama kelompok *(group name*).

Terletak di bawahnya adalah tampilan dari grafik data sensor yang ditampilkan secara *realtime*. Grafik ini akan menampilkan data sensor baru yang masuk ke server.

Bagian terakhir yang ditampilkan pada halaman ini adalah riwayat pengiriman data (*data history*). Riwayat yang ditampilkan merupakan data – data seperti nilai sensor data (*suhu, humidity, light, sound, x, y* dan *z*), waktu dan jenis koneksi yang digunakan (internet atau sms). Gambar 4.6 merupakan implementasi dari halaman detail sensor.



Gambar 4.6 Implementasi Halaman Detail Sensor

4.3 Implementasi Android *Gateway*

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

4.3.1 Implementasi Fungsi dan Kelas (Android *Gateway*)

Sesuai dengan perancangan pada bab 3.4.1, jalannya aplikasi Android *gateway* ini terdiri dari tiga proses utama. Proses - proses tersebut meliputi *handleMessage*, *NormalTask*, *StrictTask* dan *OnLocationChanged*. Keempat proses tersebut merupakan inti dari Tugas Akhir ini.

Selain tiga proses yang telah disebutkan di atas, ada beberapa proses lain yang tidak kalah penting. Proses – proses tersebut seperti proses inisialisasi awal (*onCreate*), proses menyambungkan Android *gateway* dengan *node* sensor (*onResume*), fungsi mengirimkan data sensor (*postData*) dan fungsi memperbarui lokasi (*updateLocation*). Kedua fungsi yang disebutkan terakhir (tidak dijelaskan di sini) dijalan menggunakan mekanisme HTTP *Request* dengan bantuan pustaka bernama Volley.

4.3.1.1 Fungsi *onCreate*

Pada pemrograman Android, fungsi *onCreate* adalah sebuah fungsi yang berguna untuk inisialisasi awal. Fungsi ini dijalankan diawal waktu, yaitu saat sebuah aplikasi Android baru dijalankan. Pada penerapannya di sini, fungsi *onCreate* melakukan inisialisasi variabel – variabel yang diperlukan, serta menjalankan beberapa fungsi lain yaitu seperti fungsi untuk penjadwalan, pengatur lokasi dan inisialisasi Bluetooth. Kode Sumber 4.21 menjabarkan secara khusus mengenai fungsi *onCreate* dalam bentuk sebuah *pseudocode*.

```
1  @Override
```

```
 2  function onCreate(Bundle savedInstanceState)
 3    super.onCreate(savedInstanceState)
 4    initialize all variable
 5    set btAdapter ← getDefaultAdapter()
 .    call checkBTState()
 7
 8    set listener btnNormal(onClick
 9      @Override
10      function onClick()
11        call timer.cancel()
12        set timer ← new Timer()
13        set normalTask ← new NormalTask()
14        call timer.schedule(normalTask, 10min)
15    )
16    set listener btnStrict(onClick
17      @Override
18      function onClick()
19        call timer.cancel()
20        set timer ← new Timer()
21        set strictTask ← new StrictTask()
22        call timer.schedule(strictTask, 10min)
23    )
24
25    initialize locationManager
26    initialize locationListener
27
28    if (GPS.Permission != PERMISSION_GRANTED)
29      return
30    set loc ← locationManager.
 .      getLastKnownLocation
 .      (LocationManager.GPS_PROVIDER)
31    if(loc = NULL)
32      set loc ← locationManager.
 .        getLastKnownLocation
 .        (LocationManager.NETWORK_PROVIDER)
33      if(loc != NULL)
34        call locationManager.
 .          requestLocationUpdates
 .          (LocationManager.NETWORK_PROVIDER,
 .          60000, 0, locationListener)
35    else
36      call locationManager.requestLocationUpdates
 .        (LocationManager.GPS_PROVIDER,
 .        60000, 0, locationListener)
```

```
37    set geocoder ← new Geocoder(getBaseContext(),
.       Locale.getDefault())
38
39    initialize sensorManager = getSystemService()
40    call sensorManager.registerListener(ACCELERO)
```

Kode Sumber 4.21 *Pseudocode* Fungsi *onCreate*

4.3.1.2 Fungsi *onResume*

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

Saat fungsi ini berjalan, aktivitas yang terjadi adalah Android *gateway* akan mulai menyambungkan dirinya dengan *node* sensor yang ada (di dalam Tugas Akhir ini *node* sensor berjumlah 3) dengan menggunakan Bluetooth. Koneksi dilakukan pada *node* sensor sesuai dengan alamat Bluetooth yang telah disimpan sebelumnya (*hardware address*). Berikut ini Kode Sumber 4.22.

```
1   @Override
2   function onResume()
3     super.onResume()
4     initialize all variable
5     set device ← getRemoteDevice(address)
6     set device2 ← getRemoteDevice(address2)
7     set device3 ← getRemoteDevice(address3)
8
9     set btSocket ← createBluetoothSocket(device)
10    set btSocket2 ← createBluetoothSocket(
.       device2)
11    set btSocket3 ← createBluetoothSocket(
.       device3)
12
13    call btSocket.connect()
14    call btSocket2.connect()
15    call btSocket3.connect()
16
```

```
17    set mConnectedThread ← newConnectedThread(
.       btSocket)
18    call mConnectedThread.start()
19    set mConnectedThread2 ← new ConnectedThread(
.       btSocket2)
20    call mConnectedThread2.start()
21    set mConnectedThread3 ← new ConnectedThread(
.       btSocket3)
22    call mConnectedThread3.start()
```

Kode Sumber 4.22 *Pseudocode* Fungsi *onResume*

4.3.1.3 Fungsi *handleMessage*

Seperti yang pernah dijelaskan pada bab 3.4.1.1, fungsi *handleMessage* merupakan fungsi yang berguna untuk menerima pesan (data sensor) dari *node* sensor. Setiap ada pesan dari *node* sensor kepada Android *gateway*, maka fungsi inilah yang akan mengolah data yang masuk sebelum kemudian diproses untuk dapat ditampilkan di layar.

Susunan data masuk dari *node* sensor adalah '*ID#DATA*'. '*ID*' merupakan representasi dari *sensor_id* yang memiliki nilai sesuai dengan *sensor_id* dari masing – masing sensor. Sedangkan '*DATA*' merupakan nilai dari sensor yang telah didapatkan berdasarkan keadaan lingkungan (rincian susunan '*DATA*' akan dijelaskan pada pembahasan selanjutnya). Antara '*ID*' dan '*DATA*' dipisahkan dengan tanda pagar ('#') untuk lebih mudah dalam proses pengolahan selanjutnya. Susunan '*DATA*' sendiri juga masih dapat dijabarkan menjadi '*CODE*:*VALUE*+*CODE*:*VALUE*', seperti pada Gambar 3.29. Berikut ini Kode Sumber 4.23 tentang fungsi *handleMessage*.

```
1   function handleMessage(android.os.Message msg)
2     initialize all variable
3     switch (msg.what) :
4       case RECIEVE_MESSAGE:
5         set readBuf[] ← ([]) msg.obj
6         set strIncom ← new String(readBuf, 0,
.           msg.arg1)
7         call sb.append(strIncom)
```

```
 8          set dividerIndex ← sb.indexOf("#")
 9          set endOfLineIndex ← sb.indexOf("\r\n")
10            if (endOfLineIndex > 0)
11              set code ← sb.substring(0,
.                 dividerIndex)
12              set received_data ← sb.substring(
.                 dividerIndex+1, endOfLineIndex)
13
14              if (code = sensorId1)
15                set sensorData1 ← received_data
16                call txtSensor1.setText(sensorId1)
17                call txtData1.setText(sensorData1)
18                set data_list[] ←
.                   received_data.split("\\+")
19                for i ← 0 loop till i <
.                 data_list.length by i++ each step
20                  set temp_data[] ←
.                     data_list[i].split(":")
21                  if (temp_data[0] = "TEM")
22                    call temperature1.setText(
.                       temp_data[1])
23                  if(temp_data[0] = "HUM")
24                    call humidity1.setText(
.                       temp_data[1])
25                  if (temp_data[0] = "LIG")
26                    call light1.setText(
.                       temp_data[1])
27                  if (temp_data[0] = "SOU")
28                    call sound1.setText{
.                       temp_data[1])
29                set sensorStatus1 ← true
30
31              else if (code = sensorId2)
32                set sensorData2 ← received_data
33                call txtSensor2.setText(sensorId2)
34                call txtData2.setText(sensorData2)
35                set data_list[] ←
.                   received_data.split("\\+")
36                for i ← 0 loop till i <
.                 data_list.length by i++ each step
37                  set temp_data[] ←
.                     data_list[i].split(":")
38                  if (temp_data[0] = "TEM")
39                    call temperature2.setText(
```

```
.                       temp_data[1])
40                    if (temp_data[0] = "HUM")
41                      call humidity2.setText(
.                         temp_data[1])
42                    if (temp_data[0] = "LIG")
43                      call light2.setText(
.                         temp_data[1])
44                    if (temp_data[0] = "SOU")
45                      call sound2.setText(
.                         temp_data[1])
46                  set sensorStatus2 ← true
47
48                else if (code = sensorId3)
49                  set sensorData3 ← received_data
50                  call txtSensor3.setText(sensorId3)
51                  call txtData3.setText(sensorData3)
52                  set data_list[] ←
.                     received_data.split("\\+")
53                  for i ← 0 loop till i <
.                   data_list.length by i++ each step
54                    set temp_data[] ←
.                       data_list[i].split(":")
55                    if (temp_data[0] = "TEM")
56                      call temperature3.setText(
.                         temp_data[1])
57                    if (temp_data[0] = "HUM")
58                      call humidity3.setText(
.                         temp_data[1])
59                    if (temp_data[0] = "LIG")
60                      call light3.setText(
.                         temp_data[1])
61                    if (temp_data[0] = "SOU")
62                      call sound3.setText(
.                         temp_data[1])
63                    set sensorStatus3 ← true
64                call sb.delete(0, sb.length())
65              break
```

Kode Sumber 4.23 *Pseudocode* Fungsi *handleMessage*

4.3.1.4 Kelas *NormalTask*

Pada kelas *NormalTask* diatur mengenai mekanisme pengiriman data sensor ke server jika memilih *normal mode* yang

mengacu pada perancangan di bab 3.4.1.2. Penentuan cara pengiriman data ke server ditentukan dengan melihat kondisi ketersediaan koneksi internet. Saat koneksi internet tersedia maka pengiriman data sensor dari Android *gateway* ke server dilakukan dengan menggunakan koneksi internet. Sedangkan sebaliknya saat koneksi internet tidak tersedia, maka pengiriman data sensor dari Android *gateway* ke server dilakukan dengan mengirimkan sms.

Variabel *connectionCounter* digunakan sebagai penanda yang menentukan waktu. Setiap pertambahan satu nilai dari *connectionCounter* maka sama artinya dengan 10 menit. Nilai dari *connectionCounter* ini akan bertambah satu per satu setiap 10 menit jika tidak ada koneksi internet. Jika nilainya mencapai lebih dari 2 maka artinya tidak ada koneksi internet dalam waktu 30 menit, sehingga dilakukan pengiriman data melalui sms.

Sebelum melakukan pengiriman data, dilakukan pengecekan *sensorStatus* untuk menentukan apakah *node* sensor aktif atau tidak. Jika *node* sensor tidak aktif, maka tidak dilakukan pengiriman data dari sensor tersebut. Hal ini dilakukan untuk menghindari pengiriman data dari *node* sensor yang tidak aktif. Setiap akan melakukan pengiriman data baik melalui internet maupun sms, tidak lupa pula untuk mengembalikan nilai dari *connectionCounter* menjadi 0. Pengiriman sms sesuai dengan *format* yang telah dijelaskan pada bab 3.5.3 Perancangan Susunan (*Format*) SMS. Di bawah ini merupakan Kode Sumber 4.24 kelas *NormalTask*.

```
1   class NormalTask inherits TimerTask
2     @Override
3     function run()
4       runOnUiThread(new Runnable()
5
6         @Override
7         function run()
8           if (connectionCounter <= 2)
9
10            if (InternetAvailability() = true)
```

```
11              if (sensorStatus1 = true)
12                call postData(sensorKey1,
.                   sensorData1)
13                set sensorStatus1 ← false
14              if (sensorStatus2 = true)
15                call postData(sensorKey2,
.                   sensorData2)
16                set sensorStatus2 ← false
17              if (sensorStatus3 = true)
18                call postData(sensorKey3,
.                   sensorData3)
19                set sensorStatus3 ← false
20              set connectionCounter ← 0
21            else
22              set connectionCounter++
23
24            if (connectionCounter > 2)
25              initialize smsManager ←
.                 SmsManager.getDefault()
26              if (sensorStatus1 = true)
27                set sensorStatus1 ← false
28                call smsManager.sendTextMessage
.                   (gatewayPhoneNumber, NULL,
.                   "#" + sensorKey1 + "#" +
.                   sensorData1, NULL, NULL)
29              if (sensorStatus2 = true)
30                set sensorStatus2 := false
31                call smsManager.sendTextMessage
.                   (gatewayPhoneNumber, NULL,
.                   "#" + sensorKey2 + "#" +
.                   sensorData2, NULL, NULL)
32              if (sensorStatus3 = true)
33                set sensorStatus3 ← false
34                call smsManager.sendTextMessage
.                   (gatewayPhoneNumber, NULL,
.                   "#" + sensorKey3 + "#" +
.                   sensorData3, NULL, NULL)
35              set connectionCounter ← 0
36      )
```

Kode Sumber 4.24 *Pseudocode* Kelas *NormalTask*

4.3.1.5 Kelas *StrictTask*

Pada kelas *StrictTask* ini adalah proses yang dijalankan jika yang dijalankan adalah *strict mode* (mode ketat). Pada mode yang mengacu pada bab 3.4.1.3 ini, pengiriman data lebih ketat. Setiap sepuluh menit penjadwalan yang berjalan, jika tidak ada koneksi internet maka data akan dikirim ke server melalui koneksi internet. Sedangkan jika koneksi ineternet tidak tersedia, maka pengiriman data ke server akan langsung dilakukan melalui sms tanpa menunggu hingga waktu sepuluh menit ketiga (30 menit). Di bawah ini merupakan Kode Sumber 4.25 kelas *StrictTask*.

```
 1  class StrictTask inherits TimerTask
 2    @Override
 3    function run()
 4      runOnUiThread(new Runnable()
 5
 6        @Override
 7        function run()
 8
 9            if (InternetAvailability() = true)
10              if (sensorStatus1 = true)
11                call postData(sensorKey1,
  .                 sensorData1)
12                set sensorStatus1 ← false
13              if (sensorStatus2 = true)
14                call postData(sensorKey2,
  .                 sensorData2)
15                set sensorStatus2 ← false
16              if (sensorStatus3 = true)
17                call postData(sensorKey3,
  .                 sensorData3)
18                set sensorStatus3 ← false
19
20            else
21              initialize smsManager ←
  .               SmsManager.getDefault()
22              if (sensorStatus1 = true)
23                set sensorStatus1 ← false
24                call smsManager.sendTextMessage
  .                 (gatewayPhoneNumber, NULL,
```

```
.                     "#" + sensorKey1 + "#" +
.                     sensorData1, NULL, NULL)
25                  if (sensorStatus2 = true)
26                    set sensorStatus2 := false
27                    call smsManager.sendTextMessage
.                       (gatewayPhoneNumber, NULL,
.                       "#" + sensorKey2 + "#" +
.                       sensorData2, NULL, NULL)
28                  if (sensorStatus3 = true)
29                    set sensorStatus3 ← false
30                    call smsManager.sendTextMessage
31                      (gatewayPhoneNumber, NULL,
32                      "#" + sensorKey3 + "#" +
33                      sensorData3, NULL, NULL)
34      )
```

Kode Sumber 4.25 *Pseudocode* Kelas *StrictTask*

4.3.1.6 Fungsi *onLocationChanged*

Pada fungsi *OnLocationChange* dilakukan mekanisme untuk memperbarui lokasi dari kelompok sensor (implementasi dari bab 3.4.1.4). Saat ada perubahan lokasi dari maka nilai dari *latitude* dan *longitude* akan diperbarui juga. Setelah mendapatkan nilai *latitude* dan *longitude* terbaru, dilakukan pengecekan koneksi internet. Jika terdapat koneksi internet, maka akan dilakukan pengiriman data lokasi terbaru ke server melalui internet. Di bawah ini merupakan Kode Sumber 4.26 dari fungsi *onLocationChanged*.

```
1   @Override
2   function onLocationChanged(Location location)
3     initialize all variable
4     set latitude ← location.getLatitude()
5     set longitude ← location.getLongitude()
6     call txtLatitude.setText(latitude)
7     call txtLongitude.setText(longitude)
8
9     if(InternetAvailability())
10      call updateLocation()
```

Kode Sumber 4.26 *Pseudocode* Fungsi *onLocationChanged*

4.3.1.7 Fungsi *InternetAvailability*

Fungsi *InternetAvailability* digunakan untuk menentukan ketersediaan koneksi internet (implementasi dari bab 3.4.1.5). Pada prosesnya ada beberapa fungsi yang dijalankan. Dua fungsi pertama yaitu *isConnectedWifi* dan *isConnectedMobile* digunakan untuk mengetahui apakah *wifi* atau paket data aktif. Adapun fungsi *isConnectedFast* digunakan untuk menentukan ketersediaan koneksi internet berdasarkan jenis koneksi seperti pada Tabel 3.5. Proses terakhir adalah memastikan kecepatan koneksi internet tidak *UNKNOWN* maupun *POOR*. Agar lebih mudah memahami, berikut ini lebih lengkap pada Kode Sumber 4.27.

```
 1  function InternetAvailability()
 2    initialize all variable
 3    call mConnectionClassManager.reset()
 4    set mConnectionClass ←
 .      mConnectionClassManager.
 .      getCurrentBandwidthQuality()
 5    set mTries ← 0
 6    call new TestSpeed().execute(mURL)
 7    set conn ← new Connectivity()
 8
 9    if ((conn.isConnectedWifi() OR
 .    conn. isConnectedMobile()) AND
 .    conn.isConnectedFast()
 .    AND getCurrentBandwidthQuality() !=
 .    ConnectionQuality.UNKNOWN AND
 .    getCurrentBandwidthQuality() !=
 .    ConnectionQuality.POOR)
10      return true
11    else return false
```

Kode Sumber 4.27 *Pseudocode* Fungsi *InternetAvailabilty*

4.3.1.8 Fungsi *onSensorChanged*

Fungsi ini digunakan untuk mendapatkan nilai dari sensor setiap adanya perubahan dari nilai sensor. Sensor yang digunakan adalah sensor akselerometer dari Android. Kode Sumber 4.28 menjabarkan jalannya program secara lebih rinci.

```
1  @Override
2  function onSensorChanged(SensorEvent event)
3    initialize all variable
4    if(event.getType() = TYPE_ACCELEROMETER)
5      set xVal ← event.values[0]
6      set yVal ← event.values[1]
7      set zVal ← event.values[2]
8
9      set androidStatus ← true
10     call setText()
```

Kode Sumber 4.28 *Pseudocode* Fungsi *onSensorChanged*

4.3.2 Implementasi Antarmuka Pengguna (Android *Gateway*)

Gambar 4.7 Implementasi Antarmuka Pengguna (Android *Gateway*)

Pada bagian ini, akan dijelaskan mengenai implementasi antarmuka pengguna dari aplikasi Android *gateway* berdasarkan apa yang telah dirancang pada bab 3.4.2. Data – data penting yang ditampilkan pada antarmuka aplikasi adalah lokasi (ditunjukkan dengan *Latitude*, *Longitude*), *Sensor ID*, *Received Data* yaitu data sensor yang diperoleh dari data yang dikirimkan *node* sensor, *Temperature*, *Humidity*, *Luminosity* (sensor cahaya) dan *Noise* (sensor suara). Nilai dari setiap data akan senantiasa berubah seiring dengan data yang diterima dari *node* sensor. Gambar 4.7 merupakan implementasi antarmuka pengguna.

4.4 Implementasi SMS *Receiver*

Pada bagian ini akan dijelaskan mengenai implementasi dari *node* sms *receiver* (implementasi dari bab 3.5). *Node* ini akan dibuat dengan bantuan sebuah alat bantu sms *gateway* yaitu Gammu.

Node sms *receiver* bertugas untuk menerima sms dari Android *gateway* untuk kemudian diteruskan ke server. Setelah *node* sensor menerima sms, sms tersebut akan diolah secara otomatis dengan fungsi yang dibuat dalam bahasa pemrograman Python. *Node* ini dimungkinkan berupa sebuah modem yang dapat langsung dipasang pada komputer server atau bisa pula dipasang pada komputer yang berlainan.

4.4.1 Implementasi Konfigurasi *File*

Gammu merupakan alat bantu yang dapat digunakan sebagai penerima sms. Sebelum dapat dijalankan, ada beberapa berkas yang perlu untuk diatur sesuai dengan perangkat yang dimiliki. *File* yang perlu diatur untuk dapat menggunakan Gammu adalah *file gammurc* dan *gammudrc*.

4.4.1.1 Konfigurasi *File gammurc*

Berikut ini Kode Sumber 4.29 merupakan konfigurasi yang diisikan di dalam *file gammurc*.

```
[gammu]

device = com10:
connection = at115200
; comment if you'r unsure bout device's model
; model = MF 667
```

Kode Sumber 4.29 Konfigurasi *File gammurc*

4.4.1.2 Konfigurasi *File smsdrc*

Berikut ini Kode Sumber 4.30 merupakan konfigurasi yang diisikan di dalam *file smsdrc*.

```
[gammu]
device = com10:
model = MF 667
connection = at115200

[smsd]
service = SQL
PIN =
logfile = smsdlog
commtimeout = 30
sendtimeout = 30

# Database backends congfiguration
user = root
password =
pc = localhost
database = gammu

# DBI configuration
driver = native_mysql
```

Kode Sumber 4.30 Konfigurasi *File smsdrc*

4.4.2 Implementasi Fungsi (SMS *Receiver*)

Ada dua hal utama yang akan dijelaskan dalam bagian ini yaitu fungsi *main* dan fungsi *sendSensorData*. Kedua fungsi ini dibangun mengguanakan bahasa pemrograman Python. Berikut ini penjelasan lebih lengkapnya.

4.4.2.1 Fungsi *main*

Fungsi main merupakan proses utama sms *receiver*. Hal pertama yang dilakukan adalah menjalankan sebuah *query* untuk mendapatkan data yang ada pada basis data sms *receiver*. Tiga data yang diambil adalah *ID* (*primary key* dari sms), *TextDecoded* (isi pesan pendek) dan *ReceivingDateTime* (waktu masuknya sms). Semua data yang berhasil diperoleh akan diolah satu per satu dan dimasukkan ke dalam variabel *x*. Jika tidak ada satu pun data yang masuk, maka proses akan dihentikan selama 5 detik untuk kemudian melakukan pengecekan kembali.

Setelah didapatkan adanya sms, maka diambil nilai dari *x[1]* yang merupakan isi dari sms dan dimasukkan dalam variabel *message*. Kemudian dilakukan pengecekan karakter pertama yang ada pada isi pesan, jika karakter pertama tidak sama dengan karakter pagar ('#') maka status *Processed* data akan langsung dijadikan *true* tanpa mengolah pesan untuk dikirimkan ke server.

Jika diperoleh susunan pesan yang sesuai, maka pesan akan dipecah sesuai dengan susunan '*#KEY#DATA*'. '*KEY*' di sini merupakan nilai dari *sensor_key* dan '*DATA*' merupakan nilai dari data yang didapatkan pada *node* sensor. Setelah itu berjalanlah subproses *sendSensorData* dengan data yang dikirim meliputi *temp_key*, *temp_data*, *temp_status* dan *temp_time*. Berikut ini Kode Sumber 4.31 untuk lebih jelasnya.

```
1  initialize all variable
2  IF __name__ = "__main__":
3    set db ← MySQLdb.connect(host="localhost",
.        user="root", passwd="", db="gammu")
```

```
4     set cur ← db.cursor()
5
6     while(True):
7       call cur.execute("SELECT ID, TextDecoded,
.         ReceivingDateTime FROM inbox WHERE
.         Processed = 'false'")
8       set inbox ← cur.fetchall()
9       for x in inbox:
10        set message ← x[1]
11        IF (message[0] = '#'):
12          set temp_key ← message.split('#')[1]
13          set temp_data ← message.split('#')[2]
14          set temp_status ← "SMS"
15          set temp_time ← str(x[2])
16          call sendSensorData(temp_key,
.             temp_data, temp_status, temp_time)
17        ENDIF
18        call cur.execute("UPDATE inbox SET
.           Processed = 'true' WHERE
.           ID = %s", [x[0]])
19        call db.commit()
20      ENDFOR
21      call time.sleep(5)
```

Kode Sumber 4.31 *Pseudocode* Fungsi *main*

4.4.2.2 Fungsi *sendSensorData*

Pada fungsi *sendSensorData* yang merupakan implementasi dari bab 3.5.1, empat parameter yang terdiri dari *sent_key*, *sent_data*, *status* dan *time* akan dikirimkan ke server. Pengiriman data dilakukan dengan pustaka *urllib2*. Sebelum dikirimkan, data di-*encode* dalam mode *utf-8*. Proses pengiriman dilakukan dengan membuat *request_header* dan *request_body*. Tipe konten (*Content-Type*) yang digunakan adalah *'application/x-www-form-urlencoded*'. Data tersebut ditampilkan pada layar untuk mempermudah proses *debug*. Berikut ini Kode Sumber 4.32 untuk lebih jelasnya.

```
1   FUNCTION sendSensorData(sent_key, sent_data,
.   status, time):
2     initialize URL
```

```
3     set req_body ← 'sensor_key=' + sent_key
.       + '&sensor_reading=' + sent_data
.       + '&status=' + status
.       + '&timestamp=' + time
4     set req_header ← urllib2.Request(URL)
5     set req_header.add_header('Content-Type',
.       'application/x-www-form-urlencoded')
6     set response ← urllib2.urlopen(req_header,
.       req_body)
7   ENDFUNCTION
```

Kode Sumber 4.32 *Pseudocode* Fungsi *sendSensorData*

4.5 Implementasi *Node* Sensor

Seperti apa yang telah dijelaskan dalam bab 3.6, pada bab ini akan dijelaskan mengenai implementasi yang akan dilakukan untuk membangun *node* sensor. Hal yang akan dibahas di sini berkenaan dengan rangkaian utama (perangkat keras) dari *node* sensor dan fungsi – fungsi (perangkat lunak).

4.5.1 Implementasi Rangkaian Utama

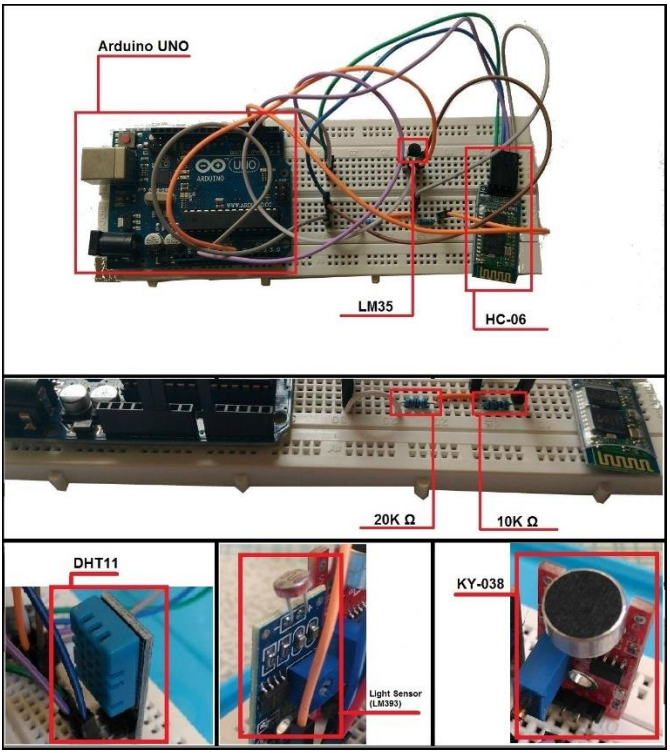
Node sensor ini akan diimplementasikan menggunakan mikrokontroler Arduino UNO R3. Mikrokontroler Arduino yang digunakan, akan dirangkai dan digabungkan dengan komponen – komponen yang lainnya. Sehingga dapat terbentuk sebuah alat sesuai apa yang telah dirancang sebelumnya yang dijabarkan pada Tabel 4.3.

Tabel 4.3 Komponen Rangkaian Implementasi *Node* Sensor

Perangkat Keras		Jumlah
Mikrokontroler	Arduino UNO Rev 3	Tiga buah
Sensor	LM35, DHT11, LM393 dan KY-038	Masing – masing tiga buah
Kabel	Kabel USB *Type A to Type B*	Tiga buah
	Kabel *jumper*	Tiga set (setiap set 36 – 45 buah)

Perangkat Keras		Jumlah
Board	*Breadboard*	Tiga buah
Transceiver	Modul Bluetooth HC-06	Tiga buah
Resistor	20K Ω	Tiga buah
	10K Ω	Tiga buah

Komponen lain yang akan digunakan yaitu modem HC-06. Modul ini merupakan modul pertukaran data (komunikasi) yaitu melalui Bluetooth. Selain itu ada juga komponen sensor yang digunakan adalah sensor suhu, kelembaban, cahaya dan suara. Gambar 4.8 memperlihatkan rangkaian utama yang telah diimplementasikan berdasarkan pada bab 3.6.1



Gambar 4.8 Implementasi Rangkaian Utama *Node* Sensor

Ada dua resistor yang digunakan untuk menyambungkan *PIN TX* dari HC-06 dengan Arduino. Resistor itu memiliki hambatan 20 Ohm dan 10 Ohm. Dua buah resistor tersebut digunakan agar modul Bluetooth tidak mudah rusak karena tegangan yang berlebihan. *Node* sensor yang akan dibuat berjumlah 3 buah. Pada Tabel 4.3 merinci komponen – komponen yang diperlukan dalam implementasi rangkaian utama.

Komponen atau perangkat keras yang digunakan tiap *node* sama satu dan lainnya. Implementasi terhadap perakitan komponen menghasilkan rangkaian yang sesuai dengan apa yang telah dirancang sebelumnya.

4.5.2 Implementasi Fungsi (*Node* Sensor)

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

Bahasa pemrograman Arduino merupakan bahasa pemrograman yang digunakan untuk mengendalikan komponen – komponen menggunakan mikrokontroler Arduino. Ada dua fungsi utama dalam Arduino, yaitu *setup* dan *loop*. Berikut ini penjelasannya.

4.5.2.1 Fungsi *setup*

Fungsi *setup* merupakan fungsi yang digunakan untuk inisialisasi awal yang merujuk pada perancangan di bab 3.6.2.1. Fungsi ini dijalankan hanya saat mikrokontroler baru dihidupkan (tersambung dengan sumber listrik). Di bawah ini Kode Sumber 4.33 yang menjabarkan mengenai fungsi *setup* dan inisialisasi variabel global (mengacu pada bab 3.6.2.1). Pertama – tama dilakukan inisialisasi *BT* dengan memberikan nilai *SoftwareSerial* pada *Pin* 2 dan *Pin* 3. Lalu nilai untuk *pin* dari sensor. Variabel

Code berisi nilai sesuai dengan *SENSOR_ID* masing – masing *node* sensor dan nilai sensor diawali dengan 0. Arduino dan Bluetooth akan berjalan dalam frekuensi *baud 9600*.

```
1  initialize SoftwareSerial BT(2, 3)
2  initialize all sensor pin
3  initialize all sensor variable
4  initialize code ← SENSOR_ID
5
6  FUNCTION setup()
7    open Serial.begin(BAUD_RATE_9600)
8    open BT.begin(BAUD_RATE_9600)
9  ENDFUNCTION
```

Kode Sumber 4.33 *Pseudocode* Fungsi *setup* dan Inisialisasi Variabel Global

4.5.2.2 Fungsi *loop*

Fungsi *loop* merupakan fungsi yang akan terus dilakukan seiring waktu selama mikrokontroler mendapat pasokan energi listrik. Setiap waktunya fungsi *loop* ini akan berhenti sejenak selama 5 detik setelah melakukan proses, sebelum kemudian berjalan kembali. Fungsi lain yang dijalankan adalah *updateTemperature*. Di dalam fungsi ini, nilai yang diambil secara langsung hanya nilai suhu. Sedangkan untuk kelembaban udara, intensitas cahaya dan intensitas suara, nilai didapatkan dengan nilai acak. Semua nilai sensor akan dikirimkan menggunakan Bluetooth dengan susunan '*SENSOR_ID#TEM:VALUE+HUM:VALUE+ LIG:VALUE+SOU:VALUE*'. Berikut ini Kode Sumber 4.34 untuk lebih jelasnya.

```
1  initialize all sensor variable
2  FUNCTION loop()
3    call updateSensor()
4    set temp_sign ← "TEM"
5    set hum_sign ← "HUM"
6    set lig_sign ← "LIG"
7    set sou_sign ← "SOU"
8    set temp ← (String) temperature
9    set hum ← (String) humidity
```

```
10    set lig ← (String) light
11    set sou ← (String) sound
12    call BT.write(code
.       + '#' + temp_sign + ':' + temp
.       + '+' + hum_sign + ':' + hum
.       + '+' + lig_sign + ':' + lig
.       + '+' + sou_sign + ':' + sou)
13    call delay(5000);
14  ENDFUNCTION
```

Kode Sumber 4.34 *Pseudocode* Fungsi *loop*

4.5.2.3 Fungsi *updateSensor*

Fungsi *updateSensor* merupakan fungsi yang digunakan untuk mendapatkan nilai terbaru dari keadaan yang ada di lingkungan. Fungsi ini merupakan implementasi dari apa yang telah dirancang pada bab 3.6.2.2. Nilai dari suhu udara terbaru akan dimasukkan pada variabel global.

Nilai dari keadaan lingkungan yang lain juga ditampung dalam variabel global. Variabel tersebut meliputi yaitu *humidity* yaitu untuk menampung nilai dari kelembaban udara. Variabel *light* untuk menampung nilai dari intensitas cahaya dan *sound* untuk menampung nilai dari intensitas suara. Berikut ini Kode Sumber 4.35 yang menjabarkan prosesnya.

```
1   initialize all sensor variable
2   FUNCTION updateSensor()
3     initialize analogValue
4     set analogValue ← analogRead(lm35Pin)
5     set temperature ← float(analogValue)/1023
6     set temperature ← (temperature*500)
7
      initialize chk
      set chk ← DHT.read11(dhtPin)
      set humidity = float(DHT.humidity)

      intialize soundValue
      set soundValue ← analogRead(soundPin)
      set sound ← float((1023-soundValue)/10.23)

      intialize lightValue
```

```
  set lightValue ← analogRead(lightPin)
  set light ← float((1023-lightValue)/10.23)
ENDFUNCTION
```

Kode Sumber 4.35 *Pseudocode* Fungsi *updateSensor*

BAB V
HASIL UJI COBA DAN EVALUASI

Pada bab ini akan dijabarkan mengenai uji coba dan evaluasi Tugas Akhir yang telah dikerjakan. Uji coba yang dilakukan secara garis besar terdiri dari dua hal utama yaitu uji coba fungsionalitas dan uji coba performa. Mekanisme uji coba dilakukan dengan mengikuti serangkaian skenario uji coba yang dibuat. Bagian akhir dari bab ini akan membahas mengenai evaluasi dari serangkaian uji coba yang telah dilakukan.

5.1 Lingkungan Uji Coba

Lingkungan uji coba mencakup perangkat dan peralatan apa saja yang digunakan dalam uji coba. Uji coba dilakukan di beberapa lokasi yang telah ditentukan sesuai dengan kebutuhan pada setiap skenario. Lingkungan uji coba memiliki spesifikasi seperti di bawah ini :

- Laptop ASUS ROG GL552JX
  - Intel(R) Core(TM) i7-4720HQ @2.6 GHz 2.59 GHz
  - RAM 8 GB
  - Microsoft Windows 8.1 Embedded 64-bit
- Tiga *node* sensor dengan setiap *node* memiliki komponen sebagai berikut :
  - Mikrokontroler Arduino UNO R3
  - Modul sensor suhu, kelembaban, cahaya dan suara
  - Modul *transceiver* Bluetooth HC-06
  - Papan kerja *breadboard*
  - Resistor 20K Ω dan 10K Ω
  - Satu set kabel *jumper* + kabel USB *Type A to Type B*
  - Baterai Panasonic 9V 220mAh
- *Smartphone* ASUS Zenfone Max ZC550KL
  - Qualcomm Snapdragon 400/410 @1.21 GHz
  - RAM 2 GB
  - Android 6.0.1 (Marsmallow)

- Perangkat SMS *Receiver* berupa Modem Turkcell 3G Vinn MF667

5.2 Skenario Uji Coba Fungsionalitas

Uji coba fungsionalitas merupakan sebuah pengujian yang dilakukan untuk menguji bekerja atau tidaknya fungsi – fungsi utama yang ada pada sistem. Fungsionalitas yang diuji dalam bagian ini yaitu fungsionalitas dari setiap *node*. *Node* yang dimaksud meliputi website, sms *receiver*, *node* sensor dan Android *gateway*.

5.2.1 Skenario Uji Coba (UJ-F01) - Menampilkan Sensor Berdasarkan Kelompok

Uji coba ini dijalankan dengan tujuan untuk menguji fungsionalitas dari website pada halaman utama. Pada halaman utama ditampilkan kelompok – kelompok beserta sensor – sensor yang terhubung dengannya. Pengujian fungsionalitas ini diawali dengan *login* pada website terlebih dahulu. Tabel 5.1 adalah tabel mengenai skenario uji coba menampilkan sensor berdasarkan kelompok (halaman utama).

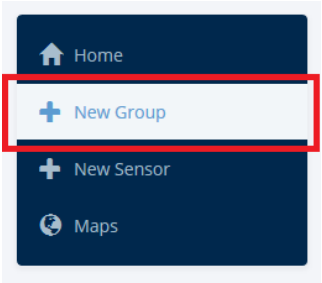
Tabel 5.1 Skenario Uji Coba Menampilkan Sensor Berdasarkan Kelompok

ID	**UJ-F01**
Nama	Uji Coba Menampilkan Sensor Berdasarkan Kelompok
Tujuan Uji Coba	Menguji fungsionalitas website untuk menampilkan sensor berdasarkan kelompok pada halaman utama.
Kondisi Awal	Membuka halaman website
Skenario	1. Membuka halaman website 2. *Login* sebagai Pengguna 3. Pengguna diarahkan ke halaman utama
Masukan	-
Keluaran	Data sensor berdasarkan kelompok

Hasil yang Diharapkan	Tampilan halaman utama yang berisi data – data sensor berdasarkan kelompok

5.2.2 Skenario Uji Coba (UJ-F02) - Mendaftarkan Kelompok Baru

Pada uji coba ini, dimaksudkan untuk menguji fungsionalitas pendaftaran kelompok baru. Uji coba ini diawali dengan memilih menu *New Group*. Berikut ini adalah menu *New Group* pada Gambar 5.1 yang ditunjuk dalam kotak warna merah.



Gambar 5.1 Menu *New Group*

Setelah memilih menu tersebut, kemudian akan muncul *form* untuk mendaftarkan kelompok baru. Setelah itu, yang perlu dilakukan adalah mengisi data yang meliputi nama kelompok dan deskripsi kelompok. Tabel 5.2 adalah tabel mengenai skenario uji coba mendaftarkan kelompok baru.

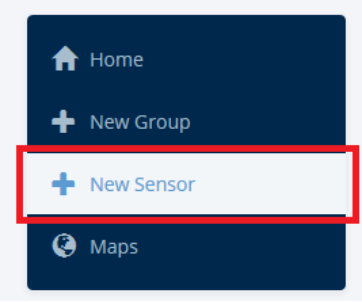
Tabel 5.2 Skenario Uji Coba Mendaftarkan Kelompok Baru

ID	**UJ-F02**
Nama	Uji Coba Mendaftarkan Kelompok Baru
Tujuan Uji Coba	Menguji fungsionalitas website untuk mendaftarkan kelompok baru.
Kondisi Awal	Website menampilkan halaman utama
Skenario	1. Website menampilkan halaman utama 2. Pengguna memilih menu *New Group* 3. Website menampilkan *form* pendaftaran kelompok baru

	4. Pengguna mengisi *form* sesuai dengan data yang tertulis pada *form* 5. Pengguna menekan tombol *Create* 6. Tampilan diarahkan kembali ke halaman utama
Masukan	Data – data yang terdiri dari nama kelompok dan deskripsi kelompok
Keluaran	Kelompok baru ditampilkan di halaman utama
Hasil yang Diharapkan	Kelompok baru berhasil didaftarkan dan ditampilkan pada halaman utama

5.2.3 Skenario Uji Coba (UJ-F03) - Mendaftarkan Sensor Baru

Selain uji coba mendaftarkan kelompok baru, uji coba pendaftaran yang lain yaitu uji coba mendaftar sensor baru. Hampir sama dengan mekanisme pendaftaran sebelumnya, yang pertama – tama harus dilakukan untuk mendaftarkan sensor baru adalah memilih menu *New Sensor*.



Gambar 5.2 Menu *New Sensor*

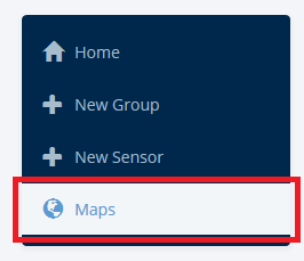
Gambar 5.2 merupakan gambar dari menu *New Sensor* yang ditunjuk dalam kotak warna merah. Setelah memilih menu tersebut akan muncul *form* untuk mendaftarkan sensor baru. Setelah itu, yang perlu dilakukan adalah mengisi data yang meliputi nama sensor, deskripsi sensor, pilihan kelompok sensor dan tipe – tipe sensor. Tabel 5.3 adalah tabel mengenai skenario uji coba mendaftarkan sensor baru.

Tabel 5.3 Skenario Uji Coba Mendaftarkan Sensor Baru

ID	**UJ-F03**
Nama	Uji Coba Mendaftarkan Sensor Baru
Tujuan Uji Coba	Menguji fungsionalitas website untuk mendaftarkan sensor baru.
Kondisi Awal	Website menampilkan halaman utama
Skenario	1. Website menampilkan halaman utama 2. Pengguna memilih menu *New Sensor* 3. Website menampilkan *form* pendaftaran sensor baru 4. Pengguna mengisi *form* sesuai dengan data yang tertulis pada *form* 5. Pengguna menekan tombol *Create* 6. Tampilan diarahkan kembali ke halaman utama
Masukan	Data – data yang terdiri dari nama sensor, deskripsi sensor, pilihan kelompok sensor dan tipe – tipe sensor
Keluaran	Sensor baru ditampilkan di halaman utama berdasarkan kelompok yang dipilih saat mendaftarkan sensor
Hasil yang Diharapkan	Sensor baru berhasil didaftarkan dan ditampilkan pada halaman utama dengan cara dikelompokkan sesuai dengan kelompok yang dipilih saat mendaftarkan sensor

5.2.4 Skenario Uji Coba (UJ-F04) - Menampilkan Peta Lokasi Keseluruhan

Bagian keempat yang akan diuji coba pada website adalah menampilkan peta lokasi keseluruhan. Uji coba ini dilakukan dengan tujuan untuk menguji lancar atau tidaknya halaman untuk menampilkan peta lokasi keseluruhan kelompok sensor. Langkah pertama untuk mengawali uji coba ini adalah dengan memilih menu *Maps*. Gambar 5.3 di bawah ini menunjukkan menu *Maps* yang ditunjuk dalam kotak warna merah.



Gambar 5.3 Menu *Maps*

Setelah memilih menu tersebut akan langsung diarahkan ke halaman ke sebuah peta lokasi keseluruhan. Tabel 5.4 adalah tabel mengenai skenario uji coba menampilkan peta lokasi keseluruhan.

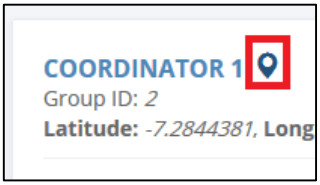
Tabel 5.4 Skenario Uji Coba Menampilkan Peta Lokasi Keseluruhan

ID	**UJ-F04**
Nama	Uji Coba Menampilkan Peta Lokasi Keseluruhan
Tujuan Uji Coba	Menguji fungsionalitas website dalam menampilkan keseluruhan peta lokasi kelompok.
Kondisi Awal	Website menampilkan halaman utama
Skenario	1. Website menampilkan halaman utama 2. Pengguna memilih menu *Maps* 3. Halaman website menampilkan peta lokasi dari keseluruhan kelompok yang telah didaftarkan
Masukan	-
Keluaran	Halaman peta lokasi yang menampilkan keseluruhan kelompok sensor yang telah didaftarkan
Hasil yang Diharapkan	Seluruh kelompok yang ada dapat ditampilkan dalam sebuah peta lokasi

5.2.5 Skenario Uji Coba (UJ-F05) - Menampilkan Peta Lokasi Khusus

Peta lokasi lain yang diuji coba adalah peta lokasi khusus. Uji coba menampilkan peta lokasi khusus ini bertujuan untuk

mengevaluasi fungsionalitas halaman peta lokasi khusus. Peta lokasi khusus yang dimaksudkan di sini adalah lokasi khusus dari suatu kelompok sensor, yang akan ditampilkan dalam sebuah peta lokasi. Mengawali uji coba ini, dilakukan dengan cara menekan ikon *location marker* yang ada di sebelah kanan nama kelompok sensor.



Gambar 5.4 Ikon *Location Marker*

Gambar 5.4 menunjukkan ikon *location marker* yang ditunjuk dalam kotak warna merah. Setelah memilih ikon tersebut akan langsung diarahkan ke halaman ke sebuah peta lokasi khusus. Tabel 5.5 adalah tabel mengenai skenario uji coba menampilkan peta lokasi khusus.

Tabel 5.5 Skenario Uji Coba Menampilkan Peta Lokasi Khusus

ID	**UJ-F05**
Nama	Uji Coba Menampilkan Peta Lokasi Khusus
Tujuan Uji Coba	Menguji fungsionalitas website dalam menampilkan peta lokasi khusus suatu kelompok.
Kondisi Awal	Website menampilkan halaman utama
Skenario	1. Website menampilkan halaman utama 2. Pengguna memilih ikon *location marker* di sebelah kanan nama kelompok 3. Halaman website menampilkan peta lokasi suatu kelompok yang dipilih untuk ditampilkan
Masukan	-
Keluaran	Halaman peta lokasi yang menampilkan lokasi khusus atau spesifik dari suatu kelompok yang dipilih
Hasil yang Diharapkan	Lokasi dari kelompok yang dipilih dapat ditampilkan dalam sebuah peta tersendiri

5.2.6 Skenario Uji Coba (UJ-F06) - Menampilkan Detail Sensor

Fungsionalitas website yang terakhir diuji coba adalah menampilkan detail sensor. Uji coba menampilkan detail sensor ini bertujuan untuk memantau fungsionalitas halaman detail sensor. Hal pertama yang perlu dilakukan untuk melihat detail sensor adalah menekan tombol *Detail* yang ada di samping kanan nama setiap sensor. Gambar 5.5 menampilkan tombol *Detail* yang ditunjukkan dalam kotak warna merah.

Gambar 5.5 Tombol *Detail*

Setelah menekan tombol tersebut, kemudian akan ditampilkan halaman detail sensor yang dipilih. Tabel 5.6 adalah tabel mengenai skenario uji coba menampilkan detail sensor.

Tabel 5.6 Skenario Uji Coba Menampilkan Detail Sensor

ID	**UJ-F06**
Nama	Uji Coba Menampilkan Detail Sensor
Tujuan Uji Coba	Menguji fungsionalitas website dalam menampilkan detail sensor.
Kondisi Awal	Website menampilkan halaman utama
Skenario	1. Website menampilkan halaman utama 2. Pengguna memilih tombol *Detail* di sebelah kanan nama setiap sensor 3. Halaman website menampilkan detail sensor
Masukan	-
Keluaran	Halaman detail sensor yang menampilkan informasi sensor, grafik data sensor dan riwayat data sensor keseluruhan
Hasil yang Diharapkan	Dapat menampilkan halaman detail sensor beserta seluruh data – data yang terkait

5.2.7 Skenario Uji Coba (UJ-F07) - Menerima SMS dan Mengirimakan Data ke Server

Pada uji coba ini akan dilakukan uji coba fungsionalitas untuk mencoba bisa atau tidaknya sms *receiver* menerima sms dan mengirimkan data yang telah diolah ke server. Hal yang perlu dilakukan untuk mengawali uji coba adalah mengirimkan sebuah sms ke nomer yang ada pada modem dengan isi pesan sesuai dengan susuan pada Gambar 3.29. Gambar 5.6 merupakan isi sms yang dikirimkan dalam rangkan uji coba.

Gambar 5.6 SMS yang dikirimkan untuk uji coba

Setelah mengirimkan sms, akan dipantau apakah sms yang dikirim dapat diterima dengan baik oleh sms *receiver* atau tidak. Bukan hanya itu saja, setelah sms diterima dan masuk ke *database* akan dipantau juga apakah data yang ada pada sms dapat diolah untuk kemudian dikirimkan ke server. Tabel 5.7 adalah tabel mengenai skenario uji coba.

Tabel 5.7 Skenario Uji Coba Menerima SMS dan Mengirimkan Data ke Server

ID	**UJ-F07**
Nama	Uji Coba Menerima SMS dan Mengirimkan Data ke Server

Tujuan Uji Coba	Menguji fungsionalitas sms *receiver* dalam menerima sms dan mengolah isi pesan untuk dikirimkan ke server.
Kondisi Awal	Pengguna mengirimkan sms sesuai susunan kepada nomer yang ada di dalam modem
Skenario	1. Pengguna mengirimkan sms 2. Dilakukan pemantauan apakah sms masuk ke basis data sms *receiver* 3. Jika iya, kemudian dilakukan pemantauan apakah pesan tersebut diolah dengan baik dan dikirimkan ke server 4. Memantau website untuk melihat apakah data masuk
Masukan	Sebuah sms sesuai *format*-nya
Keluaran	Tampilan website yang memuat data sesuai dengan data yang dikirimkan melalui sms
Hasil yang Diharapkan	Sms dapat diterima dan dikirimkan ke server

5.2.8 Skenario Uji Coba (UJ-F08) - Mendapatkan Nilai Sensor dan Mengirimkan Data ke Android *Gateway*

Fungsionalitas yang akan diuji coba di bagian ini adalah fungionalitas dari *node* sensor. Fungsi tersebut berkenaan dengan mendapatkan nilai sensor dan mengirimkan data ke Android *gateway* melalui koneksi Bluetooth. Hal yang dilakukan dalam melaksanakan uji coba ini adalah menyalakan *node* sensor secara bergantian dan memantau hasilnya satu per satu melalui *serial monitor* serta pada layar Android. Tabel 5.8 adalah tabel mengenai skenario uji coba.

Tabel 5.8 Skenario Uji Coba Mendapatkan Nilai Sensor dan Mengirimkan Data ke Android *Gateway*

ID	**UJ-F08**
Nama	Uji Coba Mendapatkan Nilai Sensor dan Mengirimkan Data ke Android *Gateway*

Tujuan Uji Coba	Menguji fungsionalitas setiap *node* sensor dalam mendapatkan nilai sensor suhu dari lingkungan dan mengirimkan data ke Android *gateway* melalui Bluetooth.
Kondisi Awal	Menyalakan *node* sensor dan Android *gateway*
Skenario	1. Menyalakan *node* sensor 1 dan Android *gateway* 2. Melakukan pemantauan suhu melalui *serial monitor* 3. Melakukan pemantauan data yang diterima Android *gateway* 4. Mengulangi langkah 1 – 3 dengan *node* sensor yang lain (*node* sensor 2 dan 3) secara bergantian
Masukan	Nilai sensor suhu
Keluaran	Tampilan nilai pada *serial monitor* dan data yang diterima Android *gateway*
Hasil yang Diharapkan	*Node* sensor bisa mendapatkan nilai sensor dan mengirimkan data yang sesuai ke Android *gateway*

5.2.9 Skenario Uji Coba (UJ-F09) - Koneksi dengan Satu Hingga Lebih dari Satu *Node* Sensor Sekaligus

Uji coba fungsionalitas ini dimaksudkan untuk menguji apakah Android *gateway* bisa terkoneksi dengan *node* sensor atau tidak. Uji coba dilakukan dengan mencoba mengkoneksikan satu *node* sensor dengan Android *gateway*. Setelah itu mencoba kembali mengkoneksikan Android *gateway* dengan 2 dan 3 *node* sensor sekaligus. Sukses atau tidaknya koneksi inilah yang akan dipantau dan dijadikan bahan evaluasi. Tabel 5.9 adalah tabel mengenai skenario uji coba.

Tabel 5.9 Skenario Uji Coba Koneksi dengan Satu Hingga Lebih dari Satu *Node* Sensor Sekaligus

ID	**UJ-F09**

Nama	Uji Coba Koneksi dengan Satu Hingga Lebih dari Satu *Node* Sensor Sekaligus
Tujuan Uji Coba	Menguji fungsionalitas Android *gateway* untuk dapat terhubung dengan satu hingga lebih dari satu *node* sensor sekaligus.
Kondisi Awal	Menyalakan Android *gateway* dengan satu, dua dan tiga *node* sensor sekaligus
Skenario	1. Menyalakan Android *gateway* dan satu *node* sensor 2. Melakukan pemantauan keberhasilan koneksi 3. Menyalakan Android *gateway* dan dua *node* sensor sekaligus 4. Melakukan pemantauan keberhasilan koneksi 5. Menyalakan Android *gateway* dan tiga *node* sensor sekaligus 6. Melakukan pemantauan keberhasilan koneksi
Masukan	Data sensor
Keluaran	Nyala lampu secara terus menerus pada modul Bluetooth HC-06 dan tampilan data pada layar Android
Hasil yang Diharapkan	Android *gateway* dapat terhubung dengan satu, dua dan tiga *node* sensor sekaligus

5.2.10 Skenario Uji Coba (UJ-F10) - Menjalankan Android *Gateway* di Beberapa Lokasi Berbeda

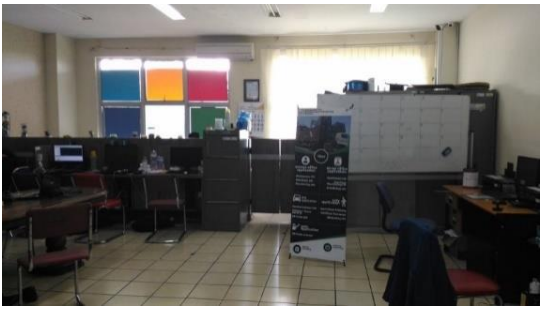
Tujuan dari uji fungsionalitas ini adalah untuk mengetahui apakah fungsi lokasi (*gps)* dapat berjalan dengan baik atau tidak. Uji coba dilakukan dengan cara menyalakan Android *gateway* pada beberapa lokasi berbeda. Pada setiap lokasi yang ada akan dilakukan pemantauan melalui website apakah lokasi dalam website sama dengan lokasi pengujian. Tabel 5.10 adalah tabel mengenai skenario uji coba.

Tabel 5.10 Skenario Uji Coba Menjalankan Android *Gateway* di Beberapa Lokasi Berbeda

ID	**UJ-F10**

Nama	Uji Coba Menjalankan Android *Gateway* di Beberapa Lokasi Berbeda
Tujuan Uji Coba	Menguji fungsionalitas Android *gateway* mengenai penggunaan fungsi lokasi (*gps*).
Kondisi Awal	Menyalakan Android *gateway* di beberapa lokasi berbeda
Skenario	1. Menyalakan Android *gateway* pada lokasi pertama 2. Melakukan pemantauan lokasi pada website 3. Menyalakan Android *gateway* pada lokasi kedua 4. Melakukan pemantauan lokasi pada website 5. Menyalakan Android *gateway* pada lokasi ketiga 6. Melakukan pemantauan lokasi pada website
Masukan	Data lokasi (*latitude, longitude*)
Keluaran	Tampilan lokasi kelompok secara khusus dalam website
Hasil yang Diharapkan	Android *gateway* dapat mengirimkan lokasi keberadaannya ke server

Lokasi yang dijadikan tempat uji coba fungsionalitas ini ada tiga. Lokasi pertama seperti Gambar 5.7 berada di Laboratorium KBJ Teknik Informatika ITS.



Gambar 5.7 Lokasi Pertama

Lokasi yang kedua pada Gambar 5.8 berada di Jalan Kejawan Putih Tambak 22A yang merupakan lokasi domisili Penulis.

Gambar 5.8 Lokasi Kedua

Sedangkan untuk lokasi ketiga seperti Gambar 5.9 berada di depan Kantin Pusat ITS.

Gambar 5.9 Lokasi Ketiga

5.2.11 Skenario Uji Coba (UJ-F11) - Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dinyalakan

Pengujian selanjutnya yang akan dilakukan adalah pengujian Android *gateway* dalam keadaan koneksi internet dinyalakan. Tujuan dari uji coba ini adalah untuk menguji lancar atau tidaknya fungsionalitas Android *gateway* untuk mengirimkan data melalui internet. Uji coba diawali dengan menyalakan

Android *gateway* dan menyalakan wifi pada *smartphone*. Android *gateway* akan dibiarkan menyala hingga waktu sepuluh menit. Setelah itu akan diamati apakah Android *gateway* dapat mengirimkan data melalui internet. Tabel 5.11 merupakan tabel mengenai skenario uji coba.

Tabel 5.11 Skenario Uji Coba Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dinyalakan

ID	**UJ-F11**
Nama	Uji Coba Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dinyalakan
Tujuan Uji Coba	Menguji fungsionalitas Android *gateway* untuk mengirimkan data melalui internet sesuai waktu setiap sepuluh menit
Kondisi Awal	Menyalakan Android *gateway* dalam keadaan paket data dinyalakan
Skenario	1. Menyalakan Android *gateway* dalam keadaan wifi dinyalakan 2. Menyalakan 3 *node* sensor 3. Menunggu sepuluh menit hingga muncul notifikasi pengiriman data melalui internet 4. Membuka basis data pada tabel *sensor_data* 5. Memantau apakah data yang dikirimkan masuk ke dalam basis data
Masukan	Data sensor
Keluaran	Data yang masuk di dalam basis data
Hasil yang Diharapkan	Android *gateway* dapat mengirimkan data sensor ke server melalui internet

5.2.12 Skenario Uji Coba (UJ-F12) - Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dimatikan

Hampir sama dengan uji coba sebelumnya, pengujian selanjutnya yang akan dilakukan adalah pengujian Android *gateway* dalam kondisi internet dimatikan. Tujuan dari uji coba ini adalah untuk menguji lancar atau tidaknya fungsionalitas Android *gateway* untuk mengirimkan data melalui sms. Uji coba diawali

dengan menyalakan Android *gateway* namun mematikan paket data pada *smartphone*. Android *gateway* akan dibiarkan menyala hingga waktu sepuluh menit ketiga atau dalam waktu tiga puluh menit. Setelah itu akan diamati dalam keadaan tidak ada koneksi internet, apakah Android *gateway* dapat mengirimkan data melalui sms. Tabel 5.12 merupakan tabel mengenai skenario uji coba.

Tabel 5.12 Skenario Uji Coba Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dimatikan

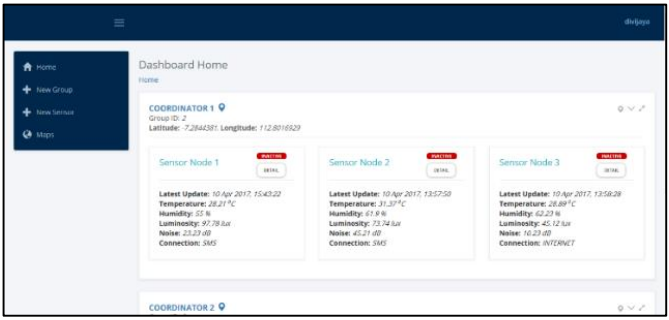
ID	**UJ-F12**
Nama	Uji Coba Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dimatikan
Tujuan Uji Coba	Menguji fungsionalitas Android *gateway* untuk mengirimkan data melalui sms jika tidak ada koneksi internet hingga sepuluh menit ketiga (30 menit)
Kondisi Awal	Menyalakan Android *gateway* dalam keadaan paket data dimatikan
Skenario	1. Menyalakan Android *gateway* dalam keadaan paket data dimatikan 2. Menyalakan 3 *node* sensor 3. Menunggu tiga puluh menit hingga muncul notifikasi pengiriman data melalui sms 4. Memantau apakah sms berhasil dikirimkan 5. Membuka basis data gammu dan memantau apakah sms berhasil diterima 6. Melihat Python *terminal* dan memantau apakah data yang masuk berhasil diolah dan dikirimkan ke server 7. Membuka basis di server pada tabel *sensor_data* dan memantau apakah data berhasil masuk ke server
Masukan	Data sensor
Keluaran	Data yang masuk di dalam basis data
Hasil yang Diharapkan	Android *gateway* dapat mengirimkan data sensor ke server melalui sms

5.3 Hasil Uji Coba Fungsionalitas

Telah dijabarkan pada bagian sebelumnya mengenai skenario dari keseluruhan uji coba fungsionalitas. Skenario uji coba yang telah dijabarkan terdiri dari beberapa hal. Secara garis besar, pengujian terhadap fungsionalitas terbagi menjadi empat yaitu fungsionalitas website, sms *receiver*, *node* sensor dan Android *gateway*.

5.3.1 Hasil Uji Coba (UJ-F01) - Menampilkan Sensor Berdasarkan Kelompok

Menurut skenario pada bab 5.2.1, Pengguna pada awalnya akan *login* ke dalam website dengan menggunakan *username* dan *password* yang telah dimiliki. Setelah berhasil *login*, maka akan ditampilkan sebuah halaman utama yang menyajikan sensor – sensor yang telah disesuaikan dengan kelompok masing – masing. Pada uji coba yang telah dilakukan, didapatkan tampilan seperti pada Gambar 5.10. Hal ini menunjukkan bahwa fungsionalitas menampilkan sensor berdasarkan kelompok bekerja dengan baik.

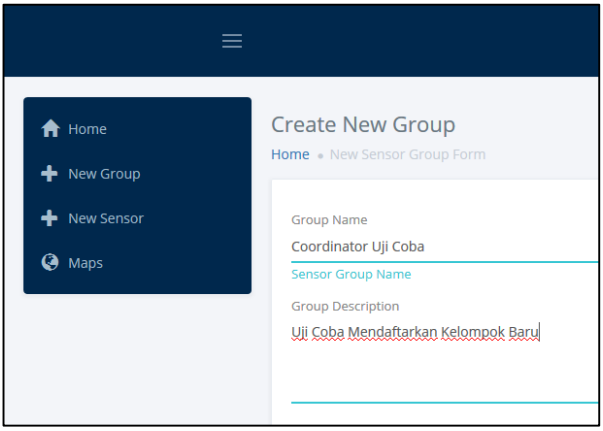


Gambar 5.10 Hasil Uji Coba UJ-F01 (Halaman Utama)

5.3.2 Hasil Uji Coba (UJ-F02) - Mendaftarkan Kelompok Baru

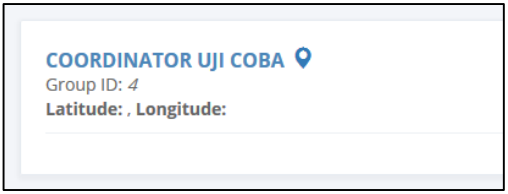
Mengacu pada skenario uji coba di bab 5.2.2, mula – mula uji coba dilakukan dengan memilih menu *New Group* sesuai

dengan Gambar 5.1. Kemudian akan muncul sebuah *form* untuk mendaftarkan kelompok baru. *Form* tersebut diisi sesuai dengan keterangan yang ada. Data yang dimasukkan pada saat proses uji coba dapat dilihat pada Gambar 5.11.



Gambar 5.11 Isian Data pada *Form New Group*

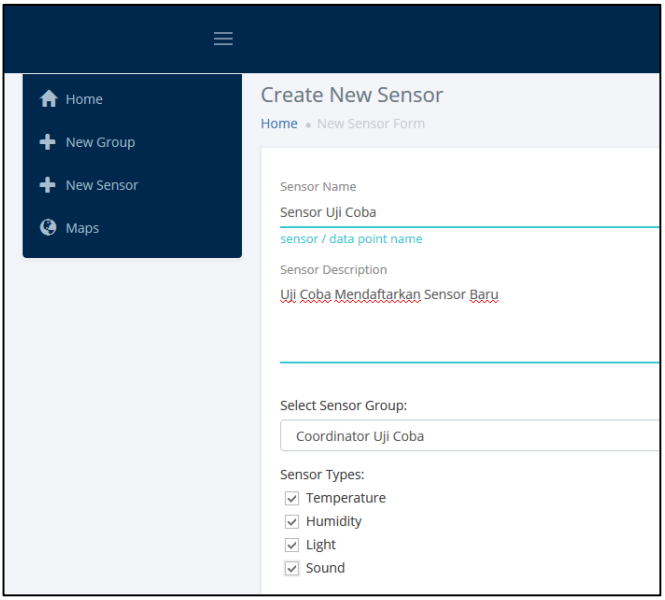
Setelah seluruh data diisikan, proses pendaftaran akan dijalankan dengan cara menekan tombol *Create*. Secara otomatis tampilan akan dialihkan menuju halaman utama dan kelompok baru berhasil didaftarkan. Gambar 5.12 menunjukkan bahwa kelompok baru yang berhasil didaftarkan. Hal ini menunjukkan bahwa fungsionalitas mendaftarkan kelompok baru berjalan lancar.



Gambar 5.12 Hasil Uji Coba UJ-F02 Berupa Tampilan Kelompok Baru

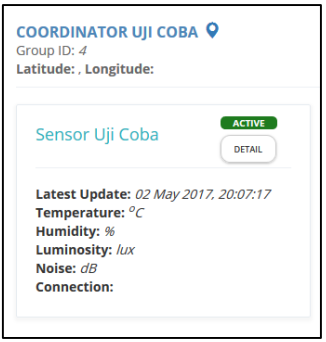
5.3.3 Hasil Uji Coba (UJ-F03) - Mendaftarkan Sensor Baru

Uji coba lain yang hampir sama dengan uji coba sebelumnya adalah uji coba mendaftarkan sensor baru. Sesuai dengan bab 5.2.3, hal pertama yang dilakukan untuk mengawali uji coba yaitu dengan memilih menu *New Sensor* yang tertera pada Gambar 5.2. Saat muncul *form* untuk mendaftarkan sensor baru, dimasukkan data – data sesuai yang diperlukan. Data yang dimasukkan pada saat proses uji coba dapat dilihat pada Gambar 5.13.



Gambar 5.13 Isian Data pada *Form New Sensor*

Setelah seluruh data diisikan, proses pendaftaran akan dijalankan dengan cara menekan tombol *Create*. Secara otomatis tampilan akan dialihkan menuju halaman utama dan sensor baru berhasil didaftarkan. menunjukkan bahwa sensor baru yang berhasil didaftarkan. Uji coba menunjukkan bahwa fungsionalitas mendaftarkan sensor baru berjalan lancar.



Gambar 5.14 Hasil Uji Coba UJ-F03 Berupa Tampilan Sensor Baru

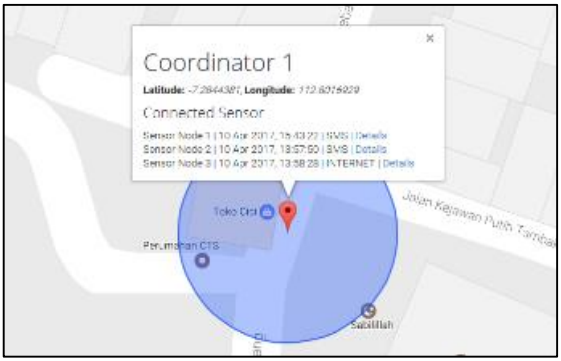
5.3.4 Hasil Uji Coba (UJ-F04) - Menampilkan Peta Lokasi Keseluruhan

Uji coba fungsionalitas ini diawali dengan memilih menu *Maps* sesuai yang ditunjukkan pada skenario di bab 5.2.4. Setelah memilih menu *Maps* tersebut, ditampilkan sebuah peta lokasi keseluruhan dari kelompok – kelompok sensor yang ada. Hasil tampilan peta lokasi keseluruhan ditampilkan pada Gambar 5.15. Berdasarkan uji coba yang dilakukan, didapatkan hasil yang menunjukkan bahwa fungsionalitas menampilkan peta lokasi keseluruhan dapat berjalan dengan semestinya.

Gambar 5.15 Hasil Uji Coba UJ-F04 Berupa Tampilan Peta Lokasi Keseluruhan

5.3.5 Hasil Uji Coba (UJ-F05) - Menampilkan Peta Lokasi Khusus

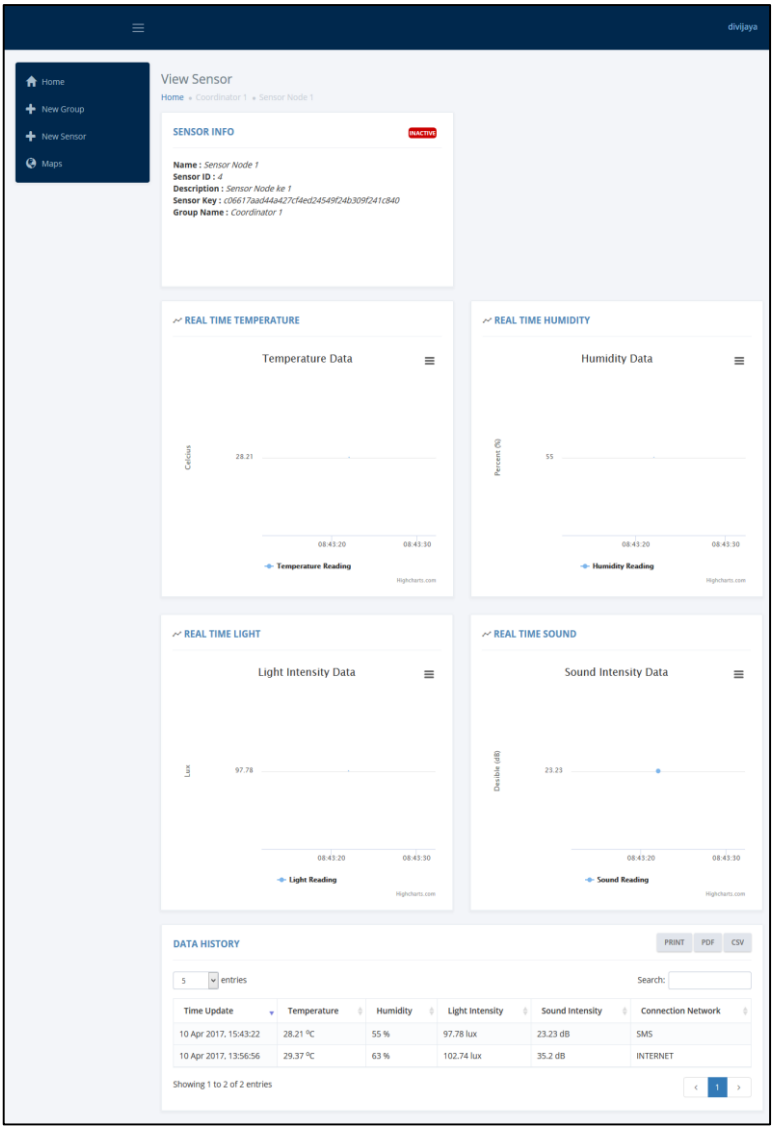
Hampir sama dengan uji coba sebelumnya, uji coba ini juga digunakan untuk menampilkan peta lokasi namun lokasi yang lebih spesifik. Skenario uji coba dapat dilihat pada bab 5.2.5. Langkah yang dilakukan pertama – tama adalah memilih ikon *location marker* (Gambar 5.4) di sebelah kanan nama kelompok. Setelah memilih ikon tersebut, akan ditampilkan sebuah peta lokasi khusus (spesifik) dari sebuah kelompok sensor. Hasil tampilan peta lokasi khusus ditampilkan pada Gambar 5.16. Berdasarkan uji coba yang dilakukan, dapat dilihat bahwa fungsionalitas menampilkan peta lokasi khusus berjalan dengan baik.



Gambar 5.16 Hasil Uji Coba UJ-F05 Berupa Tampilan Peta Lokasi Khusus

5.3.6 Hasil Uji Coba (UJ-F06) - Menampilkan Detail Sensor

Sesuai dengan skenario pada bab 5.2.6, uji coba fungsionalitas ini diawali dengan memilih tombol *Detail* yang ditunjukkan pada Gambar 5.5. Setelah itu akan ditampilkan rincian dari sensor yang dipilih. Hasil uji coba yang berhasil didapatkan dari pengujian fungsionalitas ini dapat dilihat pada Gambar 5.17. Berdasarkan hasil yang didapatkan tersebut, fungsionalitas menampilkan detail sensor bekerja dengan baik.

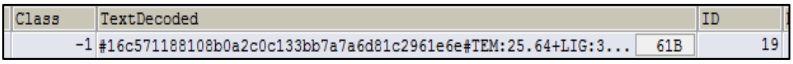


Gambar 5.17 Hasil Uji Coba UJ-F06 Berupa Tampilan Halaman Detail Sensor

5.3.7 Hasil Uji Coba (UJ-F07) - Menerima SMS dan Mengirimakan Data ke Server

Sesuai dengan uji coba pada sub bagian 5.2.7 dan rincian skenario pada Tabel 5.7, pengujian diawali dengan mengirimkan sebuah pesan teks atau sms. Isi dari pesan yang dikirimkan disesuaikan dengan *format* yang telah dibuat sebelumnya. Pesan yang dikirimkan untuk pengujian dapat dilihat pada Gambar 5.6.

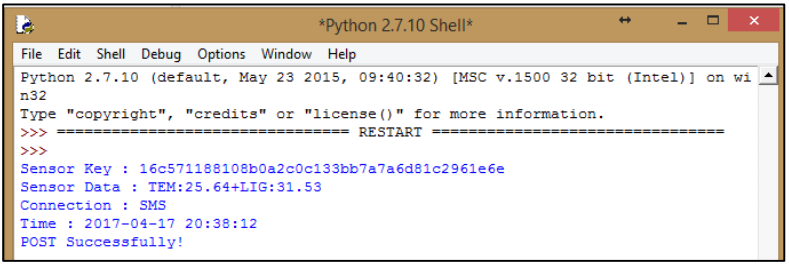
Pada pengujian yang telah dilakukan, pesan yang dikirimkan berhasil diterima oleh sms *receiver*. Pesan yang diterima, masuk ke dalam basis data gammu seperti pada Gambar 5.18 di bawah ini.



Class	TextDecoded		ID
-1	#16c571188108b0a2c0c133bb7a7a6d81c2961e6e#TEM:25.64+LIG:3...	61B	19

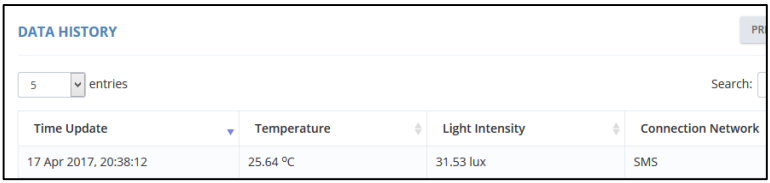
Gambar 5.18 SMS yang diterima dalam basis data gammu

Pesan yang telah diterima tersebut kemudian diolah dengan menggunakan sebuah program yang buat dalam bahasa Python. Setelah pesan diolah, kemudian akan dikirimkan ke server. Berikut ini pada Gambar 5.19 merupakan status pengiriman data ke server.



Gambar 5.19 Status pengiriman data ke server

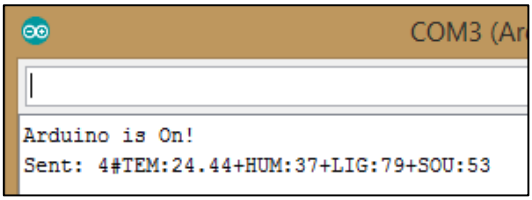
Data yang telah dikirimkan kemudian masuk ke server dan disimpan ke dalam basis data di server. Gambar 5.20 merupakan tampilan halaman website yang menunjukkan data yang masuk ke server. Berdasarkan hasil yang didapatkan pada gambar, dapat dikatakan bahwa fungsionalitas menerima sms dan mengirimkan data ke server berjalan dengan semestinya.



Gambar 5.20 Hasil Uji Coba UJ-F07 Berupa Tampilan Website yang Menunjukkan Data SMS Masuk

5.3.8 Hasil Uji Coba (UJ-F08) - Mendapatkan Nilai Sensor dan Mengirimkan Data ke Android *Gateway*

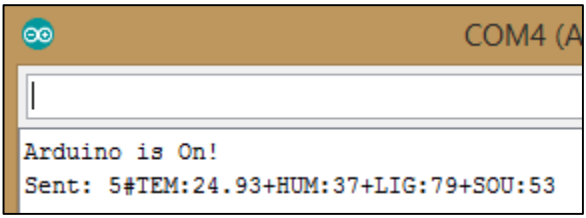
Pengujian dilakukan sesuai dengan skenario pada bab 5.2.8. Langkah pertama yang dijalankan adalah menyalakan Android *gateway* dan *node* sensor 1. Saat *node* sensor 1 dijalankan, berdasarkan pemantauan *serial monitor* pada Gambar 5.21, nilai sensor berhasil didapatkan. Selain itu, pada sisi Android *gateway* juga didapatkan hasil yang menunjukkan berhasilnya pengiriman data dari *node* sensor 1 ke Android *gateway*. Data yang berhasil dikirimkan ditunjukkan pada Gambar 5.22.



Gambar 5.21 *Serial monitor* pada *Node* Sensor 1

Gambar 5.22 Hasil Uji Coba UJ-F08 berupa data yang diterima Android *Gateway* dari *Node* Sensor 1

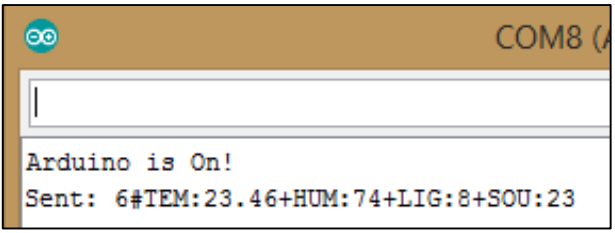
Uji coba dilanjutkan dengan melakukan pengujian pada *node* sensor 2. Saat *node* sensor 2 dijalankan, berdasarkan pemantauan *serial monitor* pada Gambar 5.23, nilai sensor berhasil didapatkan. Selain itu, pada sisi Android *gateway* juga didapatkan hasil yang menunjukkan berhasilnya pengiriman data dari *node* sensor 2 ke Android *gateway*. Data yang berhasil dikirimkan ditunjukkan pada Gambar 5.24.



Gambar 5.23 *Serial monitor* pada *Node* Sensor 2

Gambar 5.24 Hasil Uji Coba UJ-F08 berupa data yang diterima Android *Gateway* dari *Node* Sensor 2

Pengujian ketiga dilakukan pada *node* sensor 3. Saat *node* sensor 3 dijalankan, berdasarkan pemantauan *serial monitor* pada Gambar 5.25, nilai sensor berhasil didapatkan. Selain itu, pada sisi Android *gateway* juga didapatkan hasil yang menunjukkan berhasilnya pengiriman data dari *node* sensor 3 ke Android *gateway*. Data yang berhasil dikirimkan ditunjukkan pada Gambar 5.26.



Gambar 5.25 *Serial monitor* pada *Node* Sensor 3

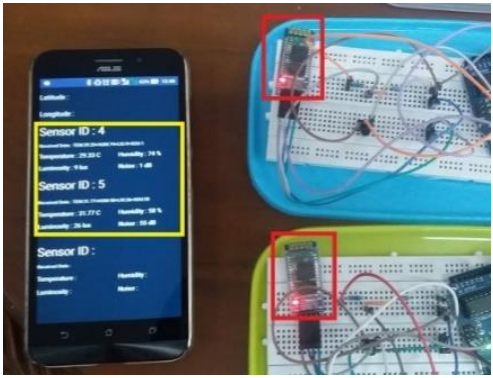
Gambar 5.26 Hasil Uji Coba UJ-F08 berupa data yang diterima Android *Gateway* dari *Node* Sensor 3

5.3.9 Hasil Uji Coba (UJ-F09) - Koneksi dengan Satu Hingga Lebih dari Satu *Node* Sensor Sekaligus

Mengacu pada skenario uji coba dari bab 5.2.9, uji coba dilakukan dengan menyalakan Android *gateway* bersamaan dengan 1, 2 dan 3 *node* sensor sekaligus. Pengujian pertama dilakukan dengan menyalakan satu *node* sensor. Hasil uji coba dapat dilihat pada Gambar 5.27. Berdasarkan hasil uji coba tersebut, dapat dipastikan bahwa Android *gateway* dapat terhubung dengan satu *node* sensor. Bukti terhubungnya *node* sensor dapat dilihat pada kotak warna merah, nyala lampu mengindikasikan modul Bluetooth terhubung. Sedangkan kotak warna kuning menunjukkan bahwa Android *gateway* menerima data dari *node* sensor.

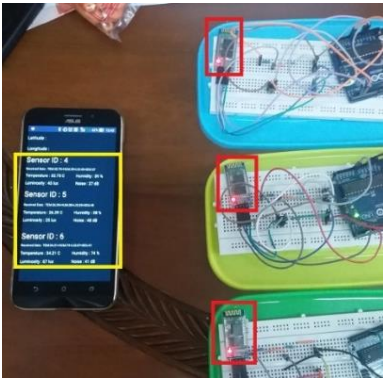
Gambar 5.27 Hasil Uji Coba UJ-F09 mengkoneksikan Android *Gateway* dengan satu *node* sensor

Pengujian kedua dilakukan dengan menyalakan dua *node* sensor sekaligus. Hasil uji coba dapat dilihat pada Gambar 5.28. Berdasarkan hasil uji coba tersebut, dapat dipastikan bahwa Android *gateway* dapat terhubung dengan dua *node gateway*. Bukti terhubungnya *node* sensor dapat dilihat pada kotak warna merah, nyala lampu mengindikasikan modul Bluetooth terhubung. Sedangkan kotak warna kuning menunjukkan bahwa Android *gateway* menerima data dari *node* sensor.



Gambar 5.28 Hasil Uji Coba UJ-F09 mengkoneksikan Android *Gateway* dengan dua *node sensor* sekaligus

Pengujian selanjutnya dilakukan dengan menyalakan tiga *node* sensor sekaligus. Hasil uji coba dapat dilihat pada Gambar 5.29. Berdasarkan hasil uji coba tersebut, dapat dipastikan bahwa Android *gateway* dapat terhubung dengan tiga *node* gateway. Bukti terhubungnya *node* sensor dapat dilihat pada kotak warna merah, nyala lampu mengindikasikan modul Bluetooth terhubung. Sedangkan kotak warna kuning menunjukkan bahwa Android *gateway* menerima data dari *node* sensor.



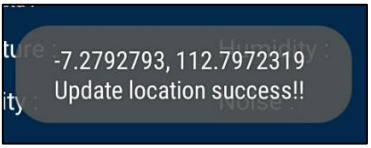
Gambar 5.29 Hasil Uji Coba UJ-F09 mengkoneksikan Android *Gateway* dengan tiga *node* sensor sekaligus

Jika dilihat dari ketiga hasil uji coba di atas, dapat dikatakan bahwa Android *gateway* dapat terhubung dengan *node* sensor. Keterhubungan tersebut dapat dapat terjadi dengan satu, dua maupun tiga *node* sensor sekaligus.

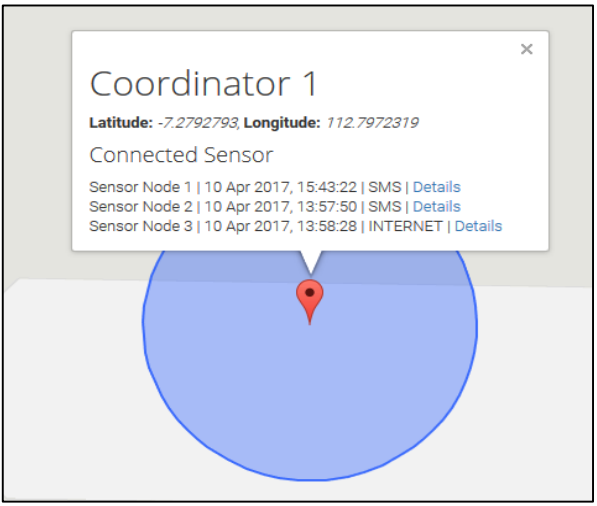
5.3.10 Hasil Uji Coba (UJ-F10) - Menjalankan Android *Gateway* di Beberapa Lokasi Berbeda

Uji coba berikut ini yaitu menjalankan Android *gateway* di beberapa lokasi berbeda. Skenario uji coba dapat dilihat pada bab 5.2.10. Pengujian pertama dilakukan pada lokasi 1, yaitu Laboratorium KBJ Teknik Informatika ITS (Gambar 5.7). Saat uji coba dijalankan, Android *gateway* berhasil mendapatkan nilai

lokasi yang berupa *latitude* dan *longitude*. Nilai lokasi dikirimkan ke server dan status pengiriman dapat dilihat pada Gambar 5.30. Hasil perubahan lokasi 1 ditampilkan dalam website yang ditunjukkan pada Gambar 5.31.



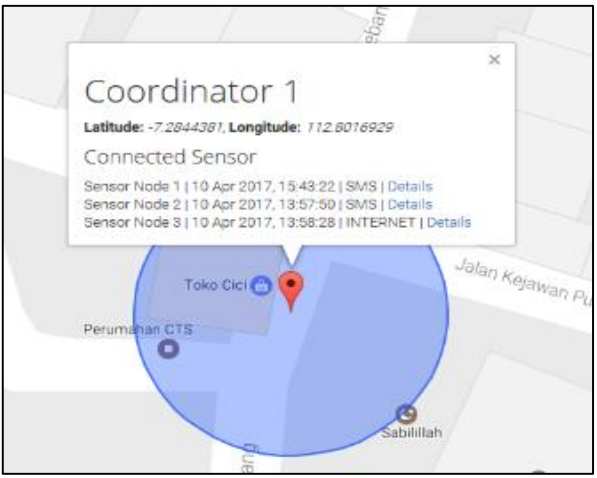
Gambar 5.30 Status pengiriman di lokasi 1



Gambar 5.31 Hasil Uji Coba F10 berupa tampilan peta yang menunjukkan lokasi 1

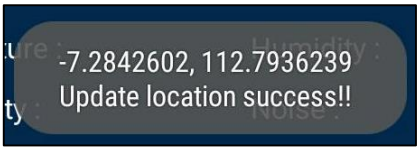
Pengujian kedua dilakukan pada lokasi 2, yaitu tempat tinggal penulis di Kejawan Putih Tambak 22A (Gambar 5.8). Saat uji coba dijalankan, Android *gateway* berhasil mendapatkan nilai lokasi yang berupa *latitude* dan *longitude*. Nilai lokasi dikirimkan ke server dan status pengiriman dapat dilihat pada Gambar 5.32. Hasil perubahan lokasi 2 ditampilkan dalam website yang ditunjukkan pada Gambar 5.33.

Gambar 5.32 Status pengiriman di lokasi 2

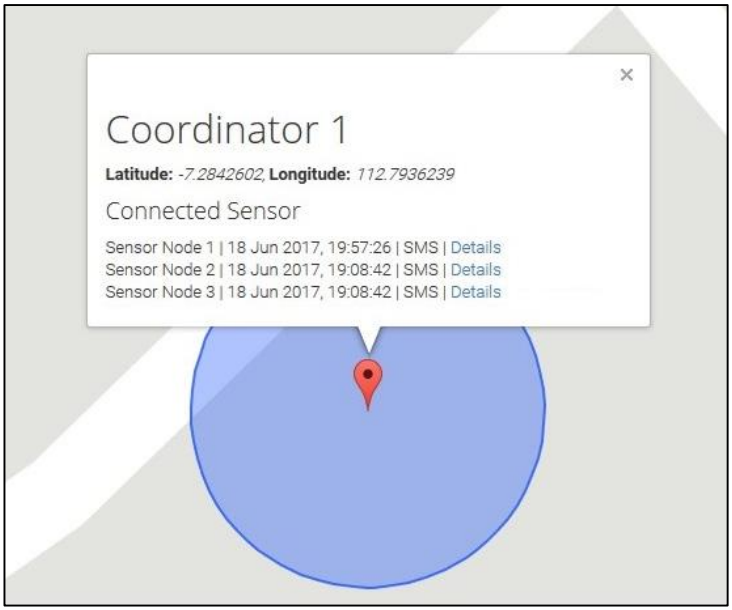


Gambar 5.33 Hasil Uji Coba F10 berupa tampilan peta yang menunjukkan lokasi 2

Pengujian ketiga dilakukan pada lokasi 3, yaitu di depan kantin pusat ITS (Gambar 5.9). Saat uji coba dijalankan, Android *gateway* berhasil mendapatkan nilai lokasi yang berupa *latitude* dan *longitude*. Nilai lokasi dikirimkan ke server dan status pengiriman dapat dilihat pada Gambar 5.34. Hasil perubahan lokasi 3 ditampilkan dalam website yang ditunjukkan pada Gambar 5.35.



Gambar 5.34 Status pengiriman di lokasi 3

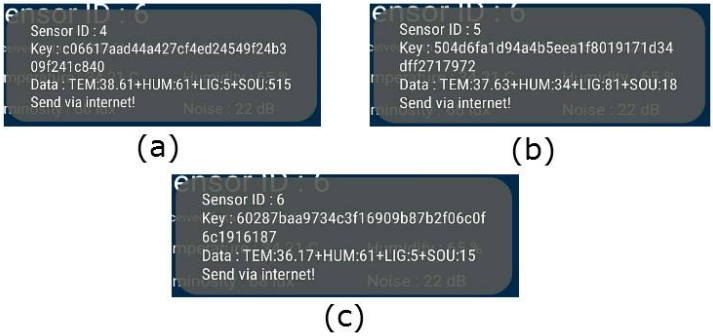


Gambar 5.35 Hasil Uji Coba F10 berupa tampilan peta yang menunjukkan lokasi 3

Jika dilihat dari hasil uji coba di tiga lokasi di atas, dapat dikatakan bahwa fungsionalitas lokasi pada Android *gateway* berjalan dengan baik.

5.3.11 Hasil Uji Coba (UJ-F11) - Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dinyalakan

Berdasarkan uji coba yang telah dilakukan sesuai dengan skenario pada 5.2.11, pengujian diawali dengan menyalakan 3 *node* sensor yang ada. Setelah itu menyalakan Android *gateway* dalam keadaan wifi juga dinyalakan. Saat waktu sepuluh menit terlewati, dilakukan pengamatan apakah dapat melakukan pengiriman data lewat internet atau tidak. Gambar 5.36 menunjukkan status pengiriman data melalui internet.



Gambar 5.36 Status pengiriman data via internet (a) *node* sensor 1, (b) *node* sensor 2, (c) *node* sensor 3

Pengamatan juga dilakukan terhadap basis data yang ada pada server. Berdasarkan pengamatan yang dilakukan, diketahui bahwa data berhasil diterima oleh server. Gambar 5.37 merupakan data yang berhasil diterima server di *database*.

37.63	34	81	18	2017-05-06 14:29:13	INTERNET
38.61	61	5	515	2017-05-06 14:29:15	INTERNET
36.17	61	5	15	2017-05-06 14:29:15	INTERNET

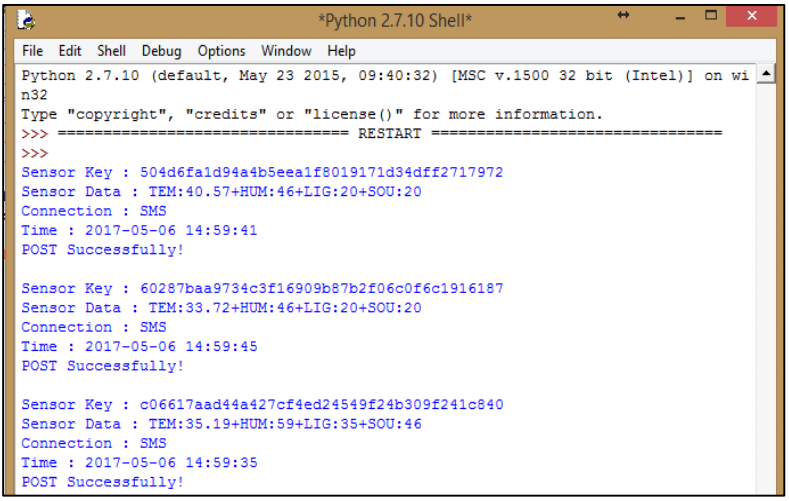
Gambar 5.37 Hasil Uji Coba UJ-F11 yaitu data yang masuk di basis data server

5.3.12 Hasil Uji Coba (UJ-F12) - Menjalankan Android *Gateway* dalam Keadaan Koneksi Internet Dimatikan

Sesuai dengan skenario uji coba pada bab 5.2.12, Android *gateway* akan dijalankan dalam keadaan wifi dan paket data dimatikan. Pada uji coba yang telah dilakukan, setelah Android *gateway* dijalankan dalam waktu sepuluh menit ketiga (tiga puluh menit), kemudian dilakukan pemantauan apakah fungsi pengiriman data lewat sms berjalan dengan baik. Setelah waktu tiga puluh menit terlewati, Android *gateway* berhasil mengirimkan data dalam bentuk sms (Gambar 5.38).

Gambar 5.38 Status pengiriman data via sms (a) *node* sensor 1, (b) *node* sensor 2, (c) *node* sensor 3

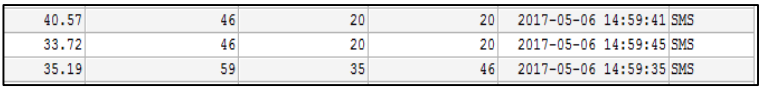
Setelah sms berhasil diterima sms *receiver*, kemudian data tersebut diolah dan dikirimkan server. Status pengiriman data dari sms *receiver* ke server ada pada Gambar 5.39.



Gambar 5.39 Status pengiriman data ke server dari 3 *node* sensor

Kemudian dilakukan pemantauan basis data server. Berdasarkan pemantauan yang telah dilakukan, diketahui bahwa

data masuk ke server. Data yang diterima ditampilkan pada Gambar 5.40.



40.57	46	20	20	2017-05-06 14:59:41	SMS
33.72	46	20	20	2017-05-06 14:59:45	SMS
35.19	59	35	46	2017-05-06 14:59:35	SMS

Gambar 5.40 Hasil Uji Coba UJ-F12 yaitu data yang masuk di basis data server

5.4 Skenario Uji Coba Performa

Uji coba performa merupakan sebuah pengujian yang dilakukan untuk menguji performa kerja dari sistem yang telah dibuat. Berikut di bawah ini pengujian – pengujian yang dilakukan.

5.4.1 Skenario Uji Coba (UJ-P01) - Lama Waktu Bluetooth untuk Terhubung

Uji coba ini dilakukan untuk mendapatkan waktu rata – rata yang diperlukan sebuah komunikasi Bluetooth untuk dapat saling terhubung. Pengujian dilakukan dengan mengkombinasikan koneksi Android *gateway* dengan satu, dua dan tiga *node* sensor sekaligus. Berikut ini Tabel 5.13 menjelaskan skenario uji coba.

Tabel 5.13 Skenario Uji Coba Lama Waktu Bluetooth untuk Terhubung

ID	**UJ-P01**
Nama	Uji Coba Lama Waktu Bluetooth untuk Terhubung
Tujuan Uji Coba	Menguji performa yang berupa waktu rata – rata yang dibutuhkan sebuah komunikasi Bluetooth untuk dapat saling terhubung
Kondisi Awal	Menyalakan Android *gateway* dan *node* sensor
Skenario	1. Menyalakan Android *gateway* dan *node* sensor 1 sebanyak 10 kali percobaan 2. Menyalakan Android *gateway* dan *node* sensor 2 sebanyak 10 kali percobaan 3. Menyalakan Android *gateway* dan *node* sensor 3 sebanyak 10 kali percobaan

	4. Menyalakan Android *gateway* dan *node* sensor 1 & 2 sebanyak 10 kali percobaan 5. Menyalakan Android *gateway* dan *node* sensor 1 & 3 sebanyak 10 kali percobaan 6. Menyalakan Android *gateway* dan *node* sensor 2 & 3 sebanyak 10 kali percobaan 7. Menyalakan Android *gateway* dan *node* sensor 1, 2 & 3 sebanyak 10 kali percobaan 8. Memantau hasil yang telah dicatatkan pada file csv
Masukan	-
Keluaran	Waktu dalam detik
Hasil yang Diharapkan	Mendapatkan waktu rata – rata yang dibutuhkan Bluetooth untuk dapat saling terhubung

5.4.2 Skenario Uji Coba (UJ-P02) - *Delay* Pengiriman Data ke Server

Pengujian pertama ini dilakukan untuk menghitung *delay* waktu rata – rata dalam proses pengiriman data ke server melalui internet. Uji coba akan dilakukan dengan mengirimkan data yang diterima dari *node* sensor melalui internet secara berkala. Pengujian kedua adalah untuk menghitung *delay* waktu rata – rata dalam proses pengiriman data ke server melalui sms. Hanya saja, pengiriman data berikutnya adalah menggunakan sms. Percobaan akan dijalankan sebanyak masing - masing sepuluh kali. Tabel 5.14 merupakan skenario uji coba secara lebih rinci.

Tabel 5.14 Skenario Uji Coba *Delay* Pengiriman Data ke Server

ID	**UJ-P02**
Nama	Uji Coba Delay Pengiriman Data ke Server
Tujuan Uji Coba	Menguji performa yang berupa *delay* rata – rata yang dibutuhkan dalam proses pengiriman data ke server melalui internet dan sms
Kondisi Awal	Mengirimkan data dari *node* sensor (setiap lima detik) dan Android *gateway* mengirimkan ke

	server (10 data melalui internet dan 10 data berikutnya melalui sms)
Skenario	1. Menjalankan *node* sensor dan Android *gateway* 2. Menunggu *node* sensor mengirimkan sepuluh data dan dikirimakn ke server melalui internet 3. Memantau hasil pada server 4. Menghitung *delay* waktu rata – rata 5. Menjalankan *node* sensor dan Android *gateway* 6. Menunggu *node* sensor mengirimkan sepuluh data sensor dalam bentuk sms 7. Memantau hasil pada server 8. Menghitung *delay* waktu rata – rata 9. Menghitung *delay* waktu rata-rata pengiriman data menggunakan Bluetooth 10. Menambahkan *delay* pengiriman data menggunakan Bluetooth ke rata-rata pengiriman melalui internet dan sms. Untuk mendapatkan *delay realtime* data.
Masukan	Data sensor
Keluaran	Delay waktu rata - rata
Hasil yang Diharapkan	Mendapatkan perbandingan *delay* waktu rata – rata pengiriman data ke server melalui internet dan sms

5.4.3 Skenario Uji Coba (UJ-P03) - Akurasi Pengiriman Data Bluetooth

Pengujian berikut ini dilakukan untuk mendapatkan akurasi dari pengiriman data menggunakan komunikasi Bluetooth. Uji coba dilakukan dengan menjalankan Android *gateway* dan tiga *node* sensor yang ada dan dilakukan pada jarak yang berbeda – beda (≤1 meter, ±5 meter dan ±10 meter). Percobaan akan diulangi sebanyak sepuluh kali. Berikut ini menjelaskan skenario uji cobanya.

Tabel 5.15 Skenario Uji Coba Akurasi Pengiriman Data Bluetooth

ID	**UJ-P03**
Nama	Uji Coba Akurasi Pengiriman Data Bluetooth

Tujuan Uji Coba	Menguji performa yang berupa akurasi pengiriman data menggunakan Bluetooth dari jarak yang berda - beda
Kondisi Awal	Menyalakan Android *gateway* dan tiga *node* sensor yang ada di jarak berbeda (≤1 meter, ±5 meter dan ±10 meter)
Skenario	1. Menyalakan Android *gateway* dan tiga *node* sensor yang ada selama kurang lebih 1 hingga 2 menit pada jarak ≤1 meter 2. Mengulang pengujian hingga 10 kali 3. Memantau hasil pada sebuah file csv 4. Mengulangi pengujian pada jarak ±5 meter 5. Mengulangi pengujian pada jarak ±10 meter
Masukan	Data sensor
Keluaran	Akurasi pengiriman data pada jarak berbeda - beda
Hasil yang Diharapkan	Mendapatkan akurasi pengiriman data menggunakan Bluetooth dari jarak berbeda - beda dan membandingkan hasilnya

5.4.4 Skenario Uji Coba (UJ-P04) – Penggunaan Daya pada *Node* Sensor

Uji coba penggunaan daya ditujukan untuk mengetahui besaran daya yang digunakan *node* sensor setiap jamnya. Skenario lebih rinci untuk uji coba ada pada Tabel 5.16.

Tabel 5.16 Skenario Uji Coba Penggunaan Daya pada *Node* Sensor

ID	**UJ-P04**
Nama	Uji Coba Penggunaan Daya pada *Node* Sensor
Tujuan Uji Coba	Menguji performa yang berupa penggunaan daya *node* sensor dalam per jam
Kondisi Awal	Mencatat daya baterai sebelum digunakan (saat masih 100%)
Skenario	1. Mencatat daya baterai sebelum digunakan (saat masih 100%) 2. Menyalakan Android *gateway* dan satu *node* sensor (dengan menggunakan baterai)

	3. Membiarkan *node* sensor menyala selama 5 menit dengan jeda waktu pengiriman data 5 detik 4. Mencatat daya baterai sesudah digunakan 5. Mengisi ulang baterai hingga daya 100% 6. Mengulangi langkah 1 – 5 sebanyak 10 kali percobaan 7. Mengulangi langkah 1 – 6 namun dengan mengganti jeda pengiriman data 10 detik
Masukan	Daya sebelum
Keluaran	Daya sesudah
Hasil yang Diharapkan	Mendapatkan rata – rata daya yang digunakan *node* sensor per jamnya

5.4.5 Skenario Uji Coba (UJ-P05) – Penggunaan Daya pada Android *Gateway*

Pada uji coba ini dilakukan pengujian untuk mengetahui penggunaan daya pada Android *gateway*. Penggunaan daya yang diuji dibagi menjadi dua yaitu penggunaan daya jika mengirim data melalui internet dan mengirim data melalui sms. Berikut ini Tabel 5.17 merinci skenario uji cobanya.

Tabel 5.17 Skenario Uji Coba Penggunaan Daya pada Android *Gateway*

ID	**UJ-P05**
Nama	Uji Coba Penggunaan Daya pada Android *Gateway*
Tujuan Uji Coba	Menguji performa yang berupa penggunaan daya Android *gateway* jika data dikirimkan melalui internet dan jika data dikirim melalui sms
Kondisi Awal	Mencatat daya baterai sebelum digunakan (saat masih 100%)
Skenario	1. Mencatat daya baterai sebelum digunakan (saat masih 100%) 2. Menyalakan Android *gateway* dan mengatur pengiriman data ke server melalui internet setiap 3 menit

	3. Menyalakan Android *gateway* selama 3 jam 4. Mencatat daya baterai sesudah digunakan 5. Mengisi ulang baterai hingga daya 100% 6. Mencatat daya baterai sebelum digunakan (saat masih 100%) 7. Menyalakan Android *gateway* dan mengatur pengiriman data ke server melalui sms setiap 3 menit 8. Menyalakan Android *gateway* selama 3 jam 9. Mencatat daya baterai sesudah digunakan 10. Membandingkan penggunaan daya antara dua uji coba yang telah dilakukan
Masukan	Daya sebelum
Keluaran	Daya sesudah dan hasil perbandingan
Hasil yang Diharapkan	Mendapatkan rata – rata daya yang digunakan dan perbandingannya

5.5 Hasil Uji Coba Performa

Pada bagian sebelumnya telah dijelaskan mengenai skenario dari uji coba performa. Berikut ini akan dijabarkan mengenai hasil dari uji coba performa yang telah dijalankan.

5.5.1 Hasil Uji Coba (UJ-P01) - Lama Waktu Bluetooth untuk Terhubung

Mengacu pada skenario uji coba di bab 5.4.1, telah dilakukan pengujian sesuai dengan skenario yang ada. Pertama – tama uji coba dilakukan dengan mengkoneksikan Android *gateway* dengan satu per satu dari tiga *node* sensor sebanyak 10 kali.

Hasil uji coba terdapat pada Tabel 5.18 berikut ini dengan rincian pada Tabel 7.1 yang ada di lampiran. Pada uji coba koneksi Bluetooth dengan satu *node* sensor, didapatkan lama waktu rata – rata yang dibutuhkan adalah 2 detik 256 milidetik.

Tabel 5.18 Hasil Uji Coba UJ-P01 pada Satu *Node* Sensor

Satu *Node*	*Node* Sensor		
	1	2	3
Rata - rata	*1,419*	*3,681*	*1,668*
Rata - rata Keseluruhan	**2,256 detik**		

Selanjutnya merupakan uji coba yang dilakukan dengan menggunakan dua *node* sensor sekaligus. Uji coba kali ini melibatkan kombinasi *node* sensor 1 dengan 2, 1 dengan 3 dan 2 dengan 3.

Hasil uji coba terdapat pada Tabel 5.19 di bawah ini dan rincian terdapat pada lampiran di Tabel 7.2. Pada uji coba koneksi Bluetooth dengan dua *node* sensor sekaligus, didapatkan lama waktu rata – rata yang dibutuhkan adalah 3 detik 328 milidetik.

Tabel 5.19 Hasil Uji Coba UJ-P01 pada Dua *Node* Sensor Sekaligus

Dua *Node* Sekaligus	*Node* Sensor		
	1 dan 2	1 dan 3	2 dan 3
Rata - rata	*4,009*	*2,259*	*3,717*
Rata - rata Keseluruhan	**3,328 detik**		

Berikutnya adalah uji coba yang dilakukan dengan menggunakan tiga *node* sensor sekaligus. Pengujian ini dilakukan dengan menjalankan tiga *node* sensor yang ada.

Hasil uji coba terdapat pada Tabel 5.20 di bawah ini dan dirinci pada Tabel 7.3. Pada uji coba koneksi Bluetooth dengan tiga *node* sensor sekaligus, didapatkan lama waktu rata – rata yang dibutuhkan adalah 3 detik 634 milidetik.

Tabel 5.20 Hasil Uji Coba UJ-P01 pada Tiga *Node* Sensor Sekaligus

Tiga *Node* Sekaligus	*Node* Sensor		
	1, 2 dan 3		
	1	2	3
Rata - rata	1,731	4,936	4,236
Rata - rata Keseluruhan	**3,634 detik**		

Berdasarkan hasil uji coba pada tiga tabel di atas, berikut ini merupakan hasil akhir uji coba. Pada Tabel 5.21 didapatkan nilai rata – rata akhir yang dibutuhkan Bluetooth untuk dapat saling terhubung yaitu **3 detik 72 milidetik**.

Tabel 5.21 Hasil Akhir Uji Coba UJ-P01

Jumlah *Node* Terkoneksi	Rata-rata	**Rata - rata Akhir**
1	2.256	**3 detik 72 milidetik**
2	3.328	
3	3.634	

5.5.2 Hasil Uji Coba (UJ-P02) - *Delay* Pengiriman Data ke Server

Sesuai dengan skenario uji coba pada bab 5.4.2, dilakukan uji coba dengan cara mengirimkan data melalui internet. Pengujian dilakukan sebanyak sepuluh kali. Berikut ini hasil uji coba pada Tabel 5.22 yang dirinci pada Tabel 7.4 di bagian lampiran.

Tabel 5.22 Hasil Uji Coba UJ-P02 *Delay* Pengiriman Data

***Delay* Pengiriman Data**		
Internet	**SMS**	**Bluetooth**
484,4 milidetik	7,702 detik	36,7 milidetik

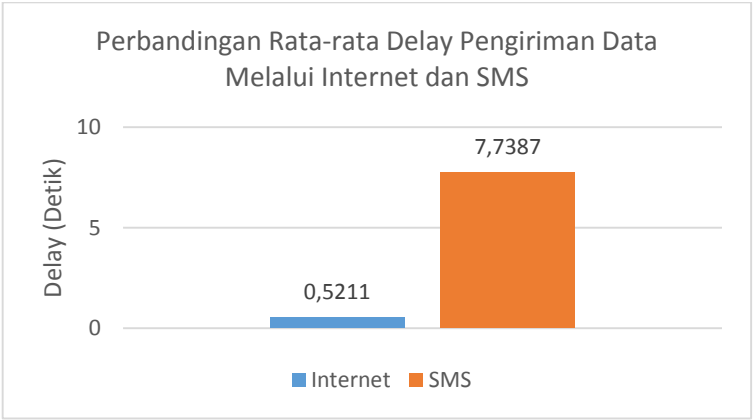
Berdasarkan hasil uji coba yang ada pada tabel di atas, didapatkan *delay* pengiriman data melalui internet adalah 484,4 milidetik.

Pengujian kedua dilakukan dengan mengirimkan sms sesuai *format* sebanyak sepuluh kali. Hasil uji coba pada Tabel 5.22 dan dirinci pada Tabel 7.5 di lampiran.

Menurut hasil uji coba yang ada pada tabel di atas, didapatkan *delay* pengiriman data melalui sms adalah 7 detik 702 milidetik.

Dikarenakan tidak adanya fungsi jam pada Arduino, untuk menguji delay *realtime* dari pengiriman data ke server dilakukan uji coba untuk menghitung *delay* pengiriman data Bluetooth. *Delay* ini akan ditambahkan ke nilai rata – rata dari pengiriman data internet dan sms. Berikut ini Tabel 5.22 yang menjabarkan *delay* dari pengiriman data Bluetooth yang juga dirinci pada Tabel 7.6.

Setelah didapatkan rata – rata delay pengiriman data Bluetooth yaitu sebesar 36,7 milidetik, rata – rata tersebut ditambahkan dengan nilai rata – rata dar pengiriman data internet dan sms untuk mendapatkan *delay realtime.*



Gambar 5.41 Grafik Perbandingan *Delay* (*Realtime*) Pengiriman Data Melalui Internet dan SMS

Berdasarkan grafik pada gambar Gambar 5.41, *delay* pengiriman data internet *realtime* adalah 0,5211 detik dan *delay* pengiriman data sms adalah 7,7387 detik. Sehingga diketahui bahwa *delay* pengiriman data melalui sms lebih lambat 7 detik 218 milidetik dibandingkan dengan pengiriman data melalui internet.

5.5.3 Hasil Uji Coba (UJ-P03) - Akurasi Pengiriman Data Bluetooth

Mengacu pada skenario uji coba di bab 5.4.3, dilakukan pengujian untuk menghitung akurasi pengiriman data menggunakan komunikasi Bluetooth. Uji coba dilakukan dengan menjalankan Android *gateway* dan tiga *node* sensor, kemudian memantau pengiriman data sensor. Setiap data yang diterima Android *gateway*, dicatat dalam sebuah file csv. Pengujian dilakukan sebanyak 10 kali dengan rentang waktu setiap pengujian adalah 1 sampai 2 menit. Berikut ini hasil uji coba yang diambil dengan jarak ≤1 meter dijelaskan pada Tabel 5.23 yang dirinci pada Tabel 7.7 di lampiran.

Tabel 5.23 Hasil Uji Coba UJ-P03 Akurasi Pengiriman Bluetooth

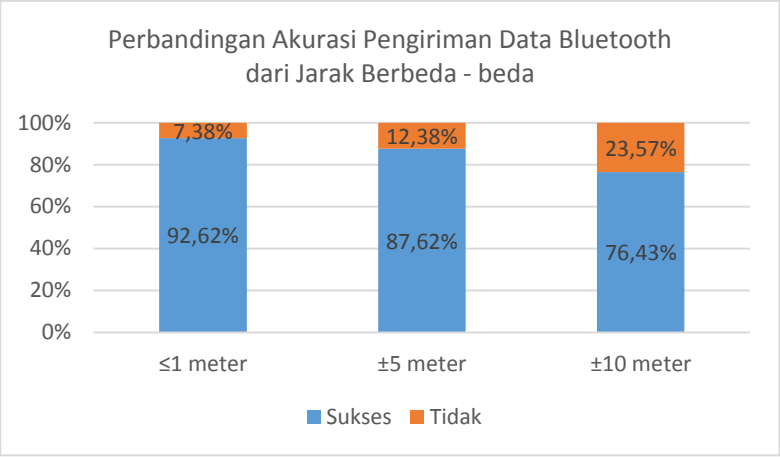
Akurasi Pengiriman Bluetooth		
≤1 meter	**±5 meter**	**±10 meter**
92,62%	87,62%	76,43%

Berdasarkan hasil uji coba akurasi pada jarak **≤1 meter**, didapatkan akurasi pengiriman data menggunakan Bluetooth sebesar **92,62%**. Tabel 5.23 menjabarkan hasil uji coba dari jarak ±5 meter yang dijabarkan lebih rinci pada Tabel 7.8 di lampiran.

Berdasarkan hasil uji coba akurasi pada jarak **±5 meter**, didapatkan akurasi pengiriman data menggunakan Bluetooth sebesar **87,62%**. Tabel 5.23 menjabarkan hasil uji coba dari jarak ±10 meter yang dirinci pada lampiran di Tabel 7.9.

Berdasarkan hasil uji coba akurasi pada jarak **±10 meter**, didapatkan akurasi pengiriman data menggunakan Bluetooth

sebesar **76,43%**. Setelah didapatkan hasil dari tiga uji coba di atas, dilakukan perbandingan seperti grafik pada Gambar 5.42.



Gambar 5.42 Grafik Perbandingan Akurasi Pengiriman Data Bluetooth

Berdasarkan grafik di atas, diketahui bahwa semakin jauh jarak komunikasi maka akurasi pengiriman data Bluetooth akan semakin menurun. Pengujian tidak dilanjutkan pada jarak yang melebihi 10 meter dikarenakan mengalami kendala koneksi yang sering terputus.

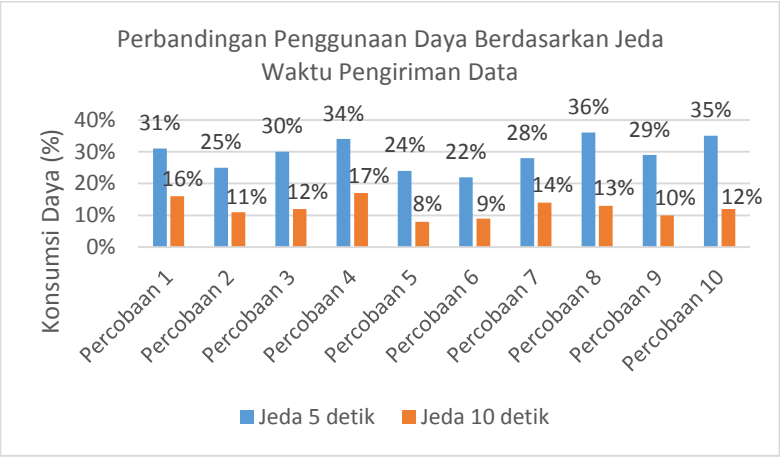
5.5.4 Hasil Uji Coba (UJ-P04) - Penggunaan Daya pada *Node* Sensor

Pada uji coba (sesuai skenario bab 5.4.4) ini digunakan sebuah baterai bertegangan 9V berdaya 220 mAh. Pengujian pertama dilakukan dengan menjalankan sebuah *node* sensor selama 5 menit dengan jeda waktu pengiriman data adalah 5 detik. Uji coba diulangi hingga sepuluh kali. Pada pengujian kedua dilakukan juga sebanyak sepuluh kali namun dengan jeda pengiriman data adalah 10 detik. Berikut ini Tabel 5.24 yang menampilkan hasil uji coba dan dijabarkan pada Tabel 7.10.

Tabel 5.24 Hasil Uji Coba UJ-P04 Penggunaan Daya *Node* Sensor

Daya Baterai 220 mAh	Prosentase Penggunaan Daya (%)	
	5 detik	10 detik
Rata - rata Prosentase (%)	29,4%	12,2%

Berdasarkan pada hasil uji coba yang, dilakukan pengukuran sisa daya menggunakan Avometer. Didapatkan rata – rata penggunaan daya *node* sensor dengan jeda pengiriman data 5 detik yaitu sebesar 64,68 mAh / 5 menit atau setara dengan 776,16 mAh / jam. Sedangkan untuk pengiriman data dengan jeda 10 detik dibutuhkan daya sebesar 26,84 mAh / 5 menit atau 322,08 mAh / jam.



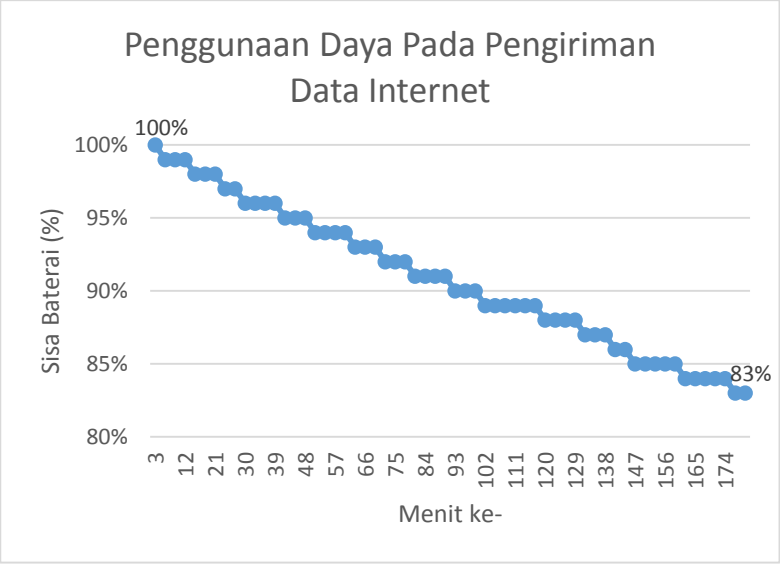
Gambar 5.43 Grafik Perbandingan Penggunaan Daya Berdasarkan Jeda Waktu Pengiriman Data

Perbandingan antara jeda waktu pengiriman data dapat dilihat pada grafik di Gambar 5.43. Berdasarkan grafik di atas, diketahui bahwa semakin besar jeda waktu pengiriman data maka penggunaan daya akan semakin kecil.

5.5.5 Hasil Uji Coba (UJ-P05) - Penggunaan Daya pada Android *Gateway*

Sesuai skenario pada bab 5.4.5, ada dua hal yang diujikan dalam uji coba satu ini. Pertama adalah menguji penggunaan daya pada proses pengiriman data lewat internet. Kedua adalah menguji penggunaan daya pada proses pengiriman data lewat sms. Uji coba dilakukan menggunakan sebuah *smartphone* berdaya 5000 mAh.

Uji coba pertama dilakukan dengan mengirimkan data dari Android *gateway* ke server melalui internet. Data dikirimkan setiap 3 menit. Lama waktu pengujian adalah 3 jam. Berikut ini hasil uji coba pertama pada Gambar 5.44.

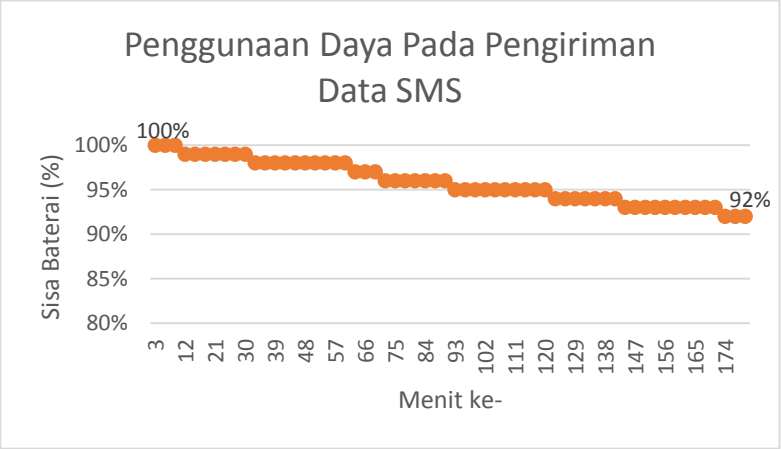


Gambar 5.44 Grafik Hasil Uji Coba UJ-P05 – Penggunaan Daya pada Pengiriman Data Melalui Internet

Berdasarkan hasil uji coba yang telah dilakukan selama 3 jam, baterai *level* yang sebelumnya 100% tersisa menjadi 83%. Artinya daya yang digunakan untuk mengirimkan data melalui

internet dalam waktu 3 jam adalah sebesar **17%** atau sebesar **850 mAh**.

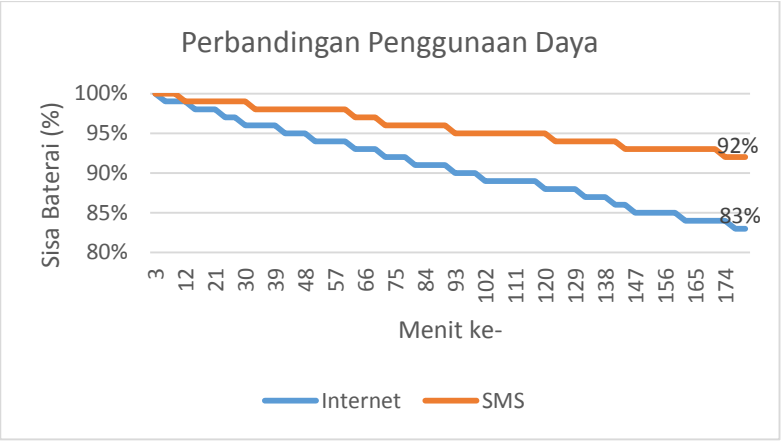
Pada uji coba yang kedua dilakukan dengan mengirimkan data dari Android *gateway* melalui sms. Data dikirimkan setiap 3 menit. Lama waktu pengujian adalah 3 jam. Berikut ini hasil uji coba kedua pada Gambar 5.45.



Gambar 5.45 Grafik Hasil Uji Coba UJ-P05 – Penggunaan Daya pada Pengiriman Data Melalui SMS

Berdasarkan hasil uji coba yang telah dilakukan selama 3 jam, baterai *level* yang sebelumnya 100% tersisa menjadi 92%. Artinya daya yang digunakan untuk mengirimkan data melalui sms dalam waktu 3 jam adalah sebesar **8%** atau sebesar **400 mAh**.

Setelah dilakukan dua uji coba sebelumnya, Gambar 5.46 menunjukkan perbandingan dari penggunaan daya internet dan sms. Perbandingan antara penggunaan daya pengiriman data melalui internet dan sms dalam waktu 3 jam adalah sebesar 9% atau 450 mAh. Sehingga dalam waktu **1 jam** pengiriman data melalui sms dapat menghemat **3%** atau **150 mAh** dibandingkan pengiriman data melalui internet.



Gambar 5.46 Grafik Hasil Perbandingan Penggunaan Daya Antara Pengiriman Data Internet dan SMS

5.6 Evaluasi Hasil Uji Coba

Pada bagian sebelumnya, telah dilakukan uji coba terhadap sistem yang dibuat. Uji coba yang telah dilakukan berkenaan dengan uji coba fungsionalitas dan uji coba performa. Berikut ini Tabel 5.25 merangkum evaluasi hasil uji coba fungsionalitas dan Tabel 5.26 merangkum hasil uji coba performa yang telah dilakukan.

Tabel 5.25 Evaluasi Hasil Uji Coba Fungsionalitas

No	Kode Uji Coba	Evaluasi
1.	UJ-F01	Dapat menampilkan sensor berdasarkan kelompok sensor
2.	UJ-F02	Dapat mendaftarkan kelompok baru
3.	UJ-F03	Dapat mendaftarkan sensor baru
4.	UJ-F04	Dapat menampilkan lokasi keseluruhan kelompok sensor yang ada
5.	UJ-F05	Dapat menampilkan lokasi spesifik kelompok sensor tertentu
6.	UJ-F06	Dapat menampilkan detail sensor

No	Kode Uji Coba	Evaluasi
7.	UJ-F07	Dapat menerima sms masuk dan mengirim kembali hasil olahnya ke server
8.	UJ-F08	Dapat nilai sensor dan mengirim data ke Android *gateway*
9.	UJ-F09	Android *gateway* dapat terhubung dengan satu, dua dan tiga sekaligus *node* sensor menggunakan Bluetooth
10.	UJ-F10	Android *gateway* dapat mengirimkan data lokasi saat ini
11.	UJ-F11	Android *gateway* dapat berperan adaptif saat koneksi internet tersedia dengan mengirimkan data server melalui internet
12.	UJ-F12	Android *gateway* dapat berperan adaptif saat koneksi internet tidak tersedia dengan mengirimkan data server melalui sms

Tabel 5.26 Evaluasi Hasil Uji Coba Performa

No	Kode Uji Coba	Evaluasi
1.	UJ-P01	Waktu rata – rata yang dibutuhkan Bluetooth untuk dapat saling terhubung yaitu 3 detik 72 milidetik
2.	UJ-P02	*Delay* (*realtime*) pengiriman data melalui sms lebih lambat 7 detik 218 milidetik dibandingkan dengan pengiriman data melalui internet
3.	UJ-P03	Akurasi pengiriman data Bluetooth berurut – turut adalah 92,62% (≤1 m), 87,62% (±5 m) dan 76,43% (±10 m). Sehingga dapat dikatakan bahwa akurasi Bluetooth menurun seiring dengan jarak yang semakin meningkat.
4.	UJ-P04	Daya yang dibutuhkan *node* sensor dengan jeda waktu pengiriman data 5 detik adalah 776,16 mAh / jam sedangkan dengan jeda waktu 10 detik adalah 322,08 mAh / jam.

No	Kode Uji Coba	Evaluasi
5.	UJ-P05	Penggunaan daya Android *gateway* jika data dikirim melalui sms (dalam rentang 3 jam) lebih hemat 9% (450 mAh) dibandingkan jika data dikirim melalui internet. Artinya pengiriman data menggunakan sms dapat menghemat daya sebesar 3% (150 mAh) setiap jamnya.

BAB VI
KESIMPULAN DAN SARAN

Bagian terakhir yang dibahas dalam buku ini adalah kesimpulan dan saran. Kesimpulan yang dijabarkan di bawah ini didapatkan berdasarkan dari hasil pengamatan uji coba sistem. Selain kesimpulan, dijabarkan pula saran yang diharapkan dapat berguna untuk pengembangan sistem dan penelitian selanjutnya.

6.1 Kesimpulan

Kesimpulan yang didapatkan berdasarkan hasil uji coba fungsional dan performa adalah sebagai berikut :

1. *Sensor Cluster Management* dapat memberikan layanan virtualisasi sensor yang bisa melakukan pengelompokkan (*cluster*) dan pengelolaan sensor – sensor tersebar.
2. Android *gateway* dapat berperan adaptif dalam menentukan cara pengiriman data ke server berdasarkan kondisi ketersediaan koneksi internet.
3. Tiga *node* sensor dapat terhubung sekaligus dengan Android *gateway* menggunakan Bluetooth sehingga pengiriman data dapat dilakukan secara bersamaan.
4. Performa dari sistem yang telah dibangun adalah sebagai berikut :
    a. Rata – rata waktu yang dibutuhkan sebuah koneksi Bluetooth untuk terhubung adalah **3 detik 72 milidetik**.
    b. *Delay* pengiriman data melalui sms lebih lambat **7 detik 218 milidetik** dibandingkan dengan pengiriman data melalui internet
    c. Akurasi pengiriman data Bluetooth berurut – turut adalah **92,62%** (≤1 m), **87,62%** (±5 m) dan **76,43%** (±10 m). Sehingga dapat dikatakan bahwa

akurasi Bluetooth menurun seiring dengan jarak yang semakin meningkat.

d. Daya yang dibutuhkan *node* sensor dengan jeda waktu pengiriman data 5 detik adalah **776,16 mAh / jam** sedangkan dengan jeda waktu 10 detik adalah **322,08 mAh / jam**. Sehingga dapat disimpulkan bahwa semakin besar jeda waktu pengiriman data maka penggunaan daya yang dibutuhkan akan semakin kecil.
e. Penggunaan daya Android *gateway* pada data menggunakan sms dapat menghemat daya sebesar **3%** (**150 mAh**) setiap jamnya.

6.2 Saran

Saran yang diberikan untuk pengembangan penelitian kedepannya adalah sebagai berikut :

1. Mengembangkan perancangan sistem yang tidak terbatas hanya pada sebuah Piconet, namun juga terdiri dari lebih dari satu Piconet yang saling terintegrasi (Scatternet).
2. Mengembangkan kemampuan sistem untuk menanggulangi keadaan saat kondisi koneksi internet tidak tersedia dan pulsa tidak mencukupi untuk mengirim sms.

DAFTAR PUSTAKA

[1] I. P. Pratama dan S. Suakanto, *Wireless Sensor Network – Teori & Praktek Berbasiskan Open Source*. Bandung: Informatika, 2015.

[2] WG802.15 - Wireless Personal Area Network (WPAN) Working Group, "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.1a: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPAN)," *IEEE Std 802151-2005 Revis. IEEE Std 802151-2002*, hal. 1–700, Jun 2005.

[3] H. B. Siregar, *Rancang Bangun Sistem Virtualisasi Sensor untuk Manajemen Sensor Tersebar Berbasis Komputasi Awan*. Surabaya, 2016.

[4] M. Yuriyama dan T. Kushida, "Sensor-Cloud Infrastructure - Physical Sensor Management with Virtualized Sensors on Cloud Computing," in *2010 13th International Conference on Network-Based Information Systems*, 2010, hal. 1–8.

[5] I. Bisio, A. Sciarrone, dan S. Zappatore, "Asset tracking architecture with Bluetooth Low Energy tags and ad hoc smartphone applications," in *2015 European Conference on Networks and Communications (EuCNC)*, 2015, hal. 460–464.

[6] Anonim, "NS2 Simulation Code for Wireless Sensor Network," *NS2 Projects*. [Daring]. Tersedia pada: https://ns2projects.org/ns2-simulation-code-for-wireless-sensor-network/. [Diakses: 08-Apr-2017].

[7] Memen, "Visualisasi Data (Jenis Visualisasi)," *Cerita Kita*, 13-Feb-2014. [Daring]. Tersedia pada: http://memen.komputer.pcr.ac.id/2014/02/12/visualisasi-data/. [Diakses: 08-Apr-2017].

[8] R. Manoharan, S. Rajarajan, S. Sashtinathan, dan K. Sriram, "A Novel Multi-hop B3G Architecture for Adaptive Gateway

Management in Heterogeneous Wireless Networks," dipresentasikan pada IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2009.

[9] G. Găşpăresc, "Development of a low-cost system for temperature monitoring," in *2013 36th International Conference on Telecommunications and Signal Processing (TSP)*, 2013, hal. 340–343.

[10] Anonim, "Arduino Store - Community and Electronics." [Daring]. Tersedia pada: https://store.arduino.cc/product/A000066. [Diakses: 06-Apr-2017].

[11] Anonim, "Mengenal dan Belajar Arduino Uno R3," *Ecadio*. [Daring]. Tersedia pada: http://ecadio.com/mengenal-dan-belajar-arduino-uno-r3. [Diakses: 06-Apr-2017].

[12] Anonim, "Bluetooth Technology," *Bluetooth Applications*, 2001. [Daring]. Tersedia pada: http://www.mobileinfo.com/Bluetooth/applic.htm. [Diakses: 07-Apr-2017].

[13] Anonim, "How It Works | Bluetooth Technology Website," *Bluetooth Technology Website*. [Daring]. Tersedia pada: https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works. [Diakses: 07-Apr-2017].

[14] J. Kahn, "Bluetooth 4.1 will bring more reliable, automatic connections, multiple roles, & more via software update," *9to5Mac*, 05-Des-2013. [Daring]. Tersedia pada: https://9to5mac.com/2013/12/05/bluetooth-4-1-will-bring-more-reliable-automatic-connections-multiple-roles-more-via-software-update/. [Diakses: 07-Apr-2017].

[15] D. Janssen dan C. Janssen, "What is Piconet? - Definition from Techopedia," *Techopedia.com*. [Daring]. Tersedia pada: https://www.techopedia.com/definition/5081/piconet. [Diakses: 19-Apr-2017].

[16] T. M. Swaminatha dan C. R. Elden, *Wireless Security and Privacy: Best Practices and Design Techniques*. Boston: Pearson Education, Inc., 2002.

[17] E. Davtyan, "Android Utility Class for Checking Device's Network Connectivity and Speed," *GistHubGist*, 11-Mar-2013. [Daring]. Tersedia pada: https://gist.github.com/emil2k/5130324. [Diakses: 21-Mei-2017].

[18] A. Petrescu, R. Rong, N. Saggar, S. T. Dane, dan D. Weaver, *Network Connection Class*. Facebook, 2017.

[19] Anonim, "Arduino and HC-06 Bluetooth Example," *Arduino Learning*, 24-Mei-2016. [Daring]. Tersedia pada: http://arduinolearning.com/learning/arduino-and-hc-06-bluetooth-example.php. [Diakses: 21-Mei-2017].

[20] M. Currey, "Arduino and HC-06 (ZS-040)," *Martyn Currey Mostly Arduino Stuff*, 08-Okt-2014. [Daring]. Tersedia pada: http://www.martyncurrey.com/arduino-and-hc-06-zs-040/. [Diakses: 07-Apr-2017].

[21] Anonim, "LM35 Temperature Sensor," *Geek Studio*, 2017. [Daring]. Tersedia pada: https://www.geeker.co.nz/sensors/temperature/lm35-temperature-sensor.html. [Diakses: 07-Apr-2017].

[22] Belajar Elektronika, "Bentuk dan Karakteristik Sensor Suhu LM35," *E-Belajar Elektronika*. [Daring]. Tersedia pada: http://e-belajarelektronika.com/. [Diakses: 07-Apr-2017].

[23] T. King, "LM35 Temperature Sensor," *ArduinoInfo*, 08-Jun-2014. [Daring]. Tersedia pada: https://arduino-info.wikispaces.com/LM35. [Diakses: 06-Apr-2017].

[24] Anonim, "Arduino Playground - LM35HigherResolution." [Daring]. Tersedia pada: http://playground.arduino.cc/Main/LM35HigherResolution. [Diakses: 06-Apr-2017].

[25] T. King, "DHT11 Humidity Temp Sensor," *ArduinoInfo*, 31-Agu-2016. [Daring]. Tersedia pada: https://arduino-

info.wikispaces.com/DHT11-Humidity-TempSensor. [Diakses: 04-Jul-2017].

[26] Hek, "Light Level Sensor - LM393," *MySensors*, Sep-2016. [Daring]. Tersedia pada: https://www.mysensors.org/build/light-lm393. [Diakses: 04-Jul-2017].

[27] M. L. Vick, "Microphone Sensor (KY-038)," *Summerfuel Robotics*. [Daring]. Tersedia pada: https://sites.google.com/site/summerfuelrobots/arduino-sensor-tutorials/microphones-with-arduino. [Diakses: 04-Jul-2017].

[28] Prambudi, "Pengertian Android dan Kegunaan Lengkap Versinya," *Bosandroid*, 26-Jan-2016. [Daring]. Tersedia pada: http://www.bosandroid.net/pengertian-android-dan-kegunaan-lengkap-versinya/. [Diakses: 07-Apr-2017].

[29] Susanto, "Pengertian dan Manfaat CodeIgniter," *WebDev*, 18-Nov-2013. [Daring]. Tersedia pada: http://webdev.blogekstra.com/susanto/pengertian-dan-manfaat-codeigniter.html. [Diakses: 07-Apr-2017].

[30] Oracle Corporation, "About MySQL," *MySQL Official Website*, 2017. [Daring]. Tersedia pada: https://www.mysql.com/about/. [Diakses: 07-Apr-2017].

[31] A. Farurroji, "Panduan Singkat Framework Bootstrap," *AFAHRURROJInet*, 23-Apr-2015. [Daring]. Tersedia pada: http://afahrurroji.net/panduan-singkat-framework-bootstrap/. [Diakses: 07-Apr-2017].

[32] Admin, "SMS Gateway - send and receive multiple texts," *Talk Internet Ltd*, 2017. [Daring]. Tersedia pada: https://www.talkinternet.co.uk/products/telephony-and-voip/sms/. [Diakses: 13-Apr-2017].

[33] Arief, "SMS Gateway," *Informatika*, 15-Des-2012. [Daring]. Tersedia pada: http://informatika.web.id/sms-gateway.htm. [Diakses: 08-Apr-2017].

[34] M. Čihař, "Gammu," *Gammu and Wammu Official Website*, 2017. [Daring]. Tersedia pada: https://wammu.eu/gammu/. [Diakses: 07-Apr-2017].

[35] A. Downey, *Think Python - How to Think Like a Computer Scientist*. Needham, Massachusetts: Green Tea Press, 2008.

[36] M. Lutz, *Learning Python - Powerful Object Oriented Programming*. Sebastopol, CA: O'Reilly Media, Inc, 2009.

[37] T. Hønsi, "Highcharts," *Highcharts Official Website*, 2017. [Daring]. Tersedia pada: http://www.highcharts.com/products/highcharts. [Diakses: 07-Apr-2017].

[38] Google Developers, "Transmitting Network Data Using Volley," *Android Developers*. [Daring]. Tersedia pada: https://developer.android.com/training/volley/index.html. [Diakses: 08-Apr-2017].

[39] G. Segato, "An Introduction to Volley," *Code Envato Tuts+*, 13-Mei-2015. [Daring]. Tersedia pada: https://code.tutsplus.com/tutorials/an-introduction-to-volley--cms-23800. [Diakses: 08-Apr-2017].

[40] Google Developers, "SmsManager," *Android Developers*. [Daring]. Tersedia pada: https://developer.android.com/reference/android/telephony/SmsManager.html. [Diakses: 08-Apr-2017].

[41] A. Dustman, "MySQL-python 1.2.5 : Python Package Index," *Python.org*, 2017. [Daring]. Tersedia pada: https://pypi.python.org/pypi/MySQL-python/1.2.5. [Diakses: 08-Apr-2017].

[42] A. Dustman, "MySQLdb: a Python interface for MySQL," *MySQLdb User's Guide*. [Daring]. Tersedia pada: http://mysql-python.sourceforge.net/MySQLdb.html. [Diakses: 08-Apr-2017].

[43] Python Software Foundation, "20.6. urllib2 — extensible library for opening URLs — Python 2.7.13 documentation," *Python.org*. [Daring]. Tersedia pada:

https://docs.python.org/2/library/urllib2.html. [Diakses: 08-Apr-2017].

[44] Anonim, "How to Use Urllib2 in Python," *Python For Beginners*, 22-Feb-2013. [Daring]. Tersedia pada: http://www.pythonforbeginners.com/python-on-the-web/how-to-use-urllib2-in-python/. [Diakses: 08-Apr-2017].

LAMPIRAN

Tabel 7.1 Rincian Hasil Uji Coba UJ-P01 pada Satu *Node* Sensor

Satu *Node*		*Node* Sensor		
		1	2	3
Percobaan ke-	1	2,825	5,767	0,976
	2	0,667	2,654	1,921
	3	0,709	6,086	2,041
	4	0,729	5,347	1,439
	5	0,658	3,124	1,685
	6	0,714	2,682	1,167
	7	0,728	2,955	2,685
	8	0,639	2,635	2,097
	9	5,74	2,412	1,419
	10	0,776	3,152	1,25
Rata - rata		*1,419*	*3,681*	*1,668*
Rata - rata Keseluruhan		**2,256 detik**		

Tabel 7.2 Rincian Hasil Uji Coba UJ-P01 pada Dua *Node* Sensor Sekaligus

Dua *Node* Sekaligus		*Node* Sensor					
		1 dan 2		1 dan 3		2 dan 3	
		1	2	1	3	2	3
Percobaan ke-	1	0,697	3,177	1,62	1,992	3,036	4,084
	2	1,697	5,673	1,776	4,708	5,406	1,974
	3	4,412	5,891	0,781	1,742	5,78	1,859
	4	4,247	5,08	4,226	4,462	1,371	1,858

Dua *Node* Sekaligus		*Node* Sensor					
		1 dan 2		1 dan 3		2 dan 3	
		1	2	1	3	2	3
	5	1,6	5,698	0,701	1,636	4,616	5,821
	6	5,011	5,164	4,501	4,595	4,395	5,811
	7	3,609	5,855	0,888	4,559	0,949	5,734
	8	1,872	5,657	0,676	1,556	1,333	5,799
	9	4,622	5,782	0,729	1,78	0,987	5,726
	10	1,777	2,665	0,669	2,575	2,073	5,718
Rata - rata		*4,009*		*2,259*		*3,717*	
Rata - rata Keseluruhan		**3,328 detik**					

Tabel 7.3 Rincian Hasil Uji Coba UJ-P01 pada Tiga *Node* Sensor Sekaligus

Tiga *Node* Sekaligus		*Node* Sensor		
		1, 2 dan 3		
		1	2	3
Percobaan ke-	1	1,051	6,159	2,934
	2	4,212	5,639	5,683
	3	0,709	5,002	2,262
	4	4,667	5,679	5,723
	5	0,954	5,791	5,706
	6	0,978	5,72	5,789
	7	0,721	2,99	5,781
	8	0,731	0,683	2,864
	9	1,727	5,518	1,303

Tiga *Node* Sekaligus		*Node* Sensor		
		1, 2 dan 3		
		1	2	3
	10	1,558	6,174	4,317
Rata - rata		*1,731*	*4,936*	*4,236*
Rata - rata Keseluruhan		**3,634 detik**		

Tabel 7.4 Rincian Hasil Uji Coba UJ-P02 (*Delay* Internet)

via Internet	Percobaan ke-										**Rata - rata**
	1	2	3	4	5	6	7	8	9	10	
Delay (milidetik)	595	895	94	207	393	544	750	929	142	295	**484,4 milidetik**

Tabel 7.5 Rincian Hasil Uji Coba UJ-P02 (*Delay* SMS)

via SMS	Percobaan ke-										Rata - rata
	1	2	3	4	5	6	7	8	9	10	
Delay (detik)	9,00	8,23	12,42	8,59	5,75	5,89	6,04	6,19	9,36	5,53	**7,702 detik**

Tabel 7.6 Rincian Hasil Uji Coba UJ-P02 (*Delay* Bluetooth)

Bluetooth	Percobaan ke-										**Rata - rata**
	1	2	3	4	5	6	7	8	9	10	
Delay (milidetik)	36	37	37	35	38	37	36	36	37	38	**36,7 milidetik**

Tabel 7.7 Rincian Hasil Uji Coba UJ-P03 pada Jarak ≤1 Meter

Pengiriman Data via Bluetooth (≤1 meter)		Percobaan ke-										Akurasi
		1	2	3	4	5	6	7	8	9	10	
Jumlah Data Masuk	Sesuai	39	34	41	42	39	35	39	39	41	40	92,62%
	Tidak	3	8	1	0	3	7	3	3	1	2	

Tabel 7.8 Rincian Hasil Uji Coba UJ-P03 pada Jarak ±5 Meter

Pengiriman Data via Bluetooth (±5 meter)		Percobaan ke-										Akurasi
		1	2	3	4	5	6	7	8	9	10	
Jumlah Data Masuk	Sesuai	29	33	32	40	38	42	42	38	38	36	87,62%
	Tidak	13	9	10	2	4	0	0	4	4	6	

Tabel 7.9 Rincian Hasil Uji Coba UJ-P03 pada Jarak ±10 Meter

Pengiriman Data via Bluetooth (±10 meter)		Percobaan ke-										Akurasi
		1	2	3	4	5	6	7	8	9	10	
Jumlah Data Masuk	Sesuai	34	31	41	36	40	35	32	24	24	24	76,43%
	Tidak	8	11	1	6	2	7	10	18	18	18	

Tabel 7.10 Rincian Hasil Uji Coba UJ-P04 Penggunaan Daya *Node* Sensor

Daya Baterai 220 mAh		Prosentase Penggunaan Daya (%)	
		5 detik	10 detik
Percobaan ke-	1	31%	16%
	2	25%	11%
	3	30%	12%
	4	34%	17%
	5	24%	8%
	6	22%	9%

Daya Baterai 220 mAh		Prosentase Penggunaan Daya (%)	
		5 detik	**10 detik**
	7	28%	14%
	8	36%	13%
	9	29%	10%
	10	35%	12%
Rata - rata Prosentase (%)		**29,4%**	**12,2%**

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Muhammad Divi Jaya Nuryanto merupakan anak dari pasangan Bapak Sugiyanto (Almarhum) dan Ibu Eny Djulianingsih. Lahir di Malang pada tanggal 26 November 1994. Penulis menempuh pendidikan formal dimulai dari TK Aisyah Bustanul Atfal 9 (ABA 9) Malang (1999-2001), Madrasah Ibtidaiyah Khadijah Malang (2001-2007), SMPN 1 Malang (2007-2010), SMAN 1 Malang (2010-2013) dan sesudah lulus dari SMAN 1 Malang melanjutkan menimba ilmu di jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember, Surabaya (2013-2017). Bidang studi yang diambil oleh penulis pada saat berkuliah di Teknik Informatika ITS adalah Komputasi Berbasis Jaringan (KBJ). Penulis aktif dalam organisasi seperti Himpunan Mahasiswa Teknik Computer-Informatika (2014-2015) dan KMI (2015-2016) sebagai Staf Departemen Pengembangan Profesi Teknik Computer-Informatika 2014-2015 dan Staf Ahli Departemen Syiar Keluarga Muslim Informatika 2015-2016. Selain itu, juga memiliki pengalaman kepanitiaan, diantaranya sebagai Staf WEB 3D Schematics 2014 dan Staf National Logic Competition (Divisi Soal) Schematics 2015. Penulis juga menyukai kegiatan sosial. Penulis memiliki hobi membaca buku dan menyukai hal baru. Penulis dapat dihubungi melalui email: muhammaddivijayanuryanto@gmail.com.