

**Laporan Tugas Akhir**  
**RANCANG BANGUN MODEL *MONITORING DAN PENGENDALIAN***  
**AIR DALAM SISTEM *SMART BUILDING BERBASIS IoT***



Laporan Tugas Akhir ini diajukan untuk Melengkapi Sebagian  
Persyaratan Ujian Akhir D3 Teknik Telekomunikasi

Disusun oleh

<b>FITRI YUNI ASTUTI</b>	<b>3.33.18.0.09</b>
<b>YULIA SETIANI</b>	<b>3.33.18.0.24</b>

**PROGRAM STUDI D3 TEKNIK TELEKOMUNIKASI**  
**JURUSAN TEKNIK ELEKTRO**  
**POLITEKNIK NEGERI SEMARANG**

2021

**Laporan Tugas Akhir**  
**RANCANG BANGUN MODEL *MONITORING DAN PENGENDALIAN***  
**AIR DALAM SISTEM *SMART BUILDING BERBASIS IoT***



Laporan Tugas Akhir ini diajukan untuk Melengkapi Sebagian  
Persyaratan Ujian Akhir D3 Teknik Telekomunikasi

Disusun oleh

<b>FITRI YUNI ASTUTI</b>	<b>3.33.18.0.09</b>
<b>YULIA SETIANI</b>	<b>3.33.18.0.24</b>

**PROGRAM STUDI D3 TEKNIK TELEKOMUNIKASI**  
**JURUSAN TEKNIK ELEKTRO**  
**POLITEKNIK NEGERI SEMARANG**

2021

## **PERNYATAAN KEASLIAN TUGAS AKHIR**

Saya menyatakan dengan sesungguhnya bahwa tugas akhir dengan judul **“RANCANG BANGUN MODEL *MONITORING DAN PENGENDALIAN AIR DALAM SISTEM SMART BUILDING BERBASIS IoT”*** yang dibuat untuk melengkapi sebagian persyaratan menjadi Ahli Madya pada Program Studi D3 Teknik Telekomunikasi Jurusan Teknik Elektro Politeknik Negeri Semarang, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari tugas akhir yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar Ahli Madya di lingkungan Politeknik Negeri Semarang maupun di Perguruan Tinggi atau Instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Semarang, April 2021

Mahasiswa I

Mahasiswa II

Fitri Yuni Astuti

NIM. 3.33.18.0.09

Yulia Setiani

NIM. 3.33.18.0.24

## **HALAMAN PERSUJUAN**

Tugas akhir yang berjudul “**RANCANG BANGUN MODEL MONITORING DAN PENGENDALIAN AIR DALAM SISTEM SMART BUILDING BERBASIS IoT**” dibuat untuk melengkapi sebagian persyaratan menjadi Ahli Madya pada Program Studi D3 Teknik Telekomunikasi, Jurusan Teknik Elektro Politeknik Negeri Semarang dan disetujui untuk diajukan dalam sidang ujian tugas akhir.

Semarang, April 2021

Pembimbing I

Pembimbing II

Sarono Widodo, S.T., M.Kom

NIP. 196403091991031003

Arif Nursyahid, Drs., M.T.

NIP. 196107171986031001

Mengetahui,

Ketua Program Studi

Helmy, S.T., M.Eng.

NIP. 197908102006041001

## **HALAMAN PENGESAHAN**

Tugas akhir dengan judul “**RANCANG BANGUN MODEL MONITORING DAN PENGENDALIAN AIR DALAM SISTEM SMART BUILDING BERBASIS IoT**” telah dipertahankan dalam ujian wawancara dan diterima sebagai syarat untuk menjadi Ahli Madya pada Program Studi D3 Teknik Telekomunikasi, Jurusan Teknik Elektro Politeknik Negeri Semarang pada tanggal **22 Desember 2020**

Tim Pengaji,

Pengaji I

Pengaji II

Pengaji III

.....

.....

.....

NIP.

NIP.

NIP.

Ketua,

Sekretaris

.....

.....

NIP.

NIP.

Mengetahui,

Ketua Jurusan Teknik Elektro

Yusnan Badruzzaman, S.T.,M.Eng.,

NIP. 197503132006041001

## KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa, yang telah memberikan rahmat dan hidayah-nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul **“RANCANG BANGUN MODEL MONITORING DAN PENGENDALIAN AIR DALAM SISTEM SMART BUILDING BERBASIS IoT”**. Tugas akhir ini merupakan salah satu persyaratan untuk menjadi Ahli Madya pada Program Studi D3 Teknik Telekomunikasi Jurusan Teknik Elektro Politeknik Negeri Semarang.

Dalam pelaksanaan dan penyelesaian laporan tugas akhir ini penulis dibantu oleh banyak pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada :

1. Allah SWT.
2. Bapak Ir. Supriyadi, M.T., selaku Direktur Politeknik Negeri Semarang.
3. Bapak Yusnan Badruzzaman, S.T., M.Eng., selaku Ketua Jurusan Teknik Elektro.
4. Bapak Helmy, S.T., M.Eng., Selaku Ketua Prodi D3 Teknik Telekomunikasi Politeknik Negeri Semarang.
5. Bapak Sarono Widodo, S.T., M.Kom. selaku pembimbing I yang telah banyak berjasa, membantu, mendukung penuh dan membimbing saya dalam penulisan Tugas Akhir ini.
6. Bapak Arif Nursyahid, Drs., M.T. selaku pembimbing II yang telah banyak berjasa, membantu, mendukung penuh dan membimbing saya dalam penulisan Tugas Akhir ini.
7. Bapak, Ibu Dosen, dan Staff Teknik Prodi D3 Teknik Telekomunikasi Politeknik Negeri Semarang.
8. Bapak, Ibu, dan semua keluarga penulis yang selalu memberikan dukungan dan doa yang tulus.
9. Sahabat-sahabat penulis yang selalu memberikan dukungan.
10. Teman-teman seperjuangan Program Studi D3 Teknik Telekomunikasi Politeknik Negeri Semarang angkatan 2018.

Semarang, April 2021

Penulis

## **ABSTRAK**

*Fitri Yuni Astuti dan Yulia Setiani. ““RANCANG BANGUN MODEL MONITORING DAN PENGENDALIAN AIR DALAM SISTEM SMART BUILDING BERBASIS IoT”, Tugas Akhir Diploma III Teknik Telekomunikasi Jurusan Teknik Elektro Politeknik Negeri Semarang, dibawah bimbingan Sarono Widodo, S.T., M.Kom. dan Arif Nursyahid, Drs., M.T.*

Air merupakan kebutuhan pokok manusia untuk melangsungkan kehidupan dan meningkatkan kesejahteraan hidup. Pembangunan di bidang Sumber Daya Air (SDA) pada dasarnya merupakan upaya untuk memenuhi kebutuhan air tersebut. Hingga saat ini, air tidak bisa terpisahkan dari aspek kehidupan. Instalasi sistem pemantauan dan pengendalian penggunaan air pada gedung-gedung instansi seperti sekolah atau perusahaan sering kali menggunakan *switch* manual yang mengakibatkan pemborosan air karena kurang terpantau jumlah pemakaian air yang dikarenakan sistem yang masih manual. Oleh karena itu dibuatlah rancang bangun *monitoring* dan pengendalian penggunaan air melalui web dan android menggunakan Arduino sebagai mikrokontroler dan memanfaatkan teknologi LoRa untuk transmisi data yang dilengkapi selenoid valve dan *flowmeter* untuk keperluan pengendalian dan *monitoring* air. Dengan menggunakan sistem ini operator dapat memonitoring penggunaan air dan apabila terjadi penggunaan air yang tidak wajar dapat dikendalian melalui web maupun android kapan saja sesuai dengan kebutuhan. Sistem ini juga dilengkapi dengan notifikasi kebocoran yang dikirim melalui *platform* Telegram.

Kata kunci : Air, *Smart Building*, Selenoid Valve, *Flowmeter*, LoRa, kontrol, *monitoring*

## **ABSTRACT**

*Fitri Yuni Astuti and Yulia Setiani. "DESIGN OF MONITORING AND WATER CONTROL MODEL IN SMART BUILDING SYSTEM BASED ON IoT", Final Project Diploma III of Telecommunication Engineering Department of Electrical Engineering Semarang State Polytechnic, under the guidance of Sarono Widodo, S.T., M.Kom. and Arif Nursyahid, Drs., M.T.*

*Water is a basic human need to carry on life and improve the welfare of life. Development in the field of Water Resources (SDA) is basically an effort to meet these water needs. Until now, water cannot be separated from aspects of life. Installations of water use monitoring and control systems in institutional buildings such as schools or companies often use manual switches which result in waste of water due to the lack of monitoring of the amount of water usage due to the manual system. Therefore a design for monitoring and controlling water use via the web and Android was made using Arduino as a microcontroller and utilizing LoRa technology for data transmission which is equipped with a selenoid valve and flowmeter for water control and monitoring purposes. By using this system the operator can monitor water usage and if there is an unreasonable use of water it can be controlled via the web or Android at any time as needed. This system is also equipped with leak notifications sent via the Telegram platform.*

*Keywords:* Water, Smart Building, Selenoid Valve, Flowmeter, LoRa, control, monitoring

## DAFTAR ISI

HALAMAN JUDUL.....	i
PERNYATAAN KEASLIAN TUGAS AKHIR.....	ii
HALAMAN PERSUJUAN.....	iii
HALAMAN PENGESAHAN.....	iv
KATA PENGANTAR .....	v
ABSTRAK .....	vi
<i>ABSTRACT</i> .....	vii
DAFTAR ISI.....	viii
DAFTAR GAMBAR .....	xi
DAFTAR TABEL.....	xiv
DAFTAR LAMPIRAN.....	xv
BAB I .....	1
PENDAHULUAN .....	1
1.1    Latar Belakang .....	1
1.2    Perumusan Masalah.....	2
1.3    Tujuan dan Manfaat.....	2
1.4    Batasan Masalah.....	3
1.5    Metode Penelitian.....	3
1.6    Sistematika Penulisan.....	4
BAB II.....	6
LANDASAN TEORI.....	6
2.1    Selenoid Valve .....	6
2.1.1    Prinsip Kerja Selenoid Valve.....	7
2.2 <i>Flowmeter</i> .....	8
2.2.1    Prinsip Kerja <i>Flowmeter</i> .....	9
2.3    Mikrokontroler .....	10
2.3.1    Prinsip Kerja Arduino Uno .....	12
2.4    Node MCU .....	13
2.5 <i>Lora Shield SX1272</i> .....	15
2.5.1    Prinsip Kerja LoRa <i>Shield</i> .....	17

2.6	LoRa Bee.....	18
2.7	<i>Power Supply</i> .....	20
2.7.1	Prinsip Kerja <i>Power Supply</i> .....	21
2.8	<i>Relay</i> .....	24
2.8.1	Prinsip Kerja <i>Relay</i> .....	25
BAB III .....		21
PERANCANGAN DAN PEMBUATAN SISTEM.....		21
3.1	Perencanaan Pembuatan Sistem .....	21
3.2	Perancangan Sistem.....	29
3.2.1	Rancang Bangun Sistem.....	30
3.2.2	Kebutuhan Alat <i>Hardware</i> (Perangkat Keras) .....	39
3.2.3	Kebutuhan Alat <i>Software</i> (Perangkat Lunak).....	41
3.2.4	Perancangan Algoritma Sistem .....	41
3.2.5	Perancangan <i>Prototype</i> .....	54
3.3	Pembuatan Sistem .....	58
3.3.1	Pembuatan <i>Software</i> (Perangkat Lunak) .....	58
3.3.2	Pembuatan <i>Prototype</i> Sistem .....	62
BAB IV .....		80
PENGUJIAN DAN ANALISA.....		80
4.1	Metodologi Pengujian .....	80
4.2	Hasil Pengujian dan Analisa.....	83
4.2.1	Hasil pengujian jarak cakupan sel gedung, koordinator, dan <i>gateway</i> .....	83
4.2.2	Pengujian pengiriman data sensor sel pusat dan gedung ke <i>node</i> koordinator .....	87
4.2.3	Pengujian pengiriman data dari <i>node</i> koordinator ke <i>gateway</i> .....	90
4.2.4	Pengujian pengiriman data dari <i>gateway</i> ke <i>database Web Server</i> . ....	95
4.2.5	Pengujian penerimaan data perintah dari <i>database Web Server</i> ke <i>gateway</i> .....	97
4.2.6	Pengujian penerimaan data perintah dari <i>gateway</i> ke <i>node</i> koordinator .....	98
4.2.7	Pengujian penerimaan data perintah dari <i>node</i> koordinator ke sel pusat atau sel gedung. ....	100

4.2.8	Pengujian pengiriman notifikasi sel gedung ke <i>platform Telegram</i> ....	102
4.2.9	Pengujian keakuratan data sensor <i>flowmeter</i> . ....	103
BAB V	.....	106
PENUTUP	.....	106
5.1	Kesimpulan.....	106
5.2	Saran .....	107
DAFTAR PUSTAKA	.....	108
LAMPIRAN		

## DAFTAR GAMBAR

Gambar 2.1 Selenoid valve .....	6
Gambar 2.2 Prinsip kerja selenoid valve.....	7
Gambar 2.3 <i>Flowmeter</i> YF-S201 G1 .....	8
Gambar 2.4 Prinsip kerja <i>flowmeter</i> .....	10
Gambar 2.5 Arduino Uno.....	11
Gambar 2.6 Cara kerja Arduino .....	12
Gambar 2.7 NodeMCU Esp-8266.....	13
Gambar 2.8 <i>Datasheet</i> NodeMCU Esp-8266 .....	14
Gambar 2.9 LoRa <i>Shield</i> .....	16
Gambar 2.10 <i>Datasheet</i> SX1272 .....	16
Gambar 2.11 Prinsip kerja modul LoRa .....	17
Gambar 2.12 Pin Diagram <i>Chip</i> SX1276.....	18
Gambar 2.13 Gambar LoRa Bee .....	18
Gambar 2.14 <i>Power Supply</i> 12Volt.....	20
Gambar 2.15 <i>Power Supply</i> 5Volt .....	20
Gambar 2.16 Diagram blok DC <i>power supply</i> .....	21
Gambar 2.17 Skematik rangkaian <i>power supply</i> .....	22
Gambar 2.18 <i>trafo step-down</i> .....	22
Gambar 2.19 <i>Rectifier</i> .....	23
Gambar 2.20 <i>Filter</i> .....	23
Gambar 2.21 Rangkaian <i>voltage regulator</i> .....	24
Gambar 2.22 <i>Relay</i> .....	24
Gambar 2.23 Jenis <i>relay</i> berdasarkan <i>pole</i> dan <i>throw</i> .....	25
Gambar 2.24 Struktur sederhana <i>relay</i> .....	26
Gambar 2.25 Struktur dasar <i>relay</i> .....	27
Gambar 3.1 <i>Layout</i> sistem air .....	30
Gambar 3.2 Blok diagram sistem air dalam <i>smart building</i> .....	31
Gambar 3.3 <i>Flowchart</i> perancangan sistem monitoring dan pengendalian air.....	42
Gambar 3.4 <i>Flowchart node</i> sel .....	43
Gambar 3.5 <i>Flowchart Node</i> Koordinator 433 Mhz.....	45

Gambar 3.6 <i>Flowchart Node</i> Koodinator 915 MHz .....	46
Gambar 3.7 <i>Flowchart gateway</i> .....	48
Gambar 3.8 Pengkabelan perangkat keras <i>node</i> pusat dan gedung .....	54
Gambar 3.9 Pemasangan LoRa <i>Shield</i> ke Arduino Uno.....	54
Gambar 3.10 Pemasangan pengkabelan <i>node</i> koordinator .....	55
Gambar 3.11 Pengkabelan pada <i>gateway</i> .....	56
Gambar 3.12 Jalur PCB <i>gateway</i> .....	57
Gambar 3.13 Tampilan <i>License Agreement</i> .....	59
Gambar 3.14 Tampilan <i>Installation Options</i> .....	59
Gambar 3.15 Tampilan <i>Installation Folder</i> .....	60
Gambar 3.16 Tampilan Proses <i>Installing</i> .....	60
Gambar 3.17 Tampilan <i>Installing Complete</i> .....	61
Gambar 3.18 Tampilan Awal Arduino IDE.....	61
Gambar 3.19 Tampilan <i>editor</i> Arduino IDE.....	62
Gambar 3.20 <i>prototype</i> alat.....	63
Gambar 3.21 Pengkabelan perangkat sel gedung .....	65
Gambar 3.22 Pengkabelan perangkat Koordinator .....	66
Gambar 3.23 Pemasangan perangkat <i>gateway</i> .....	66
Gambar 3.24 Tampilan Port COM pada sel gedung .....	68
Gambar 3.25 Tampilan proses <i>compile</i> pada sel gedung .....	69
Gambar 3.26 Tampilan <i>done compiling</i> pada sel gedung .....	71
Gambar 3.27 Tampilan proses <i>upload</i> pada sel gedung .....	72
Gambar 3.28 Tampilan <i>done uploading</i> .....	73
Gambar 3.29 Tampilan data <i>controlling</i> dan monitoring yang akan dikirim dan diterima dari sel gedung ke koordinator.....	73
Gambar 3.30 Tampilan <i>port</i> COM pada koordinator.....	74
Gambar 3.31 Tampilan proses <i>compile</i> pada koordinator .....	74
Gambar 3.32 Tampilan <i>done compiling</i> pada koordinator.....	75
Gambar 3.33 Tampilan proses <i>upload</i> pada koordinator .....	75
Gambar 3.34 Tampilan <i>done uploading</i> .....	76
Gambar 3.35 Tampilan data <i>controlling</i> dan monitoring yang akan dikirim dan diterima ke sel gedung dan <i>gateway</i> .....	76

Gambar 3.36 Tampilan <i>port COM</i> pada <i>gateway</i> .....	77
Gambar 3.37 Tampilan proses <i>compile</i> pada <i>gateway</i> .....	77
Gambar 3.38 Tampilan <i>done compiling</i> pada <i>gateway</i> .....	78
Gambar 3.39 Tampilan proses <i>upload</i> pada <i>gateway</i> .....	78
Gambar 3.40 Tampilan <i>done uploading</i> pada <i>gateway</i> .....	79
Gambar 3.41 Tampilan data <i>controlling</i> dan monitoring pada <i>gateway</i> .....	79
Gambar 4.1 Letak sel gedung, koordinator, dan <i>gateway</i> untuk pengujian pertama dilihat dari satelit.....	84
Gambar 4.2 Letak sel gedung, koordinator, dan <i>gateway</i> untuk pengujian kedua dilihat dari satelit.....	85
Gambar 4.3 Letak sel gedung, koordinator, dan <i>gateway</i> untuk pengujian ketiga dilihat dari satelit.....	86
Gambar 4.4 Gambar tampilan serial monitor pengiriman data dari <i>gateway</i> ke <i>database Web Server</i> .....	96
Gambar 4.5 Tampilan serial monitor pengujian penerimaan data perintah dari <i>database</i> ke <i>gateway</i> .....	97
Gambar 4.6 Tampilan notifikasi Telegram .....	102
Gambar 4.7 Tampilan serial monitor pengiriman keakuratan data sensor <i>flowmeter</i> .....	104

## DAFTAR TABEL

Tabel 3.1 Format data yang dikirim sel pusat dan gedung ke koordinator 433Mhz .....	35
Tabel 3.2 Format data yang diterima Koordinator 433MHz .....	35
Tabel 3.3 Format data yang diterima Koordinator 915 MHz .....	36
Tabel 3.4 Format data yang dikirim koordinator 915Mhz ke <i>gateway</i> .....	36
Tabel 3.5 Format data pengendalian yang dikirim <i>gateway</i> ke koordinator 915Mhz .....	38
Tabel 3.6 Format data pengendalian yang dikirim koordinator 915Mhz ke koordinator 433 Mhz.....	38
Tabel 3.7 Format data pengendalian yang dikirim koordinator 433Mhz ke masing-masing gedung.....	39
Tabel 3.8 Spesifikasi <i>Hardware</i> (perangkat keras).....	39
Tabel 3.9 <i>Syntax</i> program yang digunakan pada sel gedung .....	49
Tabel 3.10 <i>Syntax</i> program yang digunakan pada <i>node</i> Koordinator .....	51
Tabel 3.11 <i>Syntax</i> program yang digunakan pada <i>gateway</i> .....	52
Tabel 3.12 Pengkabelan <i>node</i> koordinator.....	55
Tabel 3.13 Pengkabelan <i>gateway</i> .....	58
Tabel 4.1 hasil pengujian data dari gedung ke <i>node</i> koordinator .....	87
Tabel 4.2 Hasil pengujian pengiriman data dari <i>node</i> koordinator ke <i>gateway</i> ....	90
Tabel 4.3 Hasil pengujian penerimaan data perintah dari <i>gateway</i> ke <i>node</i> koordinator .....	99
Tabel 4.4 Pengujian penerimaan data perintah dari <i>node</i> koordinator ke sel pusat atau sel gedung.....	101
Tabel 4.5 Hasil pengujian keakuratan data sensor <i>flowmeter</i> .....	103

## **DAFTAR LAMPIRAN**

- Lampiran 1    Program sel gedung
- Lampiran 2    Program *node* koordinator
- Lampiran 3    Program *gateway*

## **BAB I**

### **PENDAHULUAN**

#### **1.1 Latar Belakang**

Air merupakan kebutuhan pokok manusia untuk melangsungkan kehidupan dan meningkatkan kesejahteraan hidup. Pembangunan di bidang Sumber Daya Air (SDA) pada dasarnya merupakan upaya untuk memenuhi kebutuhan air tersebut. Hingga saat ini, air tidak bisa terpisahkan dari aspek kehidupan. Sistem pemanfaatan pengendalian penggunaan air banyak diterapkan pada gedung-gedung instansi seperti sekolah atau perusahaan. Sistem pengendalian penggunaan air perlu pengelolaan yang tepat. Sehingga pemanfaatan pengendalian penggunaan air dapat dilakukan secara efisien dan *real time*.

Instalasi sistem pemantauan dan pengendalian penggunaan air pada gedung-gedung instansi seperti sekolah atau perusahaan sering kali menggunakan *switch* manual yang mengakibatkan pemborosan air karena kurang terpantau jumlah pemakaian air yang dikarenakan sistem yang masih manual. Ditambah dengan faktor *human error* yang sering lupa tidak mematikan kran air dan tidak mengetahui adanya kebocoran air pada sistem yang masih manual. Sistem ini terdapat kekurangan dalam efisiensi tenaga kerja dan tidak dapat memantau secara *real time*. Sistem ini memerlukan admin untuk melakukan pemantauan dan pengendalian secara *real time* dan jarak jauh.

Dengan permasalahan diatas maka diperlukan sistem yang dapat *memonitoring* penggunaan air jarak jauh dengan memanfaatkan teknologi yang telah berkembang, *Internet of Things* (IoT). IoT bekerja dengan membaca sensor *flowmeter* dan kemudian mengirim data sensor tersebut ke *gateway* untuk dapat diakses melalui internet. Dengan menggunakan teknologi ini, sistem dapat dipantau dan dikendalikan secara *real time* dan jarak jauh melalui internet dalam sebuah halaman web dan andorid. Selain lebih efesien dalam *memonitoring* penggunaan air, sistem juga dapat mengendalikan penggunaan air yang tidak wajar serta dapat mematikan dan

menyalakan air menggunakan web atau android dan dapat di akses jarak jauh. Hal inilah yang menjadi latar belakang kami dalam membuat Rancangan Model *Monitoring* dan Pengendalian Air dalam Sistem *Smart Building* Berbasis *IoT* dengan penggunaan air dapat di *monitoring* melalui web dan atau andorid yang di rekam ke dalam *database* dan dapat melakukan pengendalian penggunaan air apabila terjadi kebocoran atau pemakaian yang tidak wajar.

## 1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, dapat dirumuskan permasalahannya sebagai berikut :

1. Bagaimana cara memonitoring pemakaian volume air yang telah digunakan?
2. Bagaimana rancang bangun pengendalian air dapat mendeteksi kebocoran?
3. Bagaimana *gateway* dapat menerima informasi dari *web* untuk diteruskan dan dapat menerima informasi dari *node koordinator* untuk dikiriman ke *web*?

## 1.3 Tujuan dan Manfaat

Tujuan dari pembuatan tugas akhir ini adalah :

1. Menghasilkan suatu sistem yang dapat digunakan untuk memonitoring penggunaan air secara *real time* dengan jarak jauh melalui *web*.
2. Menghasilkan suatu sistem yang dapat digunakan untuk mengendalikan penggunaan air yang tidak wajar apabila terdapat kebocoran melalui *web*.

Manfaat dari pembuatan tugas akhir ini adalah memberikan kemudahan *user* untuk memonitoring penggunaan air dan mengendalikan penggunaan air termasuk jika terjadi kebocoran.

#### 1.4 Batasan Masalah

Berdasarkan latar belakang yang dijabarkan, maka batasan masalah dalam tugas akhir ini adalah :

1. Pembuatan sistem yang bertujuan untuk memonitoring dan mengendalikan penggunaan air secara *real time*.
2. Pembuatan sistem menggunakan selenoid valve dan *flowmeter* untuk mengetahui kondisi alat dan memonitoring serta melakukan pengendalian penggunaan air.
3. Selenoid valve digunakan untuk mematikan dan menyalakan aliran air.
4. *Flowmeter* digunakan untuk menghitung volume atau aliran dari suatu fluida yang mengalir dalam pipa atau sambungan terbuka.
5. Digunakan selenoid valve jenis DC 12Volt pada penggunaan pipa ukuran 1/2 inch.
6. *Flowmeter* yang digunakan adalah jenis YF-S201 dengan pipa ukuran 1/2 inch.
7. *IoT* digunakan pada sistem *smart building* untuk melakukan pengendalian jika terjadi kebocoran air dan melakukan monitoring penggunaan air.
8. *Node* sel menggunakan modul *LoRa Shield* dengan frekuensi 433 MHz dan Arduino Uno.
9. *Node* koordinator menggunakan modul *LoRa Shield* dengan frekuensi 433 Mhz dan 915 MHz serta Arduino uno.
10. *Gateway* menggunakan modul *LoRa Bee* dengan frekuensi 915 MHz dan *NodeMCU* ESP8266.

#### 1.5 Metode Penelitian

Langkah-langkah yang ditempuh dalam menyelesaikan tugas akhir ini adalah sebagai berikut:

- a. Metode Observasi

Langkah ini merupakan langkah pengumpulan data-data dan menemukan latar belakang masalah yang ada pada penelitian sebelumnya, jurnal-jurnal ataupun pada lingkungan Politeknik Negeri

Semarang. dan pada langkah ini dilakukan untuk memperoleh data yang diperoleh secara langsung dengan cara mewawancara kakak tingkat yang membuat tugas akhir dengan teknologi yang sama, dan mempelajari literatur seperti jurnal, buku referensi.

b. Metode Studi Pustaka

Metode ini dilakukan untuk memperoleh teori maupun referensi yang nantinya digunakan bahan untuk menyusun dasar teori dan sebagai pendukung pembuatan tugas akhir. pada metode ini juga merupakan tahap mencari komponen-komponen beserta fungsi dan *datasheet* nya yang akan digunakan dalam pembuatan tugas akhir

c. Metode Perencanaan Sistem

Metode ini merupakan tahapan perencanaan mengenai estimasi waktu, penjadwalan, dan gambaran desain sistem.

d. Metode Perancangan Sistem

Metode ini merupakan perancangan sistem yang bertujuan untuk membantu mempermudah dalam pembuatan sistem.

e. Metode Pengujian Sistem

Metode ini dilakukan untuk memeriksa fungsi masing-masing komponen yang digunakan dan dilakukan pengujian terhadap sistem yang telah dibuat.

f. Metode Penyusunan Laporan dan Kesimpulan

Metode ini merupakan tahap final dalam pembuatan sistem. Dimana kegiatan yang telah disusun dari awal perencanaan hingga akhir pembuatan sistem dan dari data-data hasil penelitian dijelaskan pada sebuah laporan dan ditarik suatu kesimpulan.

## 1.6 Sistematika Penulisan

Dalam penulisan tugas akhir ini terbagi dalam bab-bab yang sistematik dan memberikan uraian secara rinci agar lebih mudah untuk dipahami. Adapun sistematika penulisan sebagai berikut :

**BAB I PENDAHULUAN**

Bab ini berisi tentang latar belakang masalah, tujuan dan manfaat, batasan masalah, metode penelitian, serta sistematika penulisan.

**BAB II LANDASAN TEORI**

Berisi tentang teori-teori pembuatan sistem monitoring dan pemantauan air dalam sistem *smart building* menggunakan teknologi LoRa untuk transmisi data yang dapat menerima dan mengirimkan konfirmasi nyala dan mati air secara *real time*.

**BAB III PERANCANGAN DAN PEMBUATAN SISTEM**

Berisi tentang perancangan dan pembuatan sistem serta pemasangan perangkat. Dalam hal ini dibagi dua rancangan untuk pembuatan sistem ini yaitu perancangan perangkat lunak dan perancangan perangkat keras.

**BAB IV PENGUJIAN SISTEM**

Menjelaskan tentang hasil pengujian dari sistem yang telah dibuat dan analisis hasil pengujian.

**BAB V PENUTUP**

Bab ini berisi kesimpulan yang dapat diambil dari tugas akhir ini beserta saran untuk pengembangan lebih lanjut.

## **BAB II**

### **LANDASAN TEORI**

#### **2.1 Selenoid Valve**

Selenoid Valve adalah katup yang dikendalikan oleh energi listrik, kumparan solenoid valve berfungsi sebagai penggerak untuk menggerakkan piston yang dapat digerakkan oleh arus AC atau DC. Selenoid valve mempunyai lubang keluaran, lubang masukan dan lubang *exhaust*. Lubang keluaran berfungsi sebagai keluar air, lubang masukan berfungsi sebagai masukan air dan lubang *exhaust* berfungsi mengeluarkan cairan yang terjebak saat piston bergerak. (Stephan Adriansyah Hulukati dan Irvan A Salihi, 2018 )



Gambar 2.1 Selenoid valve

( <https://www.blanja.com/katalog/p/hap/keran-solenoid-valve-12v-1-2-inch-high-pressure-nc-lurus-27434367> )

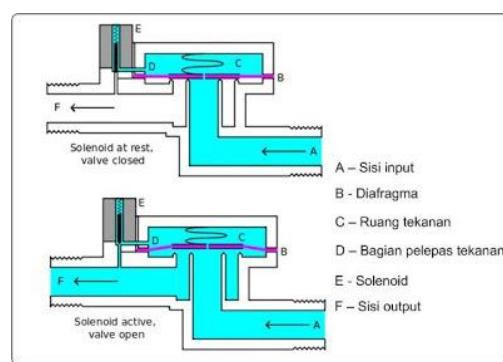
Spesifikasi *New 12V Electric Solenoid Valve Magnetic DC N/C* :

1. Selenoid valve ukuran 1/2" x 1/2" dengan *voltase* DC 12V
2. Tipe *Normally Open* :
  - a. Katup terbuka jika tidak ada aliran listrik ke solenoid.
  - b. Katup tertutup saat ada aliran listrik masuk ke solenoid.
3. Selenoid valve Plastik ini memiliki fungsi untuk membuka atau menutup aliran air.
4. *Material: Metal + plastic*
5. *Voltage: DC 12V*
6. *Power: 8W*
7. *Current: 0.6A*

8. Pressure: 0.02- 0.8Mpa
9. Max fluid temperature:100C
10. Operation mode: normally open
11. Valve type: diaphragm (operated by Servo)
12. Usage: water and low viscosity fluids

### 2.1.1 Prinsip Kerja Selenoid Valve

Prinsip kerja dari solenoid valve yaitu katup listrik yang mempunyai koil sebagai penggeraknya dimana ketika koil mendapat *supply* tegangan maka koil tersebut akan berubah menjadi medan magnet. Selenoid valve yang digunakan adalah tipe *normally open* yang artinya jika tidak diberi tegangan maka katub selenoid terbuka yang mengakibatkan air dapat mengalir dan apabila selenoid valve diberi tegangan 12V DC maka katub selenoid tertutup dan air tidak mengalir. Pada saat katub tertutup sebuah pin akan menutup karena gaya magnet yang dihasilkan dari kumparan selenoida tersebut. Dan saat pin tersebut menutup maka fluida tidak mengalir dari ruang tekanan menuju ke bagian pelepas tekanan. Sehingga katup utama terbuka dan fluida tidak dapat mengalir langsung dari masukan air ke keluaran air. (Rano Romansyah, 2016)



Gambar 2.2 Prinsip kerja selenoid valve

( <http://www.kitomaindonesia.com/article/9/solenoid-valve-pneumatic-prinsip-kerja> )

## 2.2 Flowmeter

*Flowmeter* merupakan suatu alat yang digunakan untuk mengukur jumlah atau aliran dari suatu fluida yang mengalir dalam pipa atau sambungan terbuka. Dalam proses kerja *Flowmeter* yaitu mengukur aliran yang menghasilkan keluaran berupa *flowrate* atau debit air. Debit air biasanya dinyatakan dengan menggunakan satuan liter/jam. Namun satuan liter/jam dapat di sederhanakan lagi menjadi liter/menit atau liter/detik. Satuan tersebut dapat disesuaikan dengan kebutuhan pengguna. (Saptaaji, 2016)

*Water flow sensor Sea 1/2"* atau *Flow sensor YF-S201 1/2"* merupakan *flow sensor* yang dapat digunakan untuk mendeteksi aliran air. *Flowmeter* jenis ini mampu mengukur aliran yang mempunyai *pressure* tertentu diantaranya adalah aliran air yang mempunyai tekanan sangat kecil serta kecepatan aliran air yang minim dan aliran air. *Flowmeter* jenis ini terdapat 3 kabel, diantaranya merah (5 to 18VDC), kuning (data), dan hitam (*ground*). Sensor ini memiliki *Working Voltage*: 5 to 18V DC (*min tested working voltage 4.5V*), sensor dapat berputar dan menghitung aliran air sesuai dengan putaran pada sensor jika sensor memiliki tegangan minimal 4.5V. (Hari Anggoro dan Sujono, 2020)



Gambar 2.3 *Flowmeter YF-S201 G1*

( <https://www.amazon.in/SunRobotics-YF-S201-EFFECT-WATER-SENSOR/dp/B07SB52MDF> )

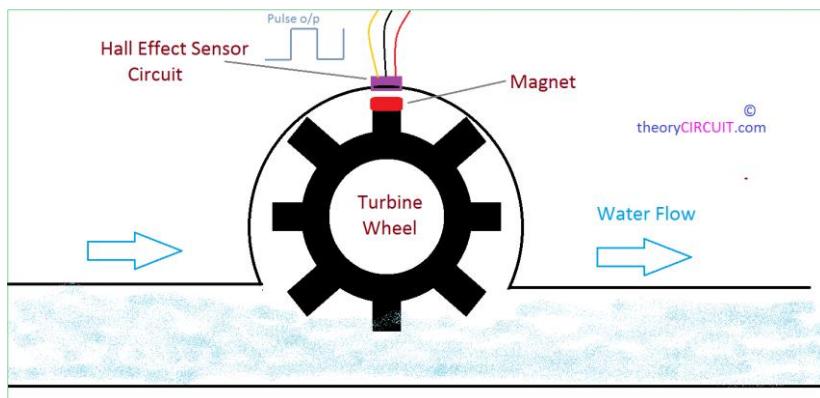
*Features :*

1. *Working Voltage: 5 to 18V DC (min tested working voltage 4.5V)*
2. *Max current draw: 15mA @ 5V*

3. *Output Type:* 5V TTL
4. *Working Flow Rate:* 1 to 30 Liters/Minute
5. *Working Temperature range:* -25 to +80°
6. *Working Humidity Range:* 35%-80% RH
7. *Accuracy:* 10%
8. *Maximum water pressure:* 2.0 MPa
9. *Output duty cycle:* 50% +-10%
10. *Output rise time:* 0.04us
11. *Output fall time:* 0.18us
12. *Flow rate pulse characteristics:* Frequency (Hz) = 7.5 \* Flow rate (L/min)
13. *Pulses per Liter:* 450
14. *Durability:* minimum 300,000 cycles
15. *Cable length:* 15cm
16. *1/2" nominal pipe connections, 0.78" outer diameter, 1/2" of thread*
17. *Size:* 2.5" x 1.4" x 1.4"

### 2.2.1 Prinsip Kerja *Flowmeter*

*Flowmeter* adalah sensor yang di gunakan untuk mengukur kecepatan aliran air per menit . Pengunaan sensor ini menghasilkan nilai kecepatan air yang mengalir melewati *flowmeter*. Prinsip kerja *flowmeter* adalah menghitung kecepatan aliran air berdasarkan dari jumlah putaran kincir yang terdapat di *body* sensor. Di dalam *body* *flowmeter* terdapat *hall effect* sensor dan kincir air. *Hall efect* sensor atau bisa disebut juga sensor medan magnet berfungsi menghitung berapa banyak kincir air berputar. Kincir air berputar sebanding dengan seberapa banyak debit air yang melewati *flowmeter*.



Gambar 2.4 Prinsip kerja *flowmeter*

(<https://theorycircuit.com/water-flow-sensor-yf-s201-arduino-interface/how-water-flow-sensor-works/>)

Air yang mengalir pada *turbine wheel*, maka *turbine wheel* akan berputar sesuai dengan kecepatan air. *Flowmeter* memanfaatkan *hall effect* sensor yang didasarkan pada efek medan magnetik terhadap partikel bermuatan bergerak. Sensor *hall effect* terdapat dalam *flowmeter* tersebut akan mengeluarkan *output* sinyal pulsa sesuai dengan besarnya aliran air. *Flowmeter* tidak akan menghasilkan tegangan apabila sensor belum dialiri air atau belum bekerja dan baru akan menghasilkan tegangan ketika sensor dialiri air. (Ma Zulkarnain, 2015).

### 2.3 Mikrokontroler

Mikrokontroler merupakan sebuah komputer kecil mempunyai prosesor dan memori terbatas yang difungsikan sebagai pengontrol. salah satu mikrokontroler yang di desain untuk memberikan kemudahan dalam mengisikan program adalah Arduino. Arduino merupakan sebuah platfrom elektronika yang bersifat *open-source* dan berbasis untuk perangkat keras dan perangkat lunak dan digunakan untuk memudahkan pengguna menggunakan secara fleksibel. Arduino merupakan perangkat keras yang berbentuk rangkaian elektronik dan memiliki fungsi sebagai pengontrol. Arduino biasanya dapat di perintah dengan menggunakan bahasa pemograman melalui *Software Integrated Development Environment* (IDE).

IDE merupakan *software* yang berfungsi untuk menulis kode program, untuk *compile* menjadi kode biner dan untuk upload kedalam memori mikrokontroler. Dalam tugas akhir ini mikrokontroler yang digunakan adalah Arduino Uno yang merupakan papan mikrokontroler bebas ATmega328P. Arduino Uno memiliki 14 pin *input/output* digital (6 diantaranya dapat digunakan sebagai *output PWM*), 6 *input* analog, 16 MHz *quaterz crystal*, koneksi USB, colokan listrik, *header ICSP*, dan tombol reset. (*Arduino Uno Rev 3*, n.d)



Gambar 2.5 Arduino Uno  
<https://store.arduino.cc/usa/arduino-uno-rev3> )

Adapun spesifikasi dari Arduino Uno adalah sebagai berikut:

Mikrokontroler	: ATmega328P
Tegangan pengoperasian	: 5V
Tegangan <i>input</i>	: 7-12V (disarankan)
Tegangan <i>input</i> (batas)	: 6-20V
Pin I/O Digital	: 14 ( 6 pin menyediakan output PWM)
Pin I/O Digital PWM	: 6
Pin <i>Input</i> Analog	: 6
Arus DC per pin I/O	: 20mA
Arus DC untuk pin 3.3V	: 50mA
<i>Flash Memory</i>	: 32 KB (ATmega328P) dengan 0.5 KB digunakan oleh <i>bootloader</i>
SRAM	: 2 KB (ATmega328P)
EEPROM	: 1 KB (ATmega328P)

Clock Speed	: 16MHz
LED_BUILTIN	: 13
Panjang	: 68.6 mm
Lebar	: 53.4 mm
Berat	: 25 g

### 2.3.1 Prinsip Kerja Arduino Uno

Pada dasarnya cara kerja Arduino terdiri dari komponen input, pin input, proses, pin output, dan komponen output, seperti yang ditunjukan pada gambar 2.3.2



Gambar 2.6 Cara kerja Arduino  
[\(https://www.aldyrazor.com/2020/07/cara-kerja-arduino.html\)](https://www.aldyrazor.com/2020/07/cara-kerja-arduino.html)

Alur kerja Arduino pada dasarnya adalah melakukan pembacaan data oleh komponen *input* yang dihubungkan ke Arduino, misalnya berupa jarak, getaran, atau suara. Selanjutnya data tersebut dikirim ke pin *input*. Pin *input* merupakan perantara yang menghubungkan antara Arduino dan komponen *input*. Sumber *input* pada Arduino berasal dari pin digital atau pin analog yang tersedia pada *board* Arduino. Data yang ada pada *input* Arduino akan dibawa ke mikrokontroler atau inti Arduino untuk masuk ke tahapan berikutnya, yaitu tahap pemrosesan data. Data yang masuk ke mikrokontroler akan diproses berdasarkan perintah atau program yang diberikan. Setelah data diproses, selanjutnya data akan dikirimkan ke pin *output* Arduino. Data yang terdapat pada pin *output* Arduino, selanjutnya akan disalurkan ke komponen *output*, misalnya *relay*, Lampu LED, dll. (Aldy Razor, 2021). Pada tugas akhir ini Arduino dihubungkan dengan LoRa *Shield*. LoRa *Shield*

dihubungkan dengan seluruh pin Arduino, karena berupa *shield* yang kompatible dengan Arduino.

#### 2.4 Node MCU

*NodeMCU* merupakan *open source paltform Internet of Things (IoT)*. Dengan menggunakan bahasa pemrograman Lua dapat membantu programmer dalam membuat *prototype* produk *IoT* atau bisa dengan *sketch* dengan arduino IDE. *NodeMCU* memiliki ukuran panjang 4.83 cm, lebar 2.54 cm, dan dengan berat 7 gram. *Board NodeMCU* telah dilengkapi dengan fitur *WiFi* dan *firmware* yang bersifat *open source*. (Siswanto, Nurhadian, dan Junaedi, 2020).

ESP8266 merupakan salah satu jenis mikrokontroller yang bekerja dengan menghubungkan ke jaringan *WiFi*. ESP8266 dapat dihubungkan dengan Arduino IDE dapat menggunakan kabel *micro USB*. (Yudita dan Nidya, 2018)

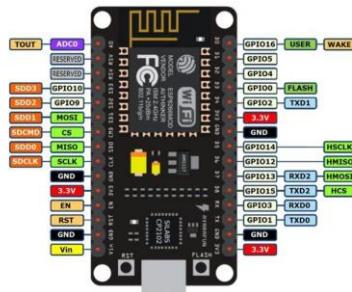


Gambar 2.7 NodeMCU Esp-8266  
<https://www.jakartanotebook.com/v3-wireless-module-nodemcu-4m-bytes-lua-esp8266-esp-12e-for-arduino#>

Spesifikasi yang dimiliki oleh *NodeMCU Esp-8266* sebagai berikut :

1. *Board* ini berbasis ESP8266 serial WiFi SoC (*Single on Chip*) dengan *onboard* USB to TTL. *Wireless* yang digunakan adalah IEE 802.11b/g/n.
2. 2 tantalum kapasitor 100 *micro farad* dan 10 *micro farad*.
3. 3.3v LDO regulator.

4. *Blue led* sebagai indikator.
5. Cp2102 usb to UART *bridge*.
6. Tombol *reset*, *port* usb, dan tombol *flash*.
7. Terdapat 9 GPIO yang di dalamnya ada 3 pin PWM, 1 x ADC *Channel*, dan pin RX TX
8. 3 pin *ground*.
9. S3 dan S2 sebagai pin GPIO
10. S1 MOSI (*Master Output Slave Input*) yaitu jalur data dari master dan masuk ke dalam *slave*, sc cmd/sc.
11. S0 MISO (*Master Input Slave Input*) yaitu jalur data keluar dari *slave* dan masuk ke dalam master.
12. SK yang merupakan SCLK dari master ke *slave* yang berfungsi sebagai *clock*.
13. Pin Vin sebagai masukan tegangan.
14. *Built* in 32-bit MCU.



Gambar 2.8 *Datasheet* NodeMCU Esp-8266  
<https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet> )

#### *Datasheet* NodeMCU esp-8266

1. RST : berfungsi mereset modul
2. ADC: *Analog Digital Converter*. Rentang tegangan masukan 0-1v, dengan skup nilai digital 0-1024
3. EN: *Chip Enable, Active High*
4. IO16 :GPIO16, dapat digunakan untuk membangunkan *chipset* dari *mode deep sleep*

5. IO14 : GPIO14; HSPI\_CLK
6. IO12 : GPIO12: HSPI\_MISO
7. IO13: GPIO13; HSPI\_MOSI; UART0\_CTS 5
8. VCC: Catu daya 3.3V (VDD)
9. CS0 : *Chip selection*
10. MISO : *Slave output, Main input*
11. IO9 : GPIO9
12. IO10 GBIO10
13. MOSI: *Main output slave input*
14. SCLK: *Clock*
15. GND: *Ground*
16. IO15: GPIO15; MTDO; HSPICS; UART0\_RTS
17. IO2 : GPIO2;UART1\_RXD
18. IO0 : GPIO0
19. IO4 : GPIO4
20. IO5 : GPIO5
21. RXD : UART0\_RXD; GPIO3
22. TXD : UART0\_TXD; GPIO1

## 2.5 **Lora Shield SX1272**

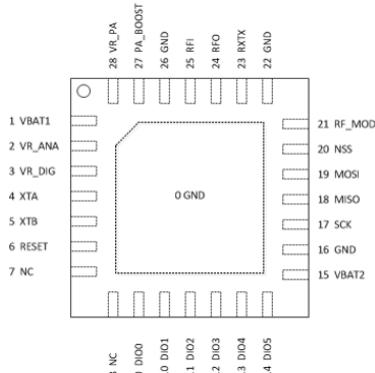
*LoRa Shield* adalah *transceiver* jarak jauh yang berbentuk *shield* yang berdasarkan *open source library*. *LoRa Shield* merupakan salah satu perangkat (*device*) yang dapat digunakan untuk melakukan komunikasi *wireless*. Perangkat ini dapat mengirimkan data hingga mencapai jarak yang jauh. Perangkat ini dapat diaplikasikan dalam jaringan sensor nirkabel seperti sistem irigasi, metering cerdas, *smart city*, *smartphone detection*, otomatisasi bangunan, dan sebagainya. Dalam tugas akhir ini menggunakan modul *LoRa Shield SX1272*. (*LoRa Shield*, n.d)



Gambar 2.9 LoRa *Shield*

(<https://www.seeedstudio.com/Dragino-LoRa-Shield-support-433MHz-frequency-p-2672.html> )

*Chip LoRa SX1272* memiliki 28 pin yang bekerja dengan mode transmisi *half duplex*.



Gambar 2.10 *Datasheet SX1272*

(<https://www.mouser.com/datasheet/2/761/sx1272-1277619.pdf> )

Perangkat ini memiliki kemampuan :

1. Kompatibel dengan papan arduino 3.3 Volt atau 5 Volt.
2. Frekuensi 915 MHz / 868 MHz / 433 MHz.
3. Konsumsi daya yang rendah.
4. Kompatibel dengan Arduino Leonardo, Uno, Mega.
5. Antena eksternal melalui konektor I-Pex

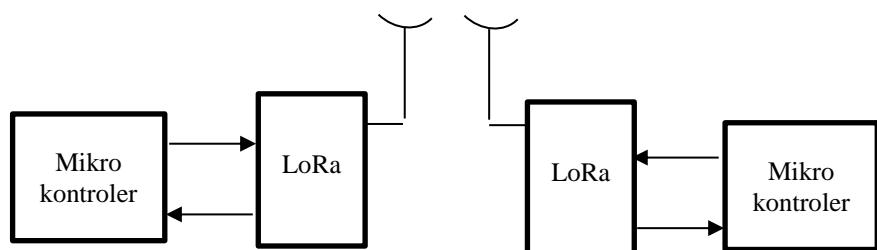
Sedangkan untuk spesifikasi dari *LoRa* adalah sebagai berikut :

1. 168 dB *maximum link budget*.
2. +20 dBm - 100 mW *constant RF output vs.*
3. +14 dBm *high efficiency PA*.
4. *Programmable bit rate up to 300 kbps*.
5. *High sensitivity: down to -148 dBm*.

6. Bullet-proof front end: IIP3 = -12.5 dBm.
7. Excellent blocking immunity.
8. Low RX current of 10.3 mA, 200 nA register retention.
9. Fully integrated synthesizer with a resolution of 61 Hz.
10. FSK, GFSK, MSK, GMSK, LoRaTM and OOK modulation.
11. Built-in bit synchronizer for clock recovery.
12. Preamble detection.
13. 127 dB Dynamic Range RSSI.
14. Automatic RF Sense and CAD with ultra-fast AFC.
15. Packet engine up to 256 bytes with CRC.
16. Built-in temperature sensor and low battery indicator[6].

### 2.5.1 Prinsip Kerja LoRa Shield

Modul LoRa dilengkapi dengan *Radio Frequency Transmitter* atau *Radio Frequency Receiver* yang berfungsi untuk komunikasi secara pengirim dengan penerima secara *half duplex*. *Half duplex* adalah komunikasi antara kedua pihak mengirimkan informasi atau menerima informasi secara dua arah yang saling bergantian. Pada prinsip kerjanya ketika mikrokontroller mengirimkan perintah pada modul LoRa Tx sebagai *transmitter* yang akan mengirimkan data ke modul LoRa Rx sebagai *receiver*. Modul LoRa yang digunakan pada perangkat *node* sel menggunakan frekuensi 433 MHz dan *node* koordinator menggunakan frekuensi 433 MHz dan 915 MHz. Pada saat transmisi data antara pengirim dan penerima, keduanya harus memiliki frekuensi yang sama.(Kiky, 2017) Prinsip kerja LoRa dapat dilihat pada gambar 2.5.3



Gambar 2.11 Prinsip kerja modul LoRa

( Dokumentasi penulis )

## 2.6 LoRa Bee

*LoRa Bee* merupakan perangkat *transceiver* jarak jauh yang berbentuk Bee berdasarkan *open source library*. *LoRa Bee* merupakan sebuah perangkat yang dapat digunakan untuk melakukan komunikasi *wireless*. Perangkat dapat mengirimkan data hingga mencapai jarak yang jauh. perangkat dapat di aplikasikan dalam jaringan sensor nirkabel seperti irigasi, metering cerdas, *smart city*, *smartphone*, *detection*, bangunan, dan lain-lain.

*LoRa Bee* menggunakan chip jenis SX1276 *transceiver* yang bekerja dengan *node* transmisi *half duplex* dengan frekuensi 915 MHz. pin diagram chip SX1276 dapat dilihat pada gambar dibawah.



Gambar 2.12 Pin Diagram *Chip* SX1276  
([https://wiki.dragino.com/index.php?title=Lora\\_BEE](https://wiki.dragino.com/index.php?title=Lora_BEE) )

Modul *LoRa Bee* memiliki kemampuan mendukung kebutuhan-kebutuhan *Wireless Sensor Network* dengan *Low-power*. Modul *LoRa Bee* dapat beroperasi pada frekuensi 433 MHz, 868MHz, dan 915 MHz dengan daya yang dibutuhkan sebesar 3,3 Volt. (*LoRa Bee*, n.d)



Gambar 2.13 Gambar LoRa Bee

([https://wiki.dragino.com/index.php?title=Lora\\_BEE](https://wiki.dragino.com/index.php?title=Lora_BEE) )

**Spesifikasi *Lora Bee* dari gambar 2.6.2 adalah :**

1. *Link budget* maximum 168 dB.
2. Tegangan *input* sebesar 3,3V.
3. RF *output* konstan +20 dBm sampai 100mW.
4. PA efisiensi tinggi +14dBm.
5. Kecepatan bit yang dapat diprogram sampai 300 Kbps.
6. Spesifikasi tinggi, terendah mencapai -148dBm.
7. Ujung depan anti peluru: IIP3 = -12,5 dBm.
8. *Blocking immunity* baik.
9. Arus RX rendah 10,3 mA, retensi register 200nA.
10. Terintegrasi *synchronizer* untuk pemulihan jam.
11. Modulasi FSK,GFSK, MSK, GMSK, *LoRa*™ dan OOK,
12. Dibangun dengan bit *synchronizer* untuk pemulihan jam.
13. *Preamble deection*.
14. 127 dB *Dynamic Range RSSI*.
15. *Automatic RF Sense* dan CAD dengan AFC ultra cepat.
16. Mesin paket hingga 256 byte dengan CRC.
17. Dirancang dengan sensor suhu dan indikator baterai rendah.
18. Dimensi 31 mm x 25 mm x 10 mm.
19. Kompatibel dengan Arduino.

#### **Dimensi dan Berat:**

1. Ukuran: 31mm \* 25mm \* 10mm.
2. Berat bersih: 4g.

#### **Fitur:**

Pita Frekuensi: 868 MHZ / 433 MHZ / 915 MHZ (Pra-konfigurasi di pabrik)

1. Modem *LoRa*™
2. FSK, GFSK, MSK, GMSK, *LoRa*™ dan modulasi OOK
3. XBee membentuk pabrik

## 2.7 Power Supply

*Power Supply* adalah suatu komponen yang berfungsi sebagai penyedia tegangan serta arus listrik untuk komponen lainnya yang telah terpasang. Arus listrik disalurkan oleh *power supply* adalah arus listrik bolak balik (AC). *Power supply* dapat juga berfungsi untuk mengubah arus AC menjadi arus DC. Fungsi utama *power supply* sebagai alat yang memberikan sebuah penyedia arus listrik untuk komponen yang sudah terpasang. (Stephan Adriansyah Hulukati dan Irvan A Salihi, 2018 )



Gambar 2.14 *Power Supply 12Volt*

( <https://www.lazada.co.id/products/adaptor-12v-3a-power-supply-switching-led-jaring-i1312240403.html> )



Gambar 2.15 *Power Supply 5Volt*

(<https://www.indiamart.com/proddetail/5v-5a-power-supply-17931879912.html>)

*AC to DC power supply* akan mengubah sumber tegangan listrik AC menjadi tegangan DC yang dibutuhkan oleh alat-alat elektronika. *AC to DC*

*power supply* biasanya mempunyai sebuah *transformator* yang digunakan untuk menurunkan tegangan dan mempunyai dioda sebagai penyuarah serta kapasitor sebagai *filter*.

**Spesifikasi Power Supply dari gambar 2.7.1 adalah :**

AC *input* : 110 – 240 V AC 50/60Hz

DC *output* : 3A 12V

Dimensi : 8,5\*3,5\*5,8 cm

*Power* : 36 W

**Spesifikasi Power Supply dari gambar 2.7.2 adalah :**

AC *input* : 110 – 240 V AC 50/60Hz

DC *output* : 3A 5V

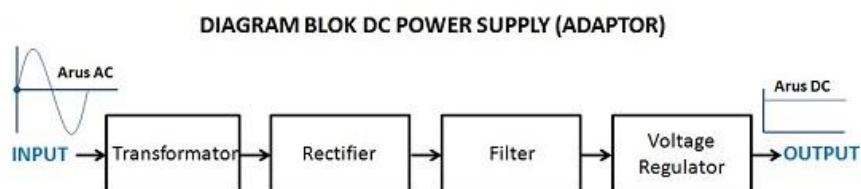
Dimensi : 8,5\*3,5\*5,8 cm

*Power* : 15 W

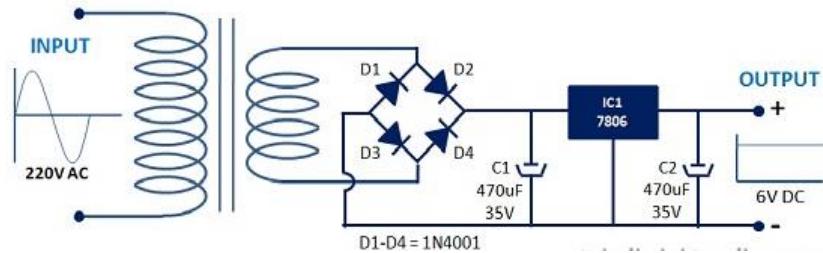
### 2.7.1 Prinsip Kerja Power Supply

*Power supply* berfungsi untuk menurunkan tegangan dan mengubah tegangan listrik dari sinyal AC (bolak balik) menjadi sinyal listrik DC (searah). Untuk menurunkan tegangan AC 220 V menjadi tegangan sebesar 5 V DC atau 12 V DC dibutuhkan *transformator step-down*.

Untuk mengubah bentuk gelombang sinyal AC ke DC diperlukan penyuarah (*rectifier*), penyaring (*filter*) dan regulasi (*regulator*). (erudisi,2021)



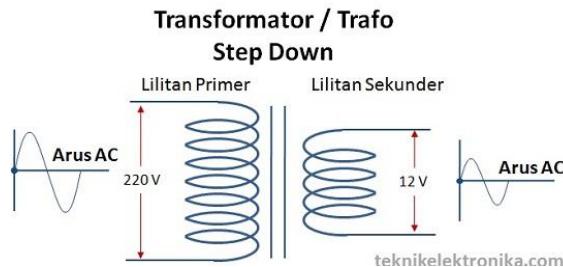
Gambar 2.16 Diagram blok DC *power supply*  
[\(https://teknikelektronika.com/prinsip-kerja-dc-power-supply-adaptor/\)](https://teknikelektronika.com/prinsip-kerja-dc-power-supply-adaptor/)



Gambar 2.17 Skematik rangkaian *power supply*  
[\(https://erudisi.com/rangkaian-power-supply/\)](https://erudisi.com/rangkaian-power-supply/)

### 2.7.1.1 *Transformer (Trafo)*

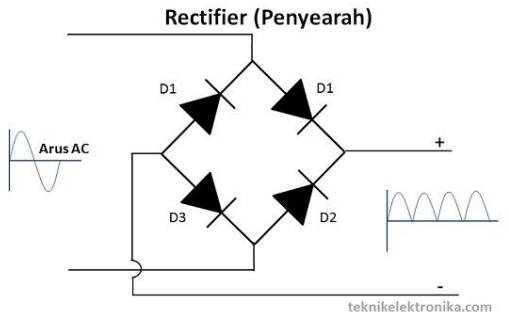
*Trafo* yang digunakan untuk DC *power supply* adalah *trafo step-down* yang berfungsi untuk menurunkan tegangan listrik yang terdapat pada rangkaian *adaptor* (DC *power supply*). *Trafo* bekerja berdasarkan lilitan primer dan sekunder. Lilitan primer merupakan *input* dari pada *trafo* sedangkan lilitan sekunder adalah *output*-nya. (teknikelektronika,2020)



Gambar 2.18 *trafo step-down*  
[\(https://teknikelektronika.com/prinsip-kerja-dc-power-supply-adaptor/ \)](https://teknikelektronika.com/prinsip-kerja-dc-power-supply-adaptor/)

### 2.7.1.2 *Rectifier*

Rectifier atau penyearah gelombang adalah rangkaian elektronika dalam *power supply* (catu daya) yang berfungsi untuk mengubah gelombang AC menjadi gelombang DC setelah tegangannya diturunkan oleh *Transformator Step down*. Rangkaian *rectifier* terdiri dari komponen 4 Dioda. (teknikelektronika,2020)

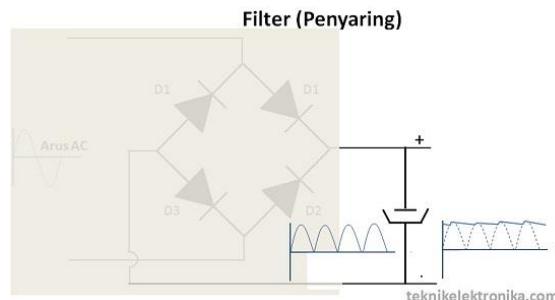


Gambar 2.19 *Rectifier*

(<https://teknikelektronika.com/prinsip-kerja-dc-power-supply-adaptor/> )

#### 2.7.1.3 *Filter (Penyaringan)*

Dalam rangkaian *Power supply (Adaptor)*, *Filter* digunakan untuk meratakan sinyal arus yang keluar dari *Rectifier*. *Filter* ini terdiri dari komponen Kapasitor (Kondensator) yang berjenis elektrolit atau ELCO (Electrolyte Capacitor). (teknikelektronika,2020)



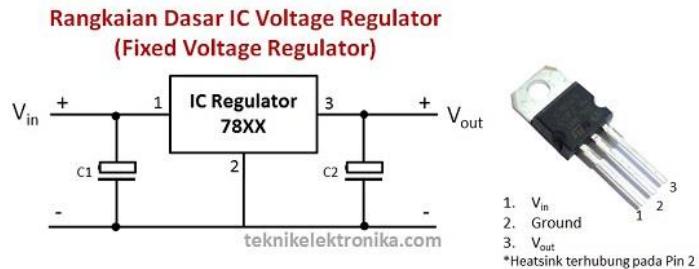
Gambar 2.20 *Filter*

(<https://teknikelektronika.com/prinsip-kerja-dc-power-supply-adaptor/> )

#### 2.7.1.4 *Voltage Regulator (Pengatur Tegangan)*

*Voltage regulator* berfungsi untuk mengatur tegangan sehingga tegangan dan arus DC tetap dan stabil. *Voltage regulator* terdiri dari Dioda Zener, Transistor atau IC (*Integrated Circuit*). *Voltage Regulator* juga terdapat *Short Circuit Protection* (perlindungan atas hubung singkat), *Current Limiting* (Pembatas Arus) ataupun *Over Voltage*

*Protection* (perlindungan atas kelebihan tegangan).  
(teknikelektronika,2020)'''



Gambar 2.21 Rangkaian *voltage regulator*  
(<https://teknikelektronika.com/prinsip-kerja-dc-power-supply-adaptor/> )

## 2.8 Relay

*Relay* adalah saklar (*switch*) yang dioperasikan secara listrik dan merupakan komponen elektromekanikal. *Relay* menggunakan prinsip elektromagnetik untuk menggerakkan kontak saklar sehingga dengan arus listrik yang kecil (*low power*) dapat menghantarkan listrik yang bertegangan lebih tinggi. (Muhamad Saleh dan Munnik Haryanti, 2017).



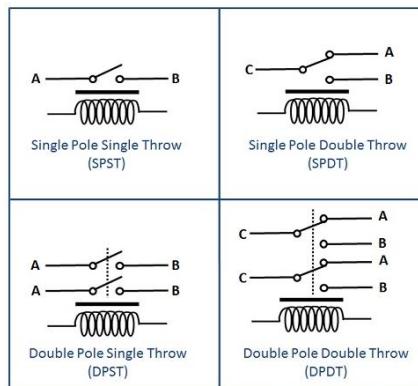
Gambar 2.22 Relay  
(<https://www.zanoor.com/pengertian-relay/> )

Karena *relay* merupakan salah satu jenis saklar, maka istilah *pole* dan *throw* yang dipakai dalam saklar juga berlaku pada *relay*. *Pole* merupakan banyaknya kontak yang dimiliki oleh sebuah *relay*. Sedangkan *throw* merupakan banyaknya kondisi yang dimiliki oleh sebuah kontak. Berdasarkan penggolongan jumlah *pole* dan *throw*-nya sebuah *relay*, maka *relay* dapat digolongkan menjadi :

- a. *Single Pole Single Throw* (SPST) : *Relay* golongan ini memiliki 4 terminal, 2 terminal untuk saklar dan 2 terminalnya lagi untuk *coil*.

- b. *Single Pole Double Throw (SPDT)* : Relay golongan ini memiliki 5 terminal, 3 terminal untuk saklar dan 2 terminalnya lagi untuk *coil*.
- c. *Double Pole Single Throw (DPST)* : Relay golongan ini memiliki 6 terminal, diantaranya 4 terminal yang terdiri dari 2 pasang terminal saklar sedangkan 2 terminal lainnya untuk *coil*. Relay DPST dapat dijadikan 2 saklar yang dikendalikan oleh 1 *coil*.
- d. *Double Pole Double Throw (DPDT)* : Relay golongan ini memiliki terminal sebanyak 8 terminal, diantaranya 6 terminal yang merupakan 2 pasang relay SPDT yang dikendalikan oleh 1 (*single*) *coil*. Sedangkan 2 terminal lainnya untuk *coil*. (Muhammad Saleh dan Munnik Haryanti, 2017)

Adapun penggolongan jumlah *pole* dan *throw* ditunjukan pada gambar 2.8.2



Gambar 2.23 Jenis *relay* berdasarkan *pole* dan *throw*  
[\( https://teknikelektronika.com/pengertian-relay-fungsi-relay/ \)](https://teknikelektronika.com/pengertian-relay-fungsi-relay/)

### 2.8.1 Prinsip Kerja *Relay*

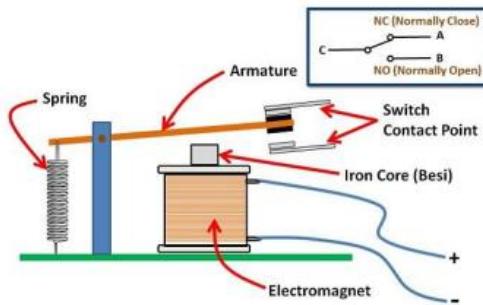
*Relay* menggunakan prinsip elektromagnetik untuk menggerakkan kontak saklar sehingga arus listrik yang kecil (*low power*) dapat menghantarkan listrik yang bertegangan lebih tinggi. Sebagai contoh, dengan *relay* yang menggunakan electromagnet 5V dan 50mA mampu menggerakan *amature relay* (yang berfungsi sebagai saklar) untuk mengantarkan listrik 220V 2A.

Pada dasarnya, *relay* terdiri dari 4 komponen dasar yaitu:

- *Electromagnet (Coil)*

- *Armature*
- *Switch Contact Point (Saklar)*
- *Spring*

Berikut ini merupakan gambar dari bagian-bagian *relay*:



Gambar 2.24 Struktur sederhana *relay*

( Sumber : <http://teknikelektronika.com/pengertian-relay-fungsi-relay/>)

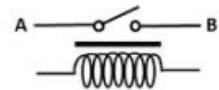
Kontak point *relay* terdiri dari 2 jenis yaitu:

- *Normally Close (NC)* yaitu kondisi awal sebelum diaktifkan selalu berada di posisi *close* (tertutup)
- *Normally Open (NO)* yaitu kondisi awal sebelum diaktifkan akan selalu berada di posisi *open* (terbuka)

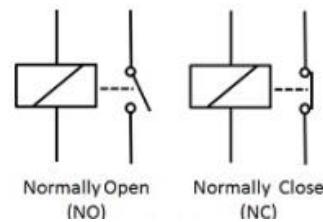
Berdasarkan gambar diatas, sebuah Besi (*Iron Core*) yang dililit oleh sebuah kumparan *Coil* yang berfungsi untuk mengendalikan Besi tersebut. Apabila Kumparan *Coil* diberikan arus listrik, maka akan timbul gaya Elektromagnet yang kemudian menarik *Armature* untuk berpindah dari Posisi sebelumnya (NC) ke posisi baru (NO) sehingga menjadi Saklar yang dapat menghantarkan arus listrik di posisi barunya (NO). Posisi dimana *armature* tersebut berada sebelumnya (NC) akan menjadi open atau tidak terhubung. Pada saat tidak dialiri arus listrik, *Armature* akan kembali lagi ke posisi awal (NC). *Coil* yang digunakan oleh *Relay* untuk menarik *contact point* ke posisi *close* pada umumnya hanya membutuhkan arus listrik yang relatif kecil. (MR Pahlevi, 2015)

Adapun simbol *relay* *Normally Close* (NC) dan *Normally Open* (NO) ditunjukan pada gambar 2.8.4.

**Simbol Relay**



atau



Gambar 2.25 Struktur dasar *relay*

( <https://teknikelektronika.com/pengertian-relay-fungsi-relay/> )

## **BAB III**

### **PERANCANGAN DAN PEMBUATAN SISTEM**

Perancangan dan pembuatan sistem rancang bangun model monitoring dan pengendalian air dilaksanakan melalui beberapa tahapan yang dilakukan secara sistematis. Adapun tahapan-tahapan perancangan dan pembuatan sistem meliputi, perencanaan pembuatan sistem, perancangan sistem seperti perancangan rancang bangun, perancangan kebutuhan alat berupa kebutuan *hardware* (perangkat keras) dan *software* (perangkat lunak), perancangan algoritma sistem dan perancangan *prototype* sistem, dan pembuatan sistem seperti pembuatan sistem *software* (perangkat lunak) dan pembuatan *prototype* sistem.

#### **3.1 Perencanaan Pembuatan Sistem**

Perencanaan pembuatan sistem adalah tahapan yang dilakukan untuk merancang sistem dengan sensor-sensor yang digunakan pada perancangan alat di masing-masing *node* sel, *node* koordinator dan *gateway*. Perencanaan pembuatan sistem ini bertujuan untuk mengetahui rencana yang dibuat dalam perancangan sistem model monitoring dan pengendalian, seperti monitoring volume dan debit air, pengendalian kebocoran pada sistem dan perencanaan sel-sel yang dibutuhkan dalam pembuatan alat.

Perencanaan pembuatan sistem monitoring dan pengendalian air ini membuat 5 *node* sel yang terdiri dari 4 sel gedung dan 1 sel pusat, 1 *node* koordinator dan 1 *gateway*. Pada masing-masing *node* sel terdiri dari sensor *flowmeter* dan sensor selenoid valve. Sensor *flowmeter* digunakan untuk menghitung debit air yang digunakan pada masing-masing sel yang dikirimkan ke *database* yang diolah menjadi data debir air yang keluar dan volume air yang digunakan pada masing-masing *node* sel kemudian akan ditampilkan atau di monitoring melalui web dan andorid. Sensor selenoid valve digunakan untuk melakukan pengendalian dengan cara membuka katub agar air mengalir dan menutup katub agar air tidak mengalir jika terjadi kebocoran pada sistem alat yang dibuat. Sensor *flowmeter* dan selenoid valve dipasang di masing-masing *node* sel untuk melakukan

monitoring dan pengendalian. Mikrokontroller yang digunakan pada masing-masing *node* sel adalah Arduino Uno dan *device* komunikasi LoRa *Shield* 433 MHz.

Perencanaan *Node* koordinator menggunakan 2 buah mikrokontroller Arduino Uno dan 1 buah LoRa *Shield* 433 MHz dan 1 buah LoRa *Shield* 915 MHz yang akan difungsikan sebagai *repeater* pengiriman dan penerimaan data dari masing-masing sel ke *gateway* dan sebaliknya. Perencanaan *gateway* menggunakan mikrokontroler *NodeMCU* ESP8266 dan LoRa Bee 915 MHz. *NodeMCU* ESP8266 berfungsi sebagai *platform IoT* yang digunakan untuk mengkoneksikan antara data *hardware* ke *database* dengan menggunakan jaringan WiFi. LoRa Bee 915 MHz difungsikan sebagai pengiriman dan penerimaan data dari *nodeMCU* ESP8266 ke *database*.

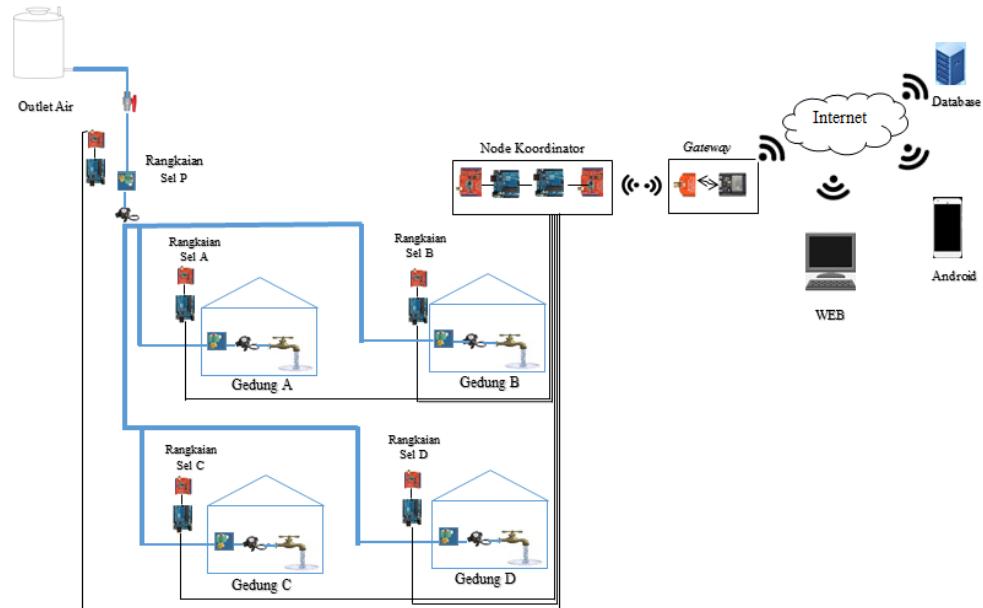
Kebocoran sistem akan dirancang agar ketika terjadi kebocoran, *user* mendapat notifikasi dimana terjadi kebocoran. Perencanaan kebocoran ini menggunakan *platform Telegram* untuk mendapatkan notifikasi kebocoran pada sel. Jika terjadi kebocoran di sel gedung A maka sensor akan mengirim data yang kemudian diproses di *gateway* untuk mengirim notifikasi ke *platform Telegram*. Kemudian akan dikendalikan untuk membuka katub atau mematikan air dan menutup katub atau mematikan air dengan menggunakan web atau android.

### 3.2 Perancangan Sistem

Perancangan sistem adalah tahapan yang digunakan untuk membuat sistem alat dengan alat-alat yang telah ditentukan di perencanaan pembuatan sistem. Perancangan sistem bertujuan untuk menentukan desain alat yang akan dibuat, kuantitas bahan yang dibutuhkan dan cara kerja sistem alat dengan menggunakan *flowchart*. Perancangan sistem meliputi :

### 3.2.1 Rancang Bangun Sistem

Sistem rancang bangun model monitoring dan pengendalian air ini dirancang dengan layout ditunjukkan gambar 3.1



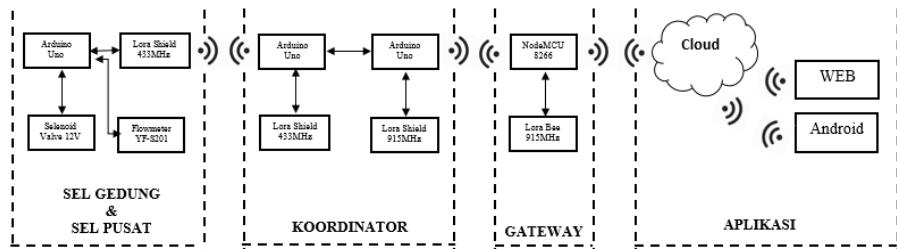
Gambar 3.1 Layout sistem air

Sumber: Dokumentasi Penulis

Pada pusat dan masing-masing gedung terdiri dari 1 sel monitoring dan pengendalian. Sel tersebut terdapat Arduino Uno dan LoRa *Shield* pada masing-masing bagian. Arduino Uno berfungsi sebagai mikrokontroler dan LoRa *Shield* berfungsi untuk menerima data dari dan ke *node* koordinator. Data yang diperoleh Lora *Shield* dari *node* koordinator akan diproses oleh Arduino Uno sehingga dapat mengendalikan nyala air. Dan data yang dikirim LoRa *Shield* ke *node* koordinator untuk dapat memonitoring air.

Pada saat sistem mengirim data berupa informasi menggunakan sensor *flowmeter* ke *node* koordinator yang selanjutnya dikirim ke *gateway* akan diproses oleh database untuk memonitor penggunaan air pada *platform* web. Pada saat sistem menerima data berupa perintah untuk mematikan dan menghidupkan selenoid valve dari *platform* web, maka katup selenoid akan menutup yang mengakibatkan air dalam keadaan off. Kemudian jika *gateway*

mendeteksi kebocoran air maka *gateway* akan mengirim notifikasi terdapat kebocoran air pada gedung yang mengirim data melebihi batas wajar penggunaan. Data tersebut diambil dari data sensor *flowmeter* yang diambil dari pusat dan masing-masing sel gedung melalui *node* koordinator. Adapun alur kerja dari sistem monitoring dan pengendalian air sebagaimana ditunjukkan pada gambar 3.2.



Gambar 3.2 Blok diagram sistem air dalam *smart building*

Sumber: Dokumentasi Penulis

Pada tugas akhir ini dibuat 1 sel pusat dan 4 sel gedung. Masing-masing sel gedung dan sel pusat terdiri dari 1 buah mikrokontroler Arduino Uno , 1 buah LoRa *Shield* 433Mhz, 1 buah sensor *flowmeter* dan 1 buah selenoid valve. Sensor *flowmeter* dihubungkan dengan sumber tegangan *power supply* 5V DC, dan data PIN 3 pada mikrokontroler dan Lora *Shield*. Sedangkan selenoid valve dihubungkan dengan sumber tegangan *power supply* 12V DC dengan menggunakan 1 buah *relay* SPDT. Relay dihubungkan dengan mikrokontroler dan LoRa *Shield* Vcc 5V, GND, dan data PIN 4 dengan *contact point normally close*. Selenoid valve menggunakan tipe *normally open* jika tidak ada tegangan katup selenoid akan membuka dan ketika diberi tegangan katup selenoid akan menutup. Data yang diperoleh dari *flowmeter* akan diproses oleh mikrokontroler Arduino Uno, Data yang telah diproses dikirimkan LoRa *Shield* 433 MHz ke *node* koordinator 433 MHz. Kemudian ketika sel pusat atau gedung menerima data dari *node* koordinator 433 MHz berupa data perintah akan diproses oleh mikrokontroler Arduino Uno yang diteruskan ke selenoid valve.

*Node* koordinator berfungsi sebagai repeater. *Node* koordinator terdiri dari 2 buah mikrokontroler Arduino Uno dan 2 buah LoRa *Shield* dengan frekuensi 433 Mhz dan frekuensi 915 MHz. Pada masing-masing *Node* koordinator dihubungkan dengan komunikasi serial. PIN 0 *node* frekuensi 433MHz dihubungkan dengan PIN 1 *node* frekuensi 915 MHz. PIN 1 *node* frekuensi 433 MHz dihubungkan dengan PIN 0 *node* frekuensi 915 MHz dan GND *node* frekuensi 433 MHz dihubungkan GND *node* frekuensi 915 MHz. Pin-pin tersebut dihubungkan dengan menggunakan jumper. Data-data yang dikirimkan ke sel pusat dan gedung dan yang dikirimkan ke *gateway* dengan bergantian. Komunikasi yang digunakan menggunakan komunikasi TDM (*Time Division Multiplexing*). Ketika LoRa *Shield* 433 Mhz menerima data dari sel gedung atau sel pusat, data akan diproses oleh mikrokontroler Arduino Uno. Data yang sudah diproses oleh mikrokontroler akan diteruskan menggunakan komunikasi serial ke LoRa *Shield* 915 MHz dan diteruskan ke *gateway*. Jika LoRa *Shield* 915 MHz menerima data dari *gateway*, data akan diproses oleh mikrokontroler Arduino Uno. Data yang sudah diproses oleh mikrokontroler akan diteruskan menggunakan komunikasi serial ke LoRa *Shield* 433 MHz ke sel pusat atau sel gedung. *Node* koordinator dihubungkan dengan *gateway* dengan menggunakan gelombang radio.

*Gateway* terdiri dari *NodeMCU* ESP8266 dan LoRa Bee 915 Mhz. *Gateway* berfungsi sebagai komunikasi dengan *Web Server* dengan menggunakan internet. *NodeMCU* ESP8266 adalah perangkat yang digunakan untuk mengambil dan menerima data *Web Server*. Ketika *gateway* menerima data dari *node* koordinator, *NodeMCU* ESP8266 akan mengirimkan data ke *database Web Server*. Jika *gateway* menerima data berupa perintah dari *Web Server*, LoRa Bee 915 MHz akan mengirimkan data tersebut ke *node* koordinator dengan menggunakan media transmisi gelombang radio yang kemudian diteruskan ke sel pusat atau sel gedung.

Data yang dikirim dari sel pusat dan sel gedung ke *node* koordinator berupa data *rate/debit*. *Rate/debit* air adalah data rata-rata air yang digunakan dalam satuan L/menit. Data tersebut dikirimkan ke *node* koordinator berupa data *realtime* yang dihitung oleh *flowmeter*. Data *flowmeter* yang diterima *node* koordinator akan diteruskan *gateway* kemudian *gateway* diteruskan ke *database Web Server*. Data *flowmeter* ini yang ditampilkan untuk digunakan untuk data monitoring.

Data yang dikirim *database Web Server* ke *gateway* berupa perintah *on/off* diproses mikrokontroler *NodeMCU ESP8266* yang dikirimkan ke *node* koordinator 915MHz oleh LoRa Bee 915 MHz. Data yang telah diproses dikirimkan LoRa *Shield* 915 MHz ke LoRa *Shield* 433 MHz. Data yang diterima LoRa *Shield* 433MHz diproses mikrokontroler Arduino Uno. Jika data sudah diproses Lora *Shield* 433 MHz ke tujuan data ke sel pusat atau sel gedung yang dituju. Kemudian data akan diterima oleh LoRa *Shield* 433 MHz pada sel pusat atau sel gedung. Data yang diterima akan diproses Arduino Uno yang kemudian data tersebut digunakan untuk memberikan perintah ke perangkat solenoid valve untuk menghidupkan atau mematikan katub selenoid valve.

Pengendalian sistem ini di identifikasi ketika air yang keluar diluar batas pemakaian dengan sistem waktu. *Platform* yang digunakan adalah Telegram. Pada sistem ini teridentifikasi mengalami kebocoran ketika air yang mengalir pada setiap sel gedung lebih dari 10 menit. Sel gedung akan mengirimkan data jika terjadi kebocoran ke *node* koordinator yang kemudian akan dikirimkan ke *gateway*. *Gateway* tersambung ke ID telegram untuk mengirimkan notifikasi kebocoran pada sel gedung tertentu. *Gateway* mengirimkan data dengan menggunakan *NodeMCU ESP8266* yang terkoneksi ke internet. Jadi ketika data *flowmeter* gedung mendeteksi air mengalir dalam waktu lebih dari 10 menit. Telegram mendapatkan pesan “Terjadi Kebocoran pada Gedung ‘A/B/C/D’ ”. Ketika Telegram

mendapatkan notifikasi, *user* akan mengoperasikan *platform Web Server* untuk mengirimkan perintah ke sel gedung untuk mematikan selenoid agar air tidak mengalir katub selenoid ditutup. Jika katub selenoid sudah tertutup maka gedung kembali mengirim data berupa informasi ke *node* koordinator yang diteruskan ke *gateway*. *Gateway* akan mengirim data informasi tersebut ke *platform Telegram* untuk mengkonfirmasi bahwa sel gedung yang mengalami kebocoran sudah diperbaiki. *Platform Telegram* mendapatkan pesan berupa “Selenoid Gedung ‘A/B/C/D/Pusat’ Off (Air Mati) yang berarti katub selenoid pada sel yang terjadi kebocoran telah ditutup.

Data yang dikirim sel pusat dan sel gedung ke *node* koordinator dan *gateway* berupa data asal gedung, data *flowmeter*, data notifikasi, dan data pengendalian yang digunakan sebagai data monitoring. Data yang diterima *node* koordinator dan *gateway* merupakan data 4bit dengan bit ke-1 berisi asal gedung(A,B,C,D,P), bit ke 2 berisi data perhitungan *flowmeter*, bit ke-3 berisi data notifikasi yang terdiri dari angka 0 atau 1, angka 0 menandakan tidak terjadi kebocoran air dan angka 1 menandakan terjadinya kebocoran air. Sedangkan bit ke-4 berisi data pengendalian yang terdiri dari angka 1 atau 2, angka 1 menandakan kondisi selenoid dalam keadaan *ON*, air dapat mengalir. Sedangkan angka 2 menandakan kondisi selenoid dalam keadaan *OFF*, air tidak mengalir. Format data yang dikirim oleh gedung dan diterima oleh *node* koordinator dan *gateway* ditunjukkan pada tabel 3.1 sampai 3.4 berikut:

Tabel 3.1 Format data yang dikirim sel pusat dan gedung ke koordinator 433Mhz

Nama Sel	Data yang dikirim ke Koordinator 433MHz
Gedung Pusat	Sending Rate = 4.20 Liter/Menit Sending Notif = 0 Sending Data Pengendalian = 1
Gedung A	Sending Rate = 1.12 Liter/Menit Sending Notif = 1 Sending Data Pengendalian = 1
Gedung B	Sending Rate = 0.98 Liter/Menit Sending Notif = 0 Sending Data Pengendalian = 1
Gedung C	Sending Rate = 0.00 Liter/Menit Sending Notif = 0 Sending Data Pengendalian = 2
Gedung D	Sending Rate = 1.56 Liter/Menit Sending Notif = 0 Sending Data Pengendalian = 1

Tabel 3.2 Format data yang diterima Koordinator 433MHz

Data yang diterima dari sel pusat dan gedung	Asal gedung	Data <i>flowmeter</i>	Data Notif	Data Pengendalian
	Bit 1	Bit 2	Bit 3	Bit 4
P 4.20 0 1	P	4.20	0	1
A 1.12 1 1	A	1.12	1	1
B 0.98 0 1	B	0.98	0	1
C 0.00 0 2	C	0.00	0	2
D 1.56 0 1	D	1.56	0	1

Tabel 3.3 Format data yang diterima Koordinator 915 MHz

Data yang diterima dari Koordinator 915 Mhz	Asal gedung	Data <i>flowmeter</i>	Data Notif	Data Pengendalian
	Bit 1	Bit 2	Bit 3	Bit 4
P 4.20 0 1	P	4.20	0	1
A 1.12 1 1	A	1.12	1	1
B 0.98 0 1	B	0.98	0	1
C 0.00 0 2	C	0.00	0	2
D 1.56 0 1	D	1.56	0	1

Tabel 3.4 Format data yang dikirim koordinator 915Mhz ke *gateway*

Data yang dikirim koordinator 915Mhz	Data yang diterima <i>gateway</i>
P 4.20 0 1	Asal gedung = P Rate gedung = 4.20 Liter/Menit Data notifikasi = 0 Data pengendalian = 1

A 1.12 1 1	Asal gedung = A Rate gedung = 1.12 Liter/Menit Data notifikasi = 1 Data pengendalian = 1
B 0.98 0 1	Asal gedung = B Rate gedung = 0.98 Liter/Menit Data notifikasi = 0 Data pengendalian = 1
C 0.00 0 2	Asal gedung = C Rate gedung = 0.00 Liter/Menit Data notifikasi = 0 Data pengendalian = 2
D 1.56 0 1	Asal gedung = D Rate gedung = 1.56 Liter/Menit Data notifikasi = 0 Data pengendalian = 1

Data yang dikirim *gateway* ke *node* koordinator, sel pusat dan gedung berupa data pengendalian yang digunakan untuk mengendalikan katup selenoid pada gedung jika terjadi kebocoran. Data yang dikirim berupa data 2bit dengan bit ke-1 merupakan data perintah yang terdiri dari angka 1 atau 2, angka 1 berfungsi untuk membuka katup selenoid agar air dapat mengalir atau dalam posisi ON, sedangkan angka 2 berfungsi untuk menutup katup selenoid agar air tidak mengalir atau dalam posisi OFF. Sedangkan bit ke-2 merupakan data gedung tujuan, pada *gateway* data gedung tujuan berupa angka 1,2,3,4, atau 5. Selanjutnya pada koordinator 915Mhz data gedung tujuan tersebut dikonversi menjadi huruf agar dapat dieksekusi dan diterima pada tiap gedungnya. Angka 1 dikonversi menjadi huruf A (gedung A), angka 2 dikonversi menjadi huruf B (gedung B), angka 3 dikonversi menjadi huruf C (gedung C), angka 4 dikonversi menjadi huruf D (gedung D), angka 5 dikonversi

menjadi huruf P (gedung Pusat). Selanjutnya data tersebut diteruskan ke koordinator 433MHz melalui komunikasi serial dan dikirimkan ke masing-masing gedung tujuan. Format data yang dikirim *gateway* ke *node* koordinator dan diteruskan ke masing-masing gedung ditunjukkan pada tabel berikut:

Tabel 3.5 Format data pengendalian yang dikirim *gateway* ke koordinator 915Mhz

Data yang dikirim <i>gateway</i> ke koordinator 915Mhz	Data yang diterima koordinator 915Mhz	
	Bit 1	Bit 2
Perintah 1/0 = 1	1	A
Gedung = 1		
Perintah 1/0 = 1	1	B
Gedung = 2		
Perintah 1/0 = 2	2	C
Gedung = 3		
Perintah 1/0 = 1	1	D
Gedung = 4		
Perintah 1/0 = 1	1	P
Gedung = 5		

Tabel 3.6 Format data pengendalian yang dikirim koordinator 915Mhz ke koordinator 433 Mhz

Data yang dikirim koordinator 915Mhz ke Koordinator 433Mhz		Data yang diterima koordinator 433Mhz	
Bit 1	Bit 2	Bit 1	Bit 2
1	A	1	A
1	B	1	B
2	C	2	C
1	D	1	D
1	P	1	P

Tabel 3.7 Format data pengendalian yang dikirim koordinator 433Mhz ke masing-masing gedung

Data yang dikirim koordinator 433Mhz ke gedung	Data yang diterima gedung				
	Gedung A	Gedung B	Gedung C	Gedung D	Gedung P
1A 1B 2C 1D 1P	1	1	2	1	1

### 3.2.2 Kebutuhan Alat *Hardware* (Perangkat Keras)

*Hardware* (perangkat keras) yang dibutuhkan dalam perancangan dalam pembuatan rancang bangun model monitoring dan pengendalian air antara lain mikrokontroler Arduino Uno, *NodeMCU ESP8266*, LoRa *Shield* 433MHz, LoRa *Shield* 915MHz, LoRa Bee 915MHZ, *flowmeter* YF-S201, selenoid valve 12V DC, *relay*, *power supply* DC 12V, *power supply* DC 5V dan peralatan tambahan seperti PCB, kabel, jumper, terminal strip, *pin header* dan peralon. Adapun spesifikasi dari perangkat-perangkat tersebut dapat dilihat pada tabel 3.1 berikut:

Tabel 3.8 Spesifikasi *Hardware* (perangkat keras)

No.	Nama Perangkat	Spesifikasi	
1.	Arduino Uno	Tegangan pengoperasian	5V
		Tegangan input	7-12V
		Pin I/O Digital	14 Pin (6 Pin untuk PWM)
		Pin I/O Analog	6 Pin
		<i>Flash Memory</i>	32 KB

		SRAM	2 KB
		EEROM	1 KB
		<i>Clock speed</i>	16 MHz
2.	<i>Lora Shield</i>	<i>Link budget max</i>	168 dB
		<i>Dynamic RSSI</i>	127 dB
		<i>High sensitivity</i>	<i>down to -148 dBm</i>
3.	<i>Lora Bee</i>	<i>Link budget max</i>	168 dB
		<i>Dynamic RSSI</i>	127 dB
		<i>High sensitivity</i>	<i>down to -148 dBm</i>
4.	<i>Solenoid Valve</i> 12V	<i>AC input</i>	110-240VAC 50/60Hz
		<i>DC output</i>	3A 12V
		<i>Power</i>	36 Watt
5.	<i>Flowmeter</i>	Tegangan	5 to 18V DC
		<i>Flow Rate</i>	1 to 30 <i>Liters/Minute</i>
		<i>Maximum water pressure</i>	2.0 MPa
6.	<i>Relay</i>	Tipe	<i>Singe Pole Double Throw (SPDT)</i>
		Tegangan input	5V
		Arus maksimum	10A
7.	<i>Power Supply</i> 12V	<i>AC input</i>	110-240 v AC 50/60Hz
		<i>DC output</i>	12V
		Arus	3A
8.	<i>Power Supply</i> 5V	<i>AC input</i>	110-240 v AC 50/60Hz
		<i>DC output</i>	5V
		Arus	3A

### 3.2.3 Kebutuhan Alat *Software* (Perangkat Lunak)

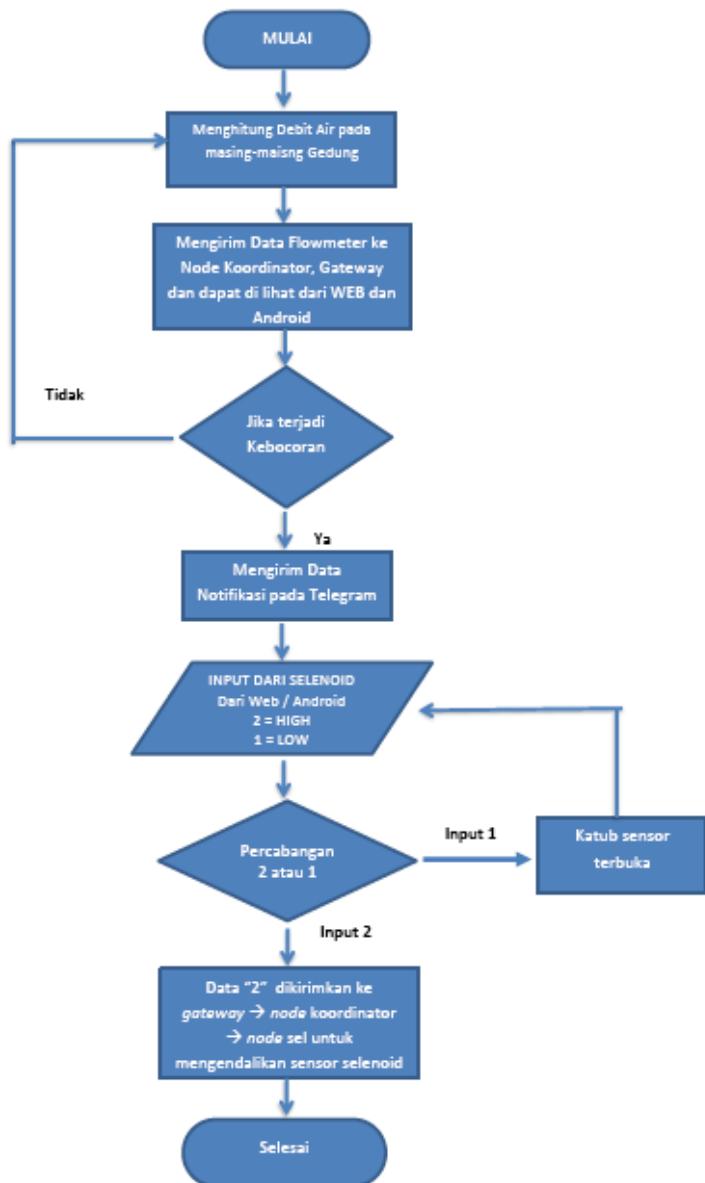
Berikut ini adalah *software* yang digunakan dalam pembuatan rancang bangun model monitoring dan pengendalian air :

1. Arduino IDE 1.8.13

Arduino IDE 1.8.13 adalah aplikasi yang digunakan dalam membuat program mikrokontroller Arduino Uno dengan LoRa *Shield* 433MHz dan 915Mhz dan *NodeMCU* ESP8266 dengan LoRa Bee 915MHz. Aplikasi ini dapat diunduh secara gratis pada halaman *website* resmi arduino di [www.arduino.cc](http://www.arduino.cc).

### 3.2.4 Perancangan Algoritma Sistem

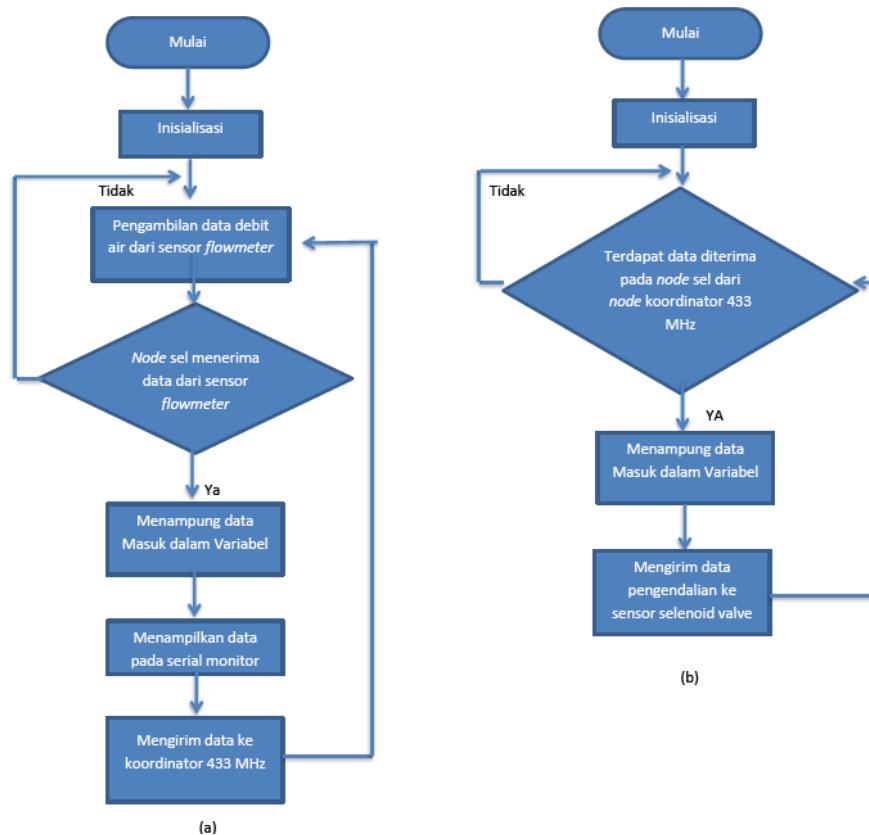
Perancangan algoritma sistem ini bertujuan untuk mengetahui kebutuhan atau alur kerja dari sistem rancang bangun model monitoring dan pengendalian air. Alur kerja sistem rancang bangun ini menggunakan *flowchart* atau diagram alir. *Flowchart* adalah suatu bagan yang berisikan simbol-simbol tertentu yang menggambarkan urutan suatu proses secara detail dan berhubungan antara suatu proses (perintah) dengan proses lainnya dalam suatu program. *Flowchart* rancang bangun model monitoring dan pengendalian air dijelaskan sebagai berikut :



Gambar 3.3 Flowchart perancangan sistem monitoring dan pengendalian air

Pada *flowchart* gambar 3.3 menjelaskan tentang alur kerja sistem monitoring dan pengendalian air pada sistem *smart building* berbasis IoT. Pada alur diatas air yang mengalir dihitung debitnya pada masing-masing *node* sel dengan menggunakan sensor *flowmeter*. Debit air yang dihitung dibaca oleh mikrokontroller masing-masing *node* sel yang kemudian dikirimkan ke *node* koordinator 433 MHz dan *node* koodinator 915 MHz. Pada *node* koordinator 915 MHz

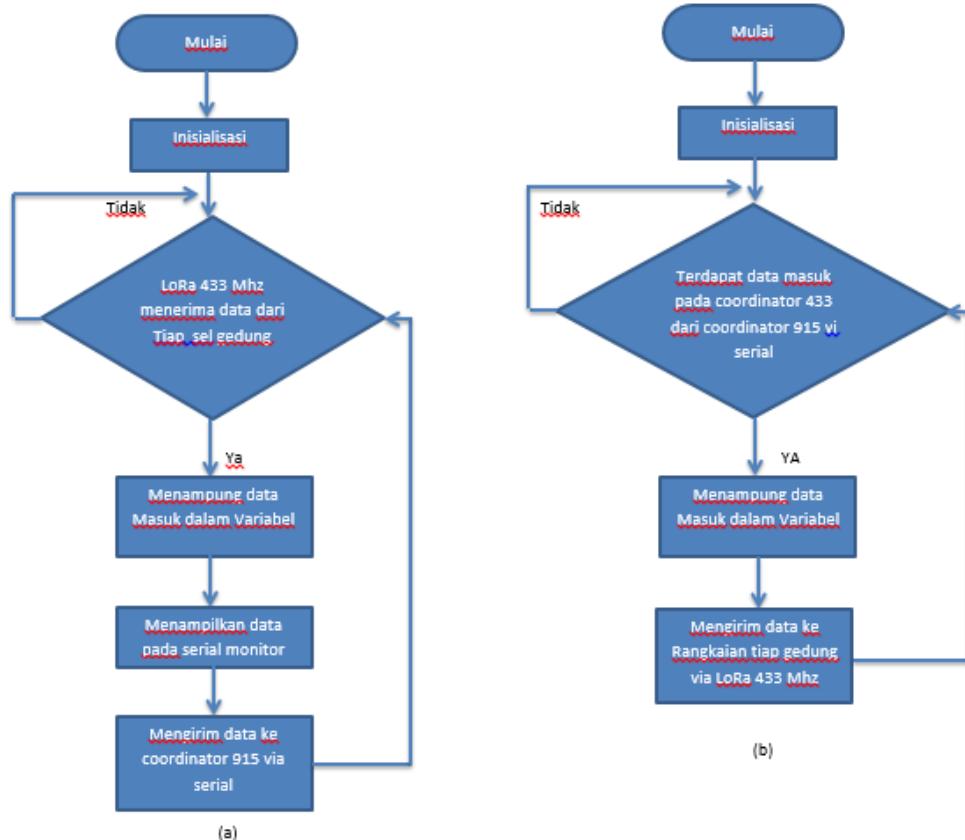
data tersebut akan diproses kemudian data dikirimkan ke *gateway* dan dikirimkan ke *database* yang diolah lalu ditampilkan pada sistem web dan android. Jika data yang dikirimkan teridentifikasi terjadi kebocoran pada *node sel* yaitu jika data sensor *flowmeter* yang dikirimkan *node sel* melebihi waktu 10 menit, maka *gateway* mengirim notifikasi kebocoran pada *platform Telegram*. Ketika terjadi kebocoran pada *node sel user* mengirim perintah “2” untuk menutup katub selenoid dengan cara menonaktifkan dengan web atau android. Data tersebut akan dikirimkan ke *gateway* kemudian ke *node koordinator* dan *node sel*. *Node sel* akan mengirimkan perintah ke sensor selenoid valve untuk menutup katub selenoid supaya air tidak dapat mengalir. Jika data 1, maka katub sensor tersebut dalam keadaan terbuka atau alir dapat mengalir.



Gambar 3.4 Flowchart node sel

- Program Monitoring
- Program Kontroling

Pada *flowchart* gambar 3.4 menjelaskan tentang program monitoring dan pengendalian yang berada di *node* sel. Pada program monitoring diatas data debit air diambil dari sensor *flowmeter* yang berada di masing-masing *node* sel. Kemudian data tersebut diterima oleh mikrokontroller yang berada di *node* sel. Jika data debit air tidak diterima oleh mikrokontroller yang berada di *node* sel maka program akan kembali ke inisialisasi awal. Akan tetapi jika data debit diterima, data tersebut akan ditampung oleh mikrokontroller dalam bentuk variable. Kemudian data ditampilkan di masing-masing serial monitor yang berada di *node* sel. Jika data sudah ditampilkan dan diterima pada mikrokontroller *node* sel maka data akan dikirimkan ke *node* koordinator untuk selanjutnya diproses lebih lanjut. Pada program kontroling diatas data pengendalian dikirim dari *node* koordinator 433 MHz dan diterima oleh masing-masing *node* sel. Jika data tidak diterima, maka akan kembali ke inisialisasi awal. Data yang diterima *node* sel akan ditampung dalam variable dan kemudian *node* sel mengirimkan data untuk mengendalikan sensor selenoid valve sesuai dengan perintah data yang dikirimkan.

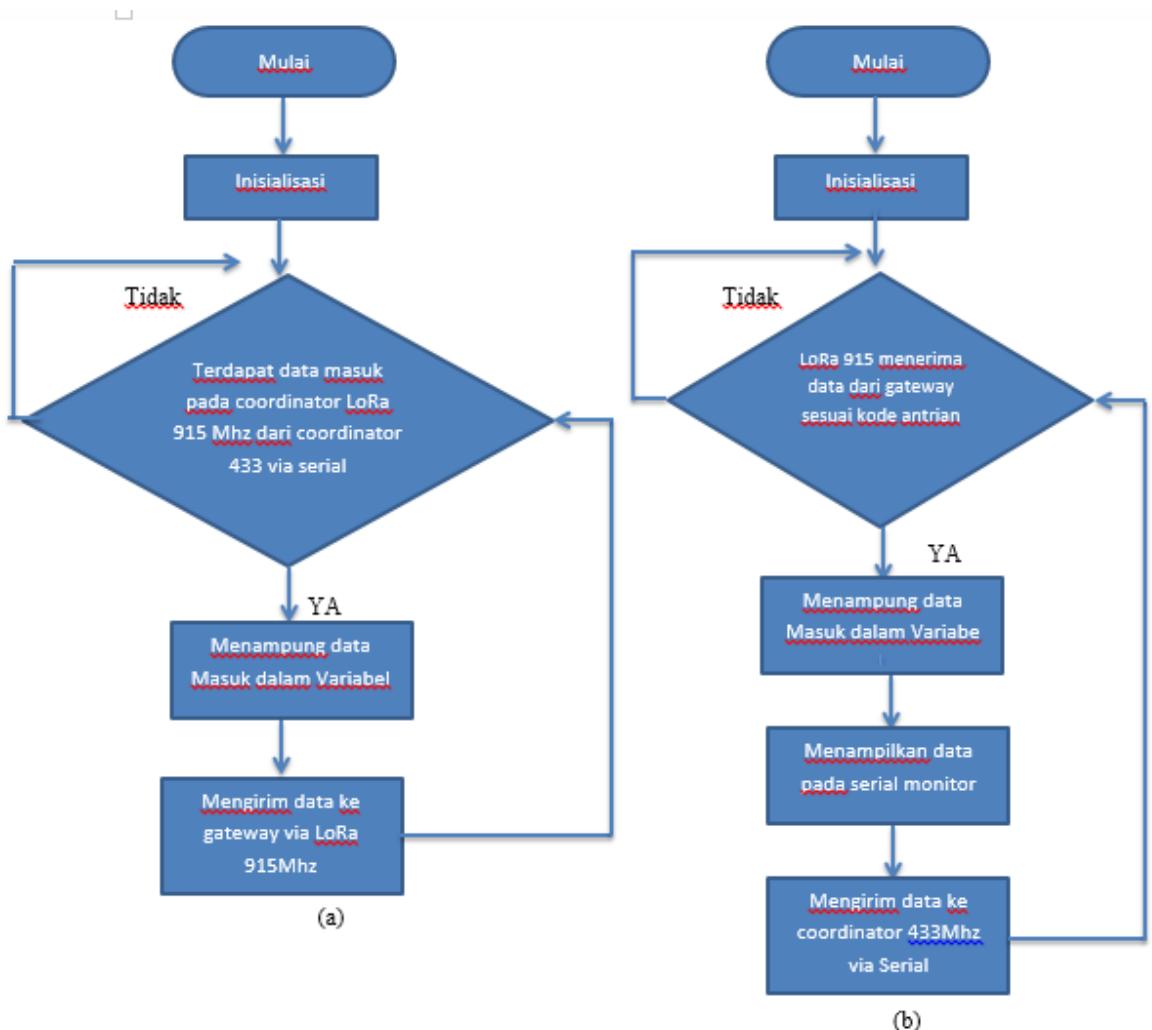


Gambar 3.5 Flowchart Node Koordinator 433 Mhz

- Program Monitoring
- Program Kontroling

Pada gambar 3.5 *flowchart* diatas menjelaskan tentang program monitoring dan kontroling pada *node* koordinator 433 MHz. Pada program monitoring *node* koordinator 433 MHz data di inisialisasi, kemudian data yang dikirimkan oleh *node* sel akan terima oleh *node* koordinator 433 MHz. Jika *node* koordinator 433 MHz tidak menerima maka program akan kembali ke inisialisasi awal. Data yang diterima oleh *node* koordinator 433 MHz ditampung dalam bentul varibel yang terlah ditentukan program. Dan kemudian data tersebut akan diteruskan ke *node* 915 MHz. Pada program kontroling *node* koordinator 433 MHz data di inisialisasi, data dikirimkan oleh *node* koordinator 915 MHz dan diterima oleh *node* koordinator 433 MHz melalui komunikasi serial. Jika data tidak diterima maka akan

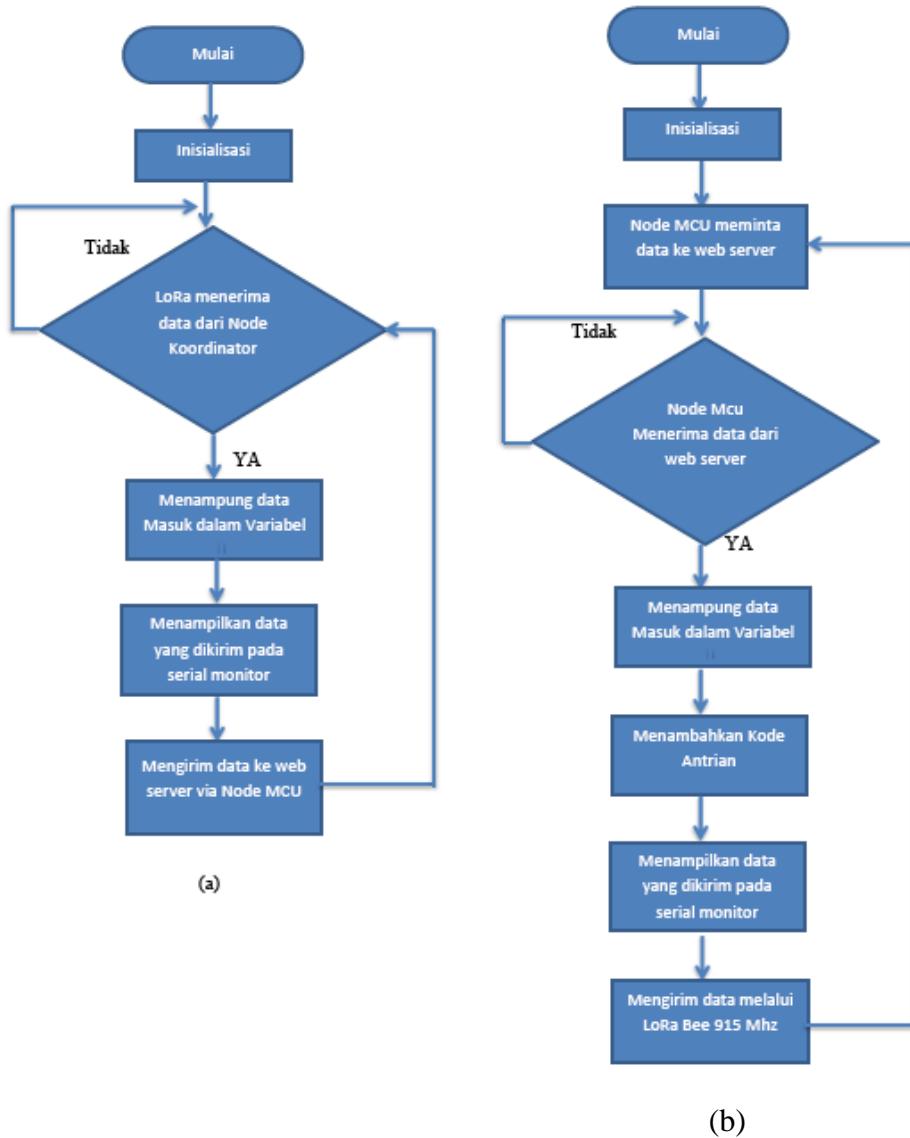
kembali ke inisialisasi awal. Data yang diterima oleh *node* koordinator 433 MHz ditampung dalam variabel yang telah ditentukan yang kemudian dikirimkan ke masing-masing *node* sel melalui komunikasi LoRa 433 MHz.



Gambar 3.6 Flowchart Node Koodinator 915 MHz

- Program Monitoring
- Program Kontroling

Pada *flowchart* gambar 3.6 diatas menjelaskan tentang program monitoring dan kontroling pada *node* koordinator 915 MHz. Pada program monitoring *node* koordinator 915 MHz data di inisialisasi, kemudian data dikirimkan oleh *node* koordinator 433 MHz dan diterima oleh *node* koordinator 915 MHz. Jika data tidak diterima maka program akan kembali ke inisialisasi awal program. Jika data diterima oleh *node* koordinator 915 MHz, data akan ditampung dalam bentuk variabel yang telah ditentukan. Kemudian data tersebut dikirimkan ke *gateway* melalui LoRa *shield* 915 MHz. Pada program kontroling *node* koordinator 915 MHz diatas data di inisialisasi, kemudian data pengendalian dikirimkan oleh *gateway* dan diterima oleh *node* koordinator 915 MHz. Jika data tidak diterima oleh *node* koordinator 915 MHz maka program akan kembali ke inisialisasi awal. Jika data diterima *node* koordinator 915 MHz, maka data selanjutnya ditampung dalam bentuk variabel tertentu. Data tersebut akan ditampilkan di serial monitor *node* koordinator 915 MHz. Kemudian data tersebut dikirimkan ke *node* koordinator 433 MHz melalui komunikasi serial untuk diproses selanjutnya.



Gambar 3.7 Flowchart gateway

- a. Program Monitoring
- b. Program Kontroling

Pada *flowchart* gambar 3.7 menjelaskan tentang program monitoring dan kontroling pada *gateway*. Pada program monitoring *gateway* data di inisialisasi, kemudian data dikirimkan oleh *node* koordinator 433 MHz dan diterima oleh *gateway*. Jika data tidak diterima, maka program akan kembali ke inisialisasi awal. Jika program diterima, maka data tersebut akan ditampung *gateway* dengan variabel yang

ditentukan. Data yang diterima akan diproses dan ditampilkan ke serial monitor. kemudian Data tersebut akan dikirimkan ke *database web server* melalui *NodeMCU ESP8266*. Pada program kontroling *gateway* inisialisasi kontroling, kemudian *NodeMCU ESP8266* meminta data *databases web server*. Data kontroling diterima *gateway* dari *database web server*. Jika data tidak diterima, maka *nodeMCU ESP8266* meminta kembali data dari *database web server*. Jika data diterima oleh *gateway*, maka data tersebut ditampung dalam format varibel yang telah ditentukan. Data tersebut ditambahkan sesuai dengan kode antrian berdasarkan *time* atau waktu data. Data yang diterima *gateway* akan diproses kemudian ditampilkan ke serial monitor *gateway*. Kemudian data tersebut dikirimkan ke *node* koordinator 915 MHz melalui LoRa Bee 915 MHz untuk diproses selanjutnya.

Berikut beberapa *syntax* pada Arduino IDE yang digunakan agar LoRa *Shield* berfungsi. *Syntax* program dan arti *syntax* yang digunakan dapat dilihat pada table 3.9 berikut:

Tabel 3.9 *Syntax* program yang digunakan pada sel gedung

<i>Syntax</i>	Arti
#include <SPI.h>	Memanggil <i>library</i> SPI ke program.
#include <RH_RF95.h>	Memanggil <i>library</i> LoRa Radio Head RH_RF95 ke program.
#include <RHReliableDatagram.h>	Memanggil <i>library</i> LoRa RHReliableDatagram ke program.
#include <SimpleTimer.h>	Memanggil <i>library</i> yang digunakan untuk menggantikan fungsi <i>delay</i> pada Arduino.
Void setup()	Menginisialisasi fungsi <i>setup()</i> yang akan dijalankan sekali sejak program dijalankan dan rangkaian

	mendapatkan sumber tegangan.
Void loop()	Menginisialisasi fungsi loop() yang akan dijalankan berulang kali sejak program dijalankan dan rangkaian mendapat sumber tegangan.
RH_RF95 rf95	Menginisialisasi RH_RF95 menjadi rf95.
if(kirim.init()) if(terima.init())	Inisialisasi LoRa yang akan digunakan, jika yang digunakan tidak berfrekuensi 433MHz pada sel gedung dan 915 MHz pada koordinator B maka akan mucul “RF Radio initialization failed” pada serial monitor.
if (!rf95.setFrequency(FREQ))	Mengatur frekuensi kerja LoRa menjadi 433MHz atau 915 MHz.
kirim.sendto((uint8_t *) &data, sizeof(data),	Mengirimkan isi variabel data melalui komunikasi LoRa.
rf95.waitPacketSent(100);	Menunggu paket yang dikirim melalui komunikasi LoRa terkirim.
terima.recvfrom((uint8_t *) &data, &bufLen, &from	Menerima isi variabel data melalui komunikasi LoRa.
SimpleTimer firstTimer(5000); SimpleTimer secondTimer(0);	Membuat pengatur waktu pertama dan kedua dan menentukan intervalnya dalam milidetik.
firstTimer.isReady() secondTimer.isReady()	Mengecek pengatur waktu pertama dan kedua yang sudah siap.
firstTimer.reset() secondTimer.reset()	Mereset pengatur waktu pertama dan kedua.

Tabel 3.10 *Syntax* program yang digunakan pada *node* Koordinator

<i>Syntax</i>	Arti
#include <SPI.h>	Memanggil <i>library</i> SPI ke program.
#include <RH_RF95.h>	Memanggil <i>library</i> LoRa <i>Radio Head</i> RH_RF95 ke program.
#include <RHReliableDatagram.h>	Memanggil <i>library</i> LoRa RHReliableDatagram ke program.
#include <SimpleTimer.h>	Memanggil <i>library</i> yang digunakan untuk menggantikan fungsi <i>delay</i> pada Arduino.
Void setup()	Menginisialisasi fungsi setup() yang akan dijalankan sekali sejak program dijalankan dan rangkaian mendapatkan sumber tegangan.
Void loop()	Menginisialisasi fungsi loop() yang akan dijalankan berulang kali sejak program dijalankan dan rangkaian mendapat sumber tegangan.
RH_RF95 rf95	Menginisialisasi RH_RF95 menjadi rf95.
if(kirim.init()) if(terima.init())	Inisialisasi LoRa yang akan digunakan, jika yang digunakan tidak berfrekuensi 433MHz pada sel gedung dan 915 MHz pada koordinator B maka akan mucul “RF Radio initialization failed” pada serial monitor.
if (!rf95.setFrequency(FREQ))	Mengatur frekuensi kerja LoRa menjadi 433MHz atau 915 MHz.
kirim.sendto((uint8_t *) &data, sizeof(data),	Mengirimkan isi variabel data melalui komunikasi LoRa.
rf95.waitPacketSent(100);	Menunggu paket yang dikirim

	melalui komunikasi LoRa terkirim.
terima.recvfrom((uint8_t *) &data, &bufLen, &from)	Menerima isi variabel data melalui komunikasi LoRa.
SimpleTimer firstTimer(5000); SimpleTimer secondTimer(0);	Membuat pengatur waktu pertama dan kedua dan menentukan intervalnya dalam milidetik.
firstTimer.isReady() secondTimer.isReady()	Mengecek pengatur waktu pertama dan kedua yang sudah siap.
firstTimer.reset() secondTimer.reset()	Mereset pengatur waktu pertama dan kedua.

Berikut beberapa *syntax* pada Arduino IDE yang digunakan agar Node MCU ESP8266 dan LoRa Bee berfungsi. *Syntax* program dan arti *syntax* yang digunakan dapat dilihat pada table 3.11 berikut:

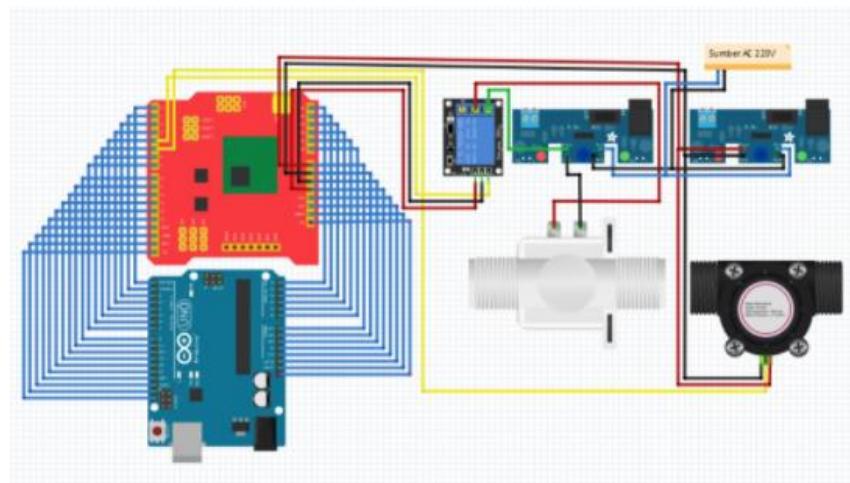
Tabel 3.11 Syntax program yang digunakan pada *gateway*

Syntax	Arti
#include <ESP8266WiFi.h>	Memanggil <i>library</i> ESP8266WiFi ke program.
#include <ESP8266HTTPClient.h>	Memanggil <i>library</i> ESP8266HTTPClient ke program.
#include <SPI.h>	Memanggil <i>library</i> SPI ke program.
#include <RH_RF95.h>	Memanggil <i>library</i> LoRa Radio Head RH_RF95 ke program.
#include <RHReliableDatagram.h>	Memanggil <i>library</i> LoRa RHReliableDatagram ke program.
#include <SimpleTimer.h>	Memanggil <i>library</i> yang digunakan untuk menggantikan fungsi <i>delay</i> pada Arduino.
#include "CTBot.h";	Memanggil <i>library</i> CTTBot ke program,

	CTBoT digunakan untuk mengelola Telegram Bot pada <i>platform</i> ESP8266.
#include <ArduinoJson.h>	Memanggil <i>library</i> ArduinoJson ke program.
CTBot notifikasi; CTBot notifikasi1; CTBot notifikasi2;	Membuat variabel CTBot untuk notifikasi Telegram.
SimpleTimer firstTimer(60000); SimpleTimer secondTimer(0);	Membuat pengatur waktu pertama dan kedua dan menentukan intervalnya dalam milidetik.
RH_RF95 rf95 (RFM95_CS, RFM95_INT)	Menginisialisasi RH_RF95 menjadi rf95 pada pin D2 dan D3.
void setup(void)	Menginisialisasi fungsi setup() yang akan dijalankan sekali sejak program dijalankan dan rangkaian mendapatkan sumber tegangan.
void loop (void)	Menginisialisasi fungsi loop() yang akan dijalankan berulang kali sejak program dijalankan dan rangkaian mendapat sumber tegangan.
firstTimer.isReady() secondTimer.isReady()	Mengecek pengatur waktu pertama dan kedua yang sudah siap.
gate.sendto((uint8_t *) &data, sizeof(data))	Mengirimkan isi variabel data melalui komunikasi LoRa.
rf95.waitPacketSent(100)	Menunggu paket yang dikirim melalui komunikasi LoRa terkirim.
gate.recvfrom((uint8_t *) &data, &bufLen, &from)	Menerima isi variabel data melalui komunikasi LoRa.

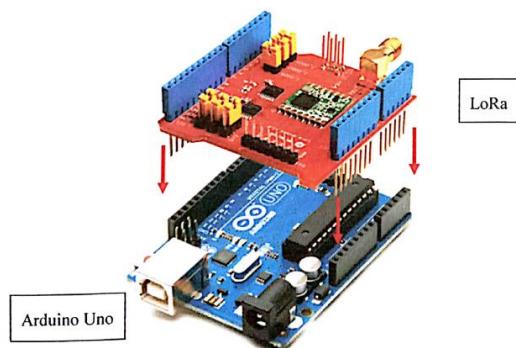
### 3.2.5 Perancangan Prototype

Perancangan *prototype* terdiri dari perancangan *node* sel, *node* koordinator dan *gateway*. Perangkat keras yang digunakan untuk membuat *prototype node* sel yang dikontrol oleh LoRa *Shield* dan Arduino terdiri dari 5 buah LoRa *Shield* sebagai penerima data perintah selenoid valve *on/off* dari web atau android melalui koordinator, 5 buah Arduino Uno sebagai Mikrokontroler, 5 buah *flowmeter* sebagai sensor penghitung kecepatan dan debit air yang dikirim ke web melalui *node* koordinator dan 5 buah *selenoid valve* untuk mengedalikan *on/off* sistem air melalui web atau android. Gambar pengkabelan perangkat keras *node* pusat dan gedung ditunjukkan pada gambar 3.8



Gambar 3.8 Pengkabelan perangkat keras *node* pusat dan gedung

Sumber: Dokumentasi Penulis



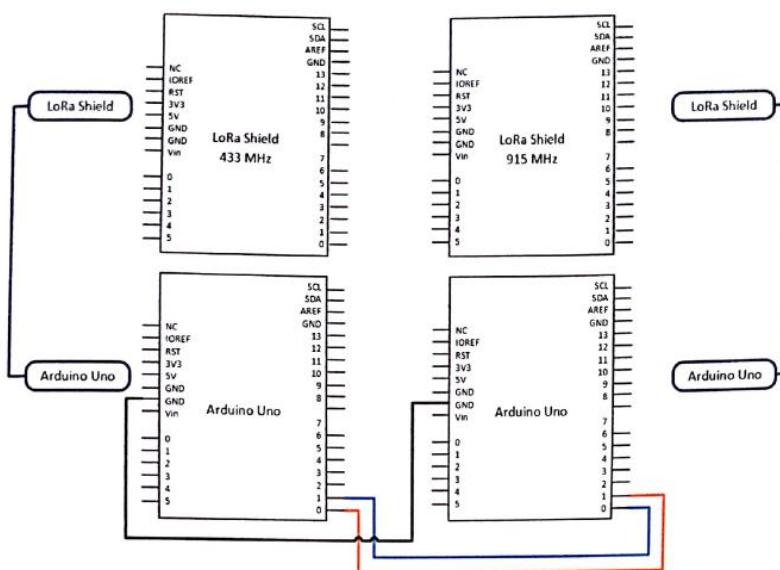
Gambar 3.9 Pemasangan LoRa *Shield* ke Arduino Uno

Sumber: Dokumentasi Penulis

*Node Koordinator* menggunakan 1 buah LoRa *Shield* 433MHz, 1 buah LoRa *Shield* 915MHz dan 2 buah Arduino Uno. Semua pin LoRa *Shield* 433MHz maupun 915MHz terhubung dengan pin pada Arduino Uno. Pengkabelan pin GND LoRa *Shield* 433 MHz terhubungan dengan pin GND LoRa *Shield* 915 MHz untuk jalur *ground*. Pin 0 LoRa *Shield* 433 MHz sebagai *receiver* (Rx) terhubung dengan Pin 1 LoRa *Shield* 915 MHz sebagai *transmitter* (Tx) sedangkan Pin 1 LoRa *Shield* 433 MHz sebagai *transmitter* (Tx) dengan pin 0 LoRa *Shield* 915 MHz sebagai *receiver* (Rx). Keduanya digunakan untuk komunikasi serial. Pengkabelan *node* koordinator dalam dilihat pada tabel 3.2 dan pemasangan kabel *node* koordinator ditunjukkan pada gambar 3.10.

Tabel 3.12 Pengkabelan *node* koordinator

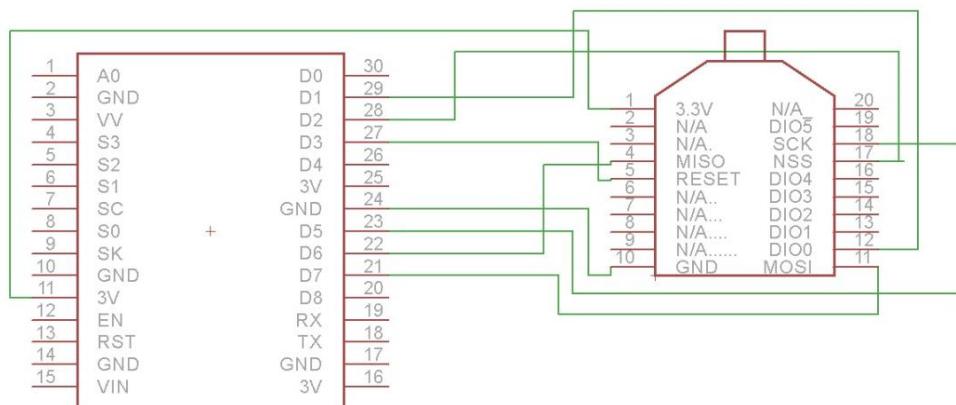
Lora <i>Shield</i> 433 MHz	Lora <i>Shield</i> 915 MHz
Rx (0)	Tx (1)
Tx (1)	Rx (0)
GND	GND



Gambar 3.10 Pemasangan pengkabelan *node* koordinator

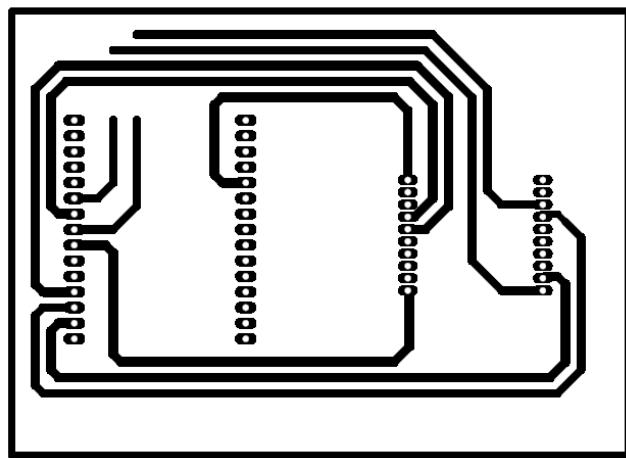
Sumber: Dokumentasi Penulis

Pada *gateway* menggunakan 1 buah NodeMCU ESP8266 dan 1 buah LoRa Bee 915 MHz. Pada pembuatan *gateway* diperlukan PCB untuk memberikan jalur pin yang digunakan. NodeMCU ESP8266 difungsikan supaya terhubung ke *Web Server*. NodeMCU ESP8266 adalah sebuah modul WiFi yang dapat terhubung ke internet dengan modul 32-bit. Kemudian LoRa Bee 915 MHz digunakan untuk bertukar data dengan *node* koordinator yang memiliki frekuensi sama. Syarat perangkat lain yang dapat berkomunikasi dengan *gateway* adalah memiliki frekuensi yang sama. LoRa Bee merupakan *transceiver* yang bersifat *half-duplex* yang artinya tidak dapat menerima dan mengirim data secara bersamaan sehingga menerima dan mengirim data dilakukan secara bergantian. Pemasangan pengkabelan *gateway* dapat dilihat gambar 3.11 dan jalur PCB *gateway* dapat dilihat pada gambar 3.12.



Gambar 3.11 Pengkabelan pada *gateway*

Sumber: Dokumentasi Penulis



Gambar 3.12 Jalur PCB *gateway*

Sumber: Dokumentasi Penulis

Pada gambar 3.11 menunjukan Pin 3.3V (Pin 1) pada LoRa Bee terhubung dengan Pin 3V pada NodeMCU ESP8266 untuk mendapatkan *input* tegangan. Pin GND (Pin 10) pada LoRa Bee terhubung dengan Pin G/GND pada NodeMCU ESP8266 untuk jalur *ground*. Pin *reset* (Pin 5) pada LoRa Bee terhubung dengan Pin D3 pada NodeMCU ESP8266 untuk melakukan fungsi *reset*. Pin DIO0 (Pin 12) pada LoRa Bee terhubung dengan Pin D1 NodeMCU ESP8266 untuk digital *input/output*. Pin SCK (Pin 18) LoRa Bee dihubungkan dengan Pin D5 NodeMCU ESP8266, Pin MISO (Pin 4) LoRa Bee dihubungkan dengan Pin D6 NodeMCU ESP8266, dan Pin MOSI (Pin 11) LoRa Bee dihubungkan dengan Pin D7 NodeMCU ESP8266. Pin SCK, MISO dan MOSI digunakan untuk komunikasi SPI. Pin SCK digunakan untuk sinkronisasi agar tidak terjadi kesalahan dalam komunikasi, Pin MISO digunakan untuk menerima data dan Pin MOSI digunakan untuk mengirim data ke mikrokontroler. Pin NSS (Pin 17) Lora Bee terhubung dengan Pin D2 NodeMCU ESP8266 untuk mengaktifkan SPI. Pengkabelan *gateway* dapat dilihat pada tabel 3.13

Tabel 3.13 Pengkabelan *gateway*

NodeMCU ESP8266	Lora Bee 915 MHz
3.3V	Pin 1
GND	Pin 10
D1	Pin 12
D2	Pin 17
D3	Pin 5
D5	Pin 18
D6	Pin 4
D7	Pin 11

### 3.3 Pembuatan Sistem

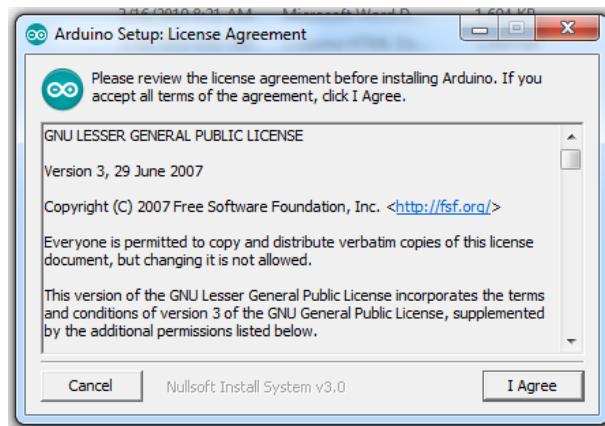
Pembuatan sistem adalah tahapan untuk membuat alat sistem berupa pembuatan *software* (perangkat lunak) dan pembuatan *hardware* (perangkat keras) atau yang disebut *prototype* alat yang dibuat. Pembuatan sistem ini adalah tahapan yang dilakukan setelah melakukan tahapan perencanaan sistem dan perancangan sistem. Pada pembuatan sistem ini dilakukan tahapan eksekusi dalam membuat *prototype* alat dalam bentu nyata. Pembuatan sistem ini dilakukan secara bertahap setelah melakukan beberapa kali pengujian bahan dan alat yang digunakan. Pembuatan sistem ini meliputi :

#### 3.3.1 Pembuatan *Software* (Perangkat Lunak)

Instalasi perangkat lunak (*software*) merupakan penginstalan aplikasi yang mendukung pada pembuatan. Perangkat lunak yang dibutuhkan adalah Arduino IDE. Arduino IDE merupakan suatu aplikasi yang telah disediakan oleh Arduino bagi para *desainer* untuk dapat melakukan pemrograman Arduino dengan basis bahasa C. Aplikasi ini tersedia gratis dan bisa di *download* secara langsung pada halaman resmi Arduino. Aplikasi ini mendukung berbagai sistem operasi seperti Windows, Linux dan Mac.

Berikut adalah proses tahapan penginstalan Arduino IDE :

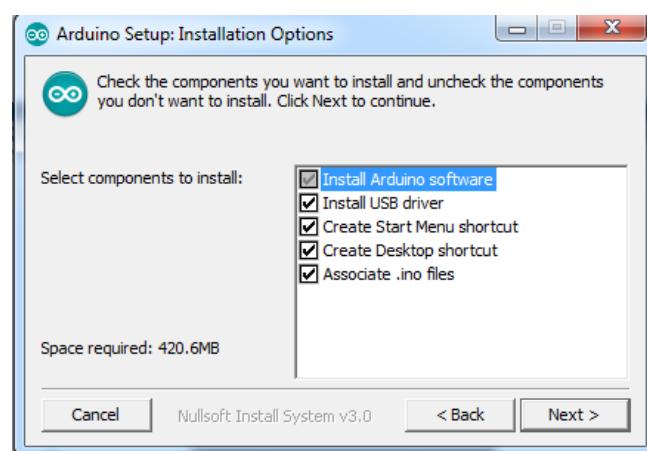
- a. *Download* aplikasi pada halaman resmi Arduino
- b. Jalankan file *installer* Arduino IDE yang telah di *dowload*.
- c. Lalu akan muncul *License Agreement* seperti pada gambar 3.13  
→ Klik tombol *I Agree*



Gambar 3.13 Tampilan *License Agreement*

Sumber: Dokumentasi Penulis

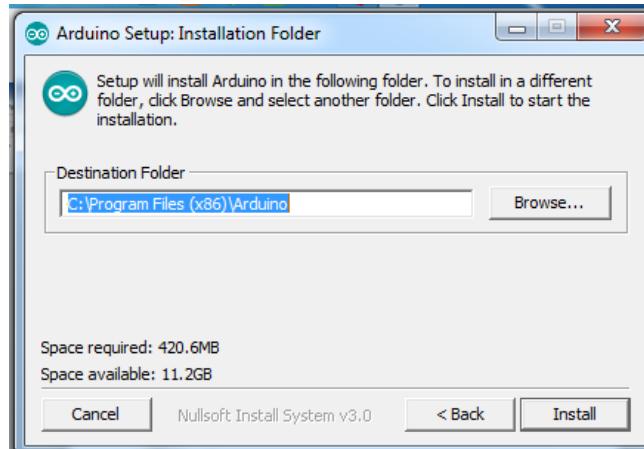
- d. Selanjutnya akan muncul jendela *installation Options* seperti pada gambar 3.14 untuk memilih komponen apa saja yang ingin di *install* → Klik *Next*



Gambar 3.14 Tampilan *Installation Options*

Sumber: Dokumentasi Penulis

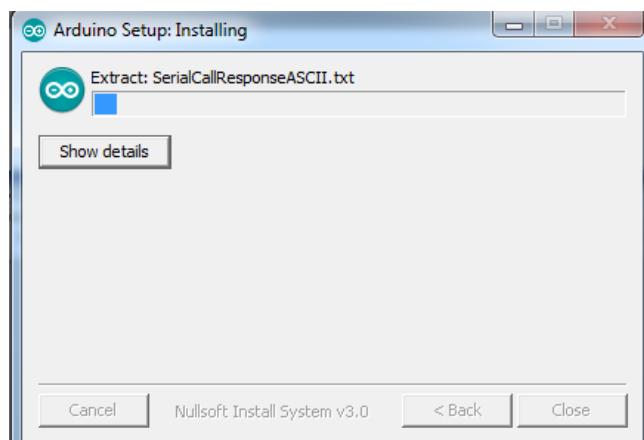
- e. Kemudian akan muncul jendela *Installation Folder* seperti pada gambar 3.15 untuk memilih lokasi folder instalasi Arduino IDE  
→ Klik *Install*



Gambar 3.15 Tampilan *Installation Folder*

Sumber: Dokumentasi Penulis

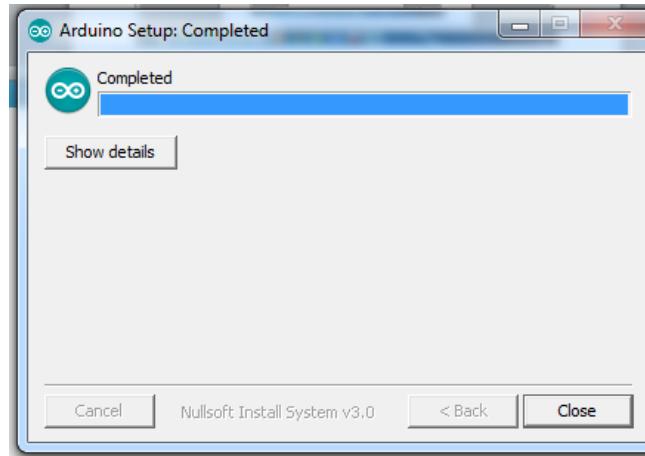
- f. Lalu muncul jendela proses instalasi Arduino IDE seperti gambar 3.16 tunggu sampai proses instalasi selesai.



Gambar 3.16 Tampilan Proses *Installing*

Sumber: Dokumentasi Penulis

- g. Jika proses instalasi sudah selesai maka muncul tampilan seperti gambar 3.17 → Klik *Close*.



Gambar 3.17 Tampilan *Installing Complete*

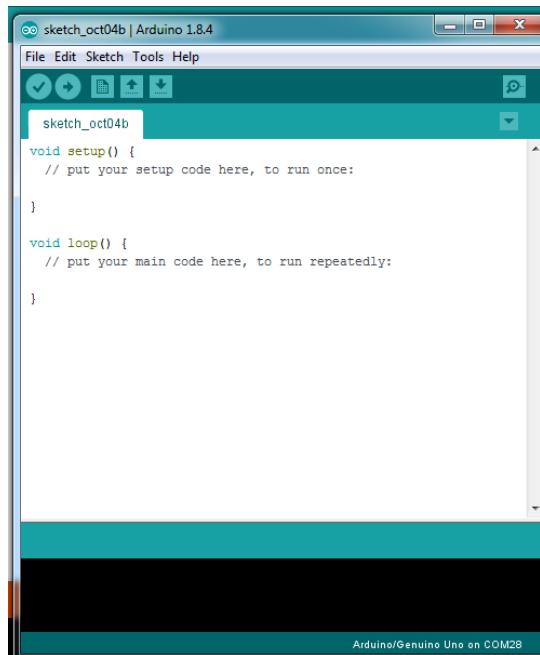
Sumber: Dokumentasi Penulis

- h. Setelah semua proses instalasi selesai, jalankan aplikasi Arduino IDE. Buka aplikasi Arduino IDE maka muncul tampilan awal seperti gambar 3.18 dan tampilan *editor* seperti gambar 3.19



Gambar 3.18 Tampilan Awal Arduino IDE

Sumber: Dokumentasi Penulis



Gambar 3.19 Tampilan *editor* Arduino IDE

Sumber: Dokumentasi Penulis

### 3.3.2 Pembuatan *Prototype* Sistem

Pembuatan *prototype* sistem merupakan tahapan yang dilakukan untuk membuat implementasi alat yang sebenarnya dalam bentuk *prototype*. Pembuatan *prototype* sistem ini dibuat di Laboratorium Telekomunikasi Barat 05 Politeknik Negeri Semarang. Pembuatan *prototype* sistem ini adalah tahapan akhir yang dilakukan setelah melakukan perencanaan sistem dan perancangan sistem dan tahapan terakhir yang dilakukan sebelum melakukan pengujian sistem. Pembuatan *prototype* sistem ditunjukkan pada gambar 3.20 (a) dan (b)



Gambar 3.20 *prototype* alat

(a) Sensor *Node* sel

(b) Mikrokontroller sistem *node* sel, *node* koordinator dan *gateway*

Sumber : Dokumentasi Penulis

Pada gambar 3.20 (a) dan (b) pembuatan *prototype* sistem ini terdapat 5 buah *node* sel yang terdiri dari 4 *node* gedung dan 1 *node* pusat, 1 *node* koordinator dan 1 *gateway*. Pada *node* sel terdapat 2 bagian yaitu bagian sensor-sensor seperti sensor *flowmeter* dan sensor selenoid valve yang diletakkan diatas kran yang berfungsi untuk monitoring dan pengendalian sistem air dan bagian mikrokontroller yang berisi Arduino Uno, LoRa *Shield* 433 MHz dan *relay* yang berfungsi untuk

*programming* program dan komunikasi untuk menjalankan sensor. Pada bagian *node* koordinator terletak dibagian mikrokontroller yang berisi 2 buah Arduino Uno dan 1 buah LoRa *Shield* 433 MHz dan 1 buah LoRa *Shield* 915 MHz dan bagian *gateway* terletak pada bagian mikrokontroller yang berisi 1 buah *NodeMCU* ESP8266, 1 buah LoRa Bee 915 MHz, 1 buah *power supply* 5V DC 3A, 1 buah *power supply* 12V DC 3A dan 2 buah terminal strip. Bagian sensor dan bagian mikrokontroller saling terkoneksi dengan menggunakan kabel. Adapun tahapan-tahapan yang dilakukan dalam pembuatan *prototype* sistem :

1. Mempersiapkan perangkat yang akan digunakan.

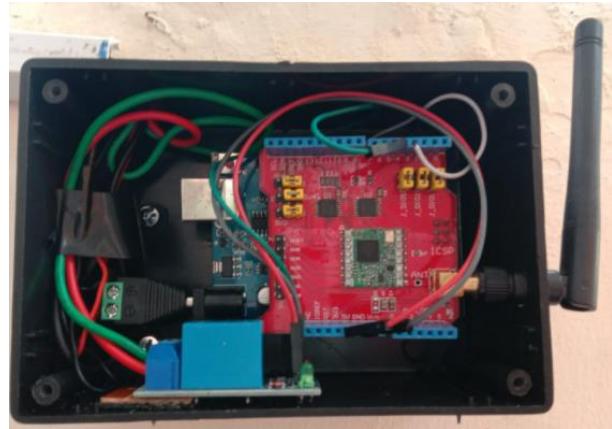
Perangkat yang dipersiapkan untuk melakukan pengujian pada sel pusat dan sel gedung adalah 5 buah mikrokontroler Arduino Uno, 5 buah LoRa *Shield* 433MHz, 4 buah *flowmeter* diamater 1/2 inchi, 1 buah *flowmeter* diameter 3/4 inchi, 4 buah selenoid valve *normally open* diameter 1/2 inchi, 1 buah selenoid valve *normally open* diameter 3/4 inchi, 5 buah *relay* SPDT, 5 buah *power supply* 12V DC, sakelar listrik 220V, kabel dan jumper secukupnya. Kemudian untuk perangkat *node* koordinator yang dipersiapkan untuk melakukan pengujian adalah 2 buah mikrokontroler Arduino Uno, 1 buah LoRa *Shield* 433MHz, 1 buah LoRa *Shield* 915MHz dan 3 buah jumper male to male. Sedangkan untuk perangkat *gateway* yang dipersiapkan yaitu modul *gateway* yang telah dibuat, 1 buah mikrokontroler *NodeMCU* ESP8266, 1 buah LoRa Bee 915MHz, serta laptop yang sudah terinstall aplikasi Arduino IDE.

2. Instalasi perangkat sel pusat, sel gedung, *node* koordinator dan *gateway*

- a. Proses instalasi untuk perangkat sel pusat dan sel gedung.

Untuk instalasi perangkat sel pusat dan sel gedung, mikrokontroler Arduino Uno dipasang menempel pada LoRa

*Shield* dengan antena frekuensi 433MHz. Kemudian untuk perakitan sensor *flowmeter*, kabel pada *flowmeter* dihubungkan dengan perangkat LoRa *Shield* yang menempel pada mikrokontroler. *Flowmeter* menggunakan PIN 3 untuk data, seperti pada gambar 3.21.



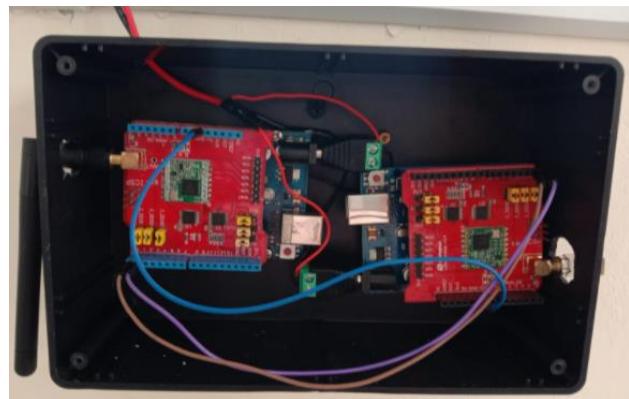
Gambar 3.21 Pengkabelan perangkat sel gedung

Sumber : Dokumentasi Penulis

Lalu untuk perakitan selenoid valve, persiapkan *relay* SPDT dan *power supply* 12V DC. Satu *port* selenoid valve dihubungkan dengan *relay* SPDT COM dan *port* lainnya dihubungkan dengan V- pada *power supply*. Kemudian pada *relay* SPDT *normally close* dihubungkan dengan V+ pada *power supply*. Dan pada *power supply port AC* dihubungkan dengan saklar yang terhubung pada AC 220V.

b. Proses instalasi untuk perangkat *node* koordinator.

Untuk instalasi perangkat *node* koordinator, mikrokontroler Arduino Uno dipasang menempel pada LoRa *Shield* dengan masing-masing antena dengan frekuensi 433MHz dan frekuensi 915MHz. Kemudian untuk LoRa *Shield* frekuensi 433MHz dengan LoRa *Shield* 915MHz dihubungkan untuk komunikasi serial, seperti pada gambar 3.22.



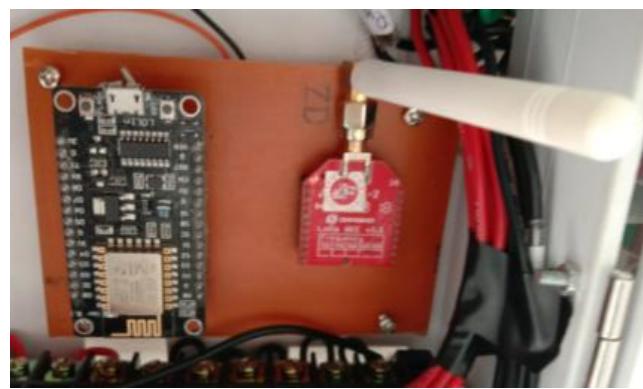
Gambar 3.22 Pengkabelan perangkat Koordinator

Sumber : Dokumentasi Penulis

Kemudian hubungkan *power* mikrokontroler Arduino Uno ke *port* USB pada laptop dengan menggunakan kabel USB, untuk kemudian diprogram dan untuk melihat data yang masuk serta data yang diolah oleh *node* koordinator.

c. Proses instalasi untuk perangkat *gateway*.

Untuk instalasi perangkat *gateway*, *NodeMCU* ESP8266 dihubungkan dengan LoRa Bee 915MHz lengkap dengan antena menggunakan modul *gateway* pada PCB yang telah dibuat, seperti pada gambar 3.23.



Gambar 3.23 Pemasangan perangkat *gateway*

Sumber: Dokumentasi Penulis

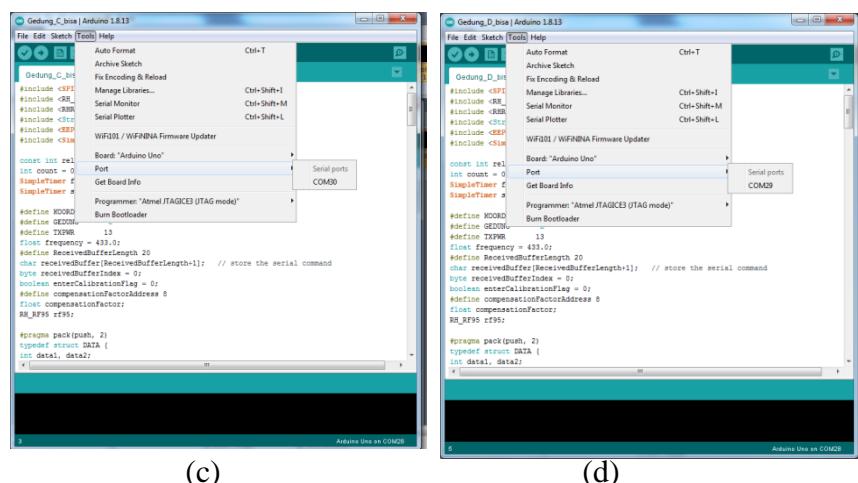
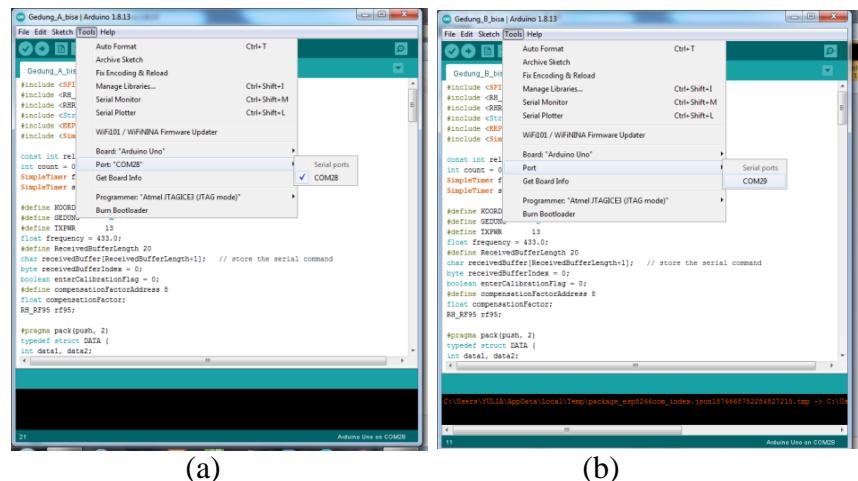
Kemudian menghubungkan perangkat yang telah di rangkai ke *port* USB laptop menggunakan kabel USB untuk

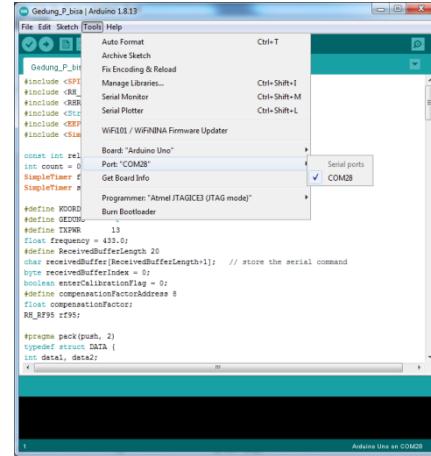
kemudian diprogram dan untuk melihat data yang masuk serta data yang diolah oleh *gateway*.

### 3. Mengupload program yang telah dibuat pada aplikasi Arduino IDE

#### a. *Upload* program untuk sel gedung

1. Membuka program sel gedung yang telah dibuat pada menu *File- Open* memilih file pada lokasi penyimpanan.
2. Selanjutnya memilih port COM yang telah terpasang pada menu *Tools- Port*, seperti pada Gambar 3.24.





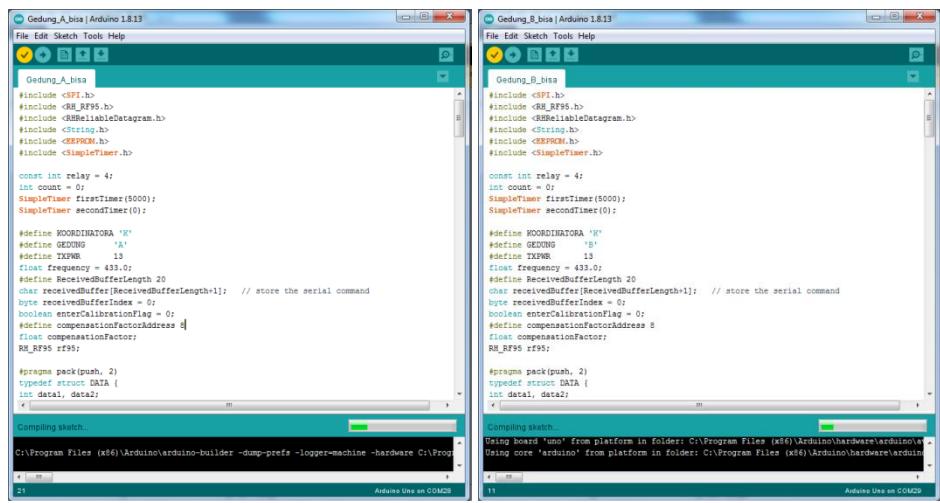
(e)

Gambar 3.24 Tampilan Port COM pada sel gedung

- (a) Sel gedung A
- (b) Sel gedung B
- (c) Sel gedung C
- (d) Sel gedung D
- (e) Sel gedung P

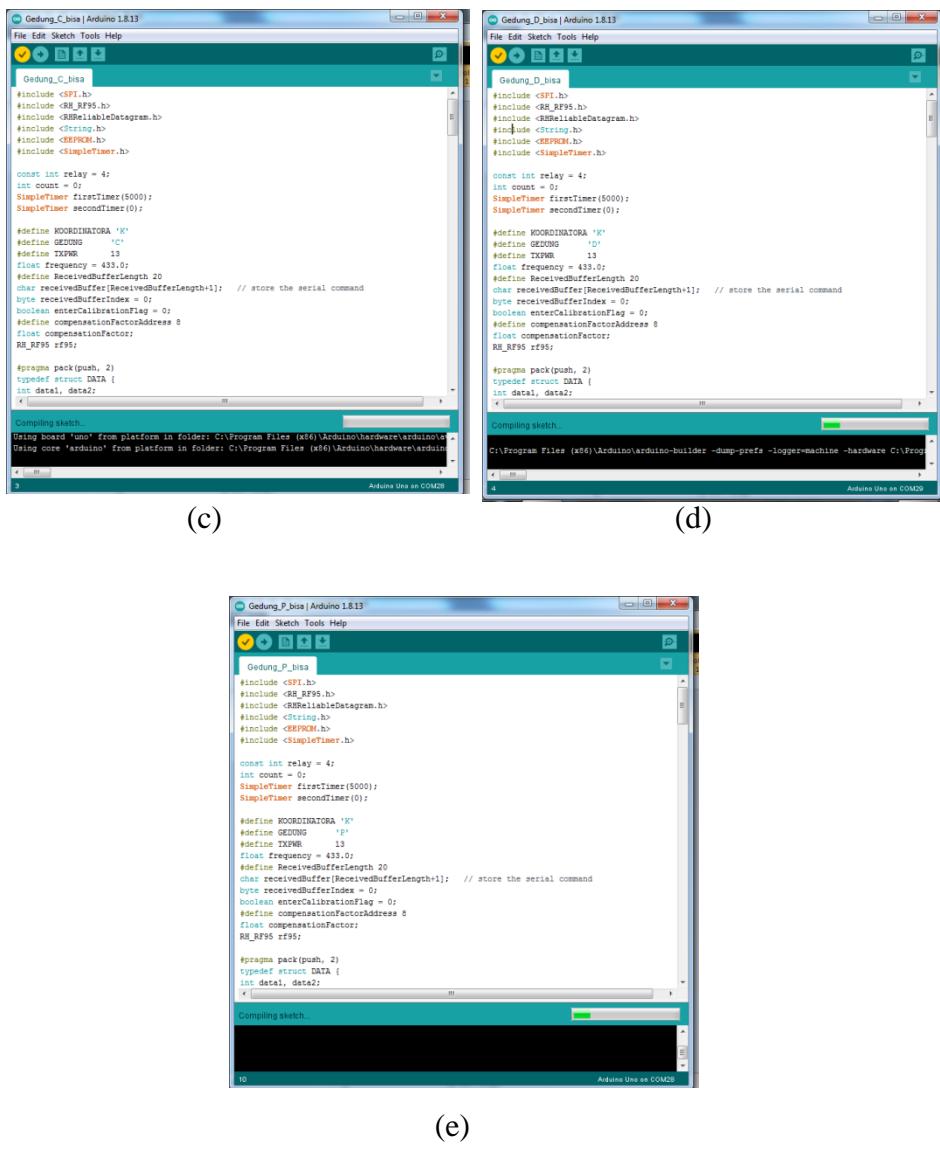
Sumber : Dokumentasi Penulis

3. Melakukan proses *Compile* dengan pilih ‘*verify*’ pada *toolbar* Arduino seperti Gambar 3.25.



(a)

(b)

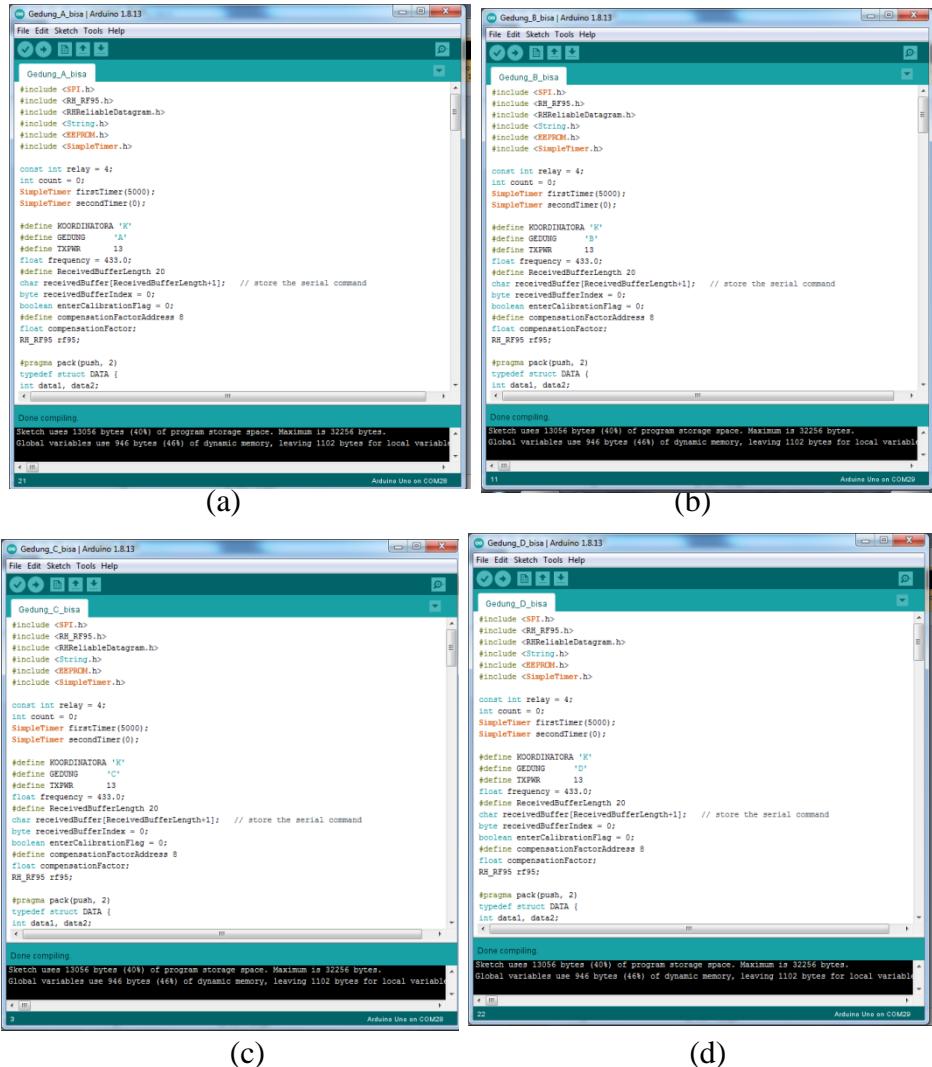


Gambar 3.25 Tampilan proses *compile* pada sel gedung

- (a) Gedung A
- (b) Gedung B
- (c) Gedung C
- (d) Gedung D
- (e) Gedung P

Sumber: Dokumentasi Penulis

4. Apabila proses *Compile* sudah selesai ditandai dengan muncul ‘*Done compiling...*’ di *message bar* Arduino, seperti Gambar 3.26.



```

Gedung_P.ino
#include <SPI.h>
#include <RF_RF95.h>
#include <RFReliableDatagram.h>
#include <String.h>
#include <EEPROM.h>
#include <SimpleTimer.h>

const int relay = 4;
int count = 0;
SimpleTimer firstTimer(5000);
SimpleTimer secondTimer(0);

#define HOODINATORA 'K'
#define GEDUNG 'P'
#define TXNR 13
float frequency = 433.0;
#define ReceivedBufferLength 20
char receivedBuffer[ReceivedBufferLength+1]; // store the serial command
byte receivedBufferIndex = 0;
boolean enterCalibrationFlag = 0;
#define compensationFactorAddress 8
float compensationFactor;
RF_RF95 rf95;

#pragma pack(push, 2)
typedef struct DATA {
    int data1, data2;
} DATA;

```

Done compiling  
Sketch uses 13054 bytes (40%) of program storage space. Maximum is 32256 bytes.  
Global variables use 946 bytes (46%) of dynamic memory, leaving 1102 bytes for local variables.

(e)

Gambar 3.26 Tampilan *done compiling* pada sel gedung

(a) Gedung A

(b) Gedung B

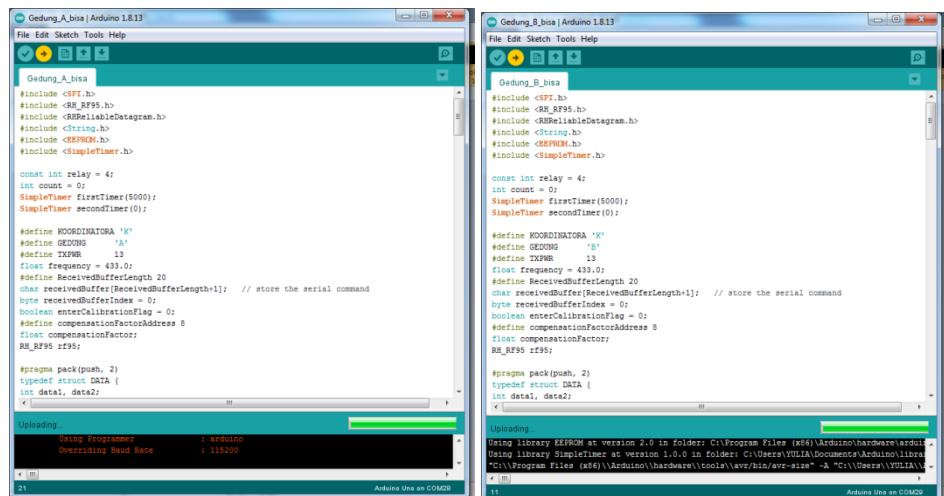
(c) Gedung C

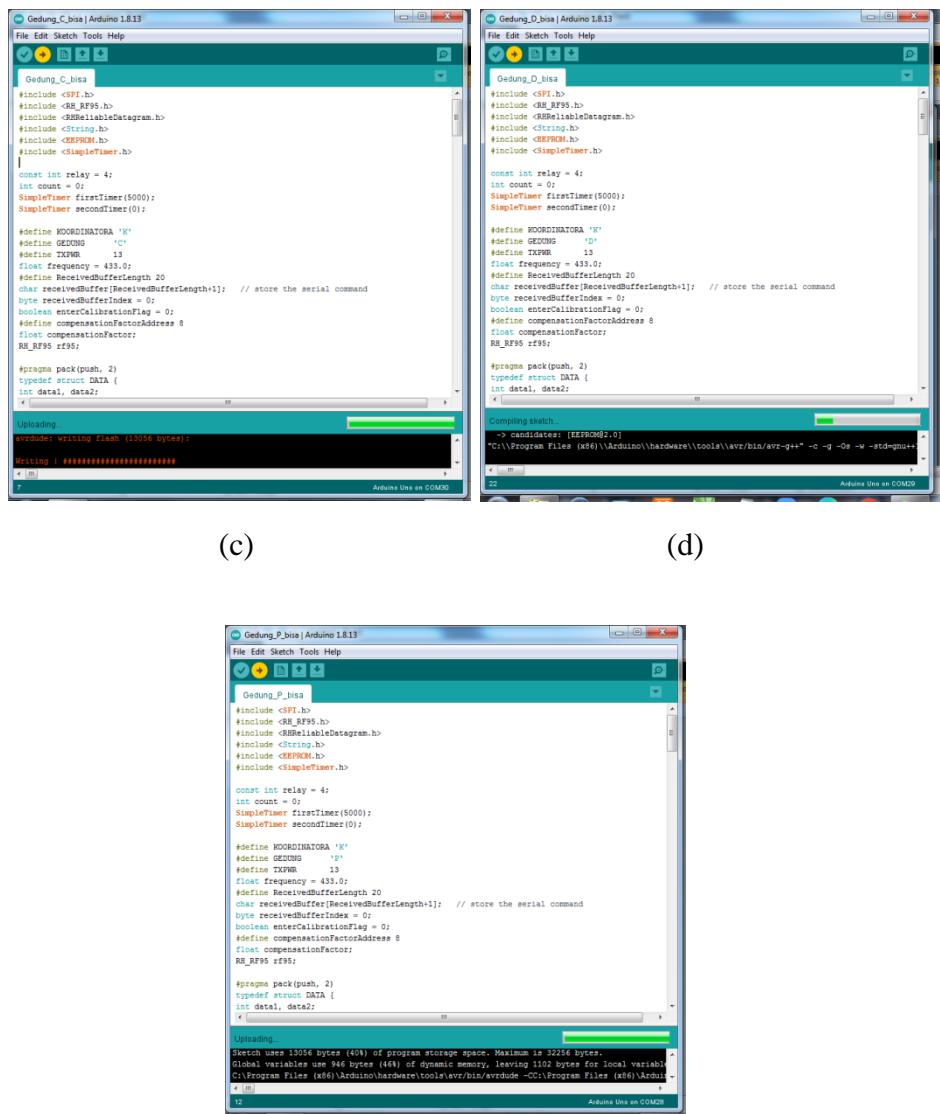
(d) Gedung D

(e) Gedung P

Sumber : Dokumentasi Penulis

5. Kemudian melakukan proses *Upload* dengan pilih ‘*Upload*’ pada *toolbar* Arduino, seperti pada Gambar 3.27.





Gambar 3.27 Tampilan proses *upload* pada sel gedung

(a) Gedung A

(b) Gedung B

(c) Gedung C

(d) Gedung D

(e) Gedung P

Sumber: Dokumentasi Penulis

6. Apabila proses *Upload* sudah selesai ditandai dengan muncul ‘*Done uploading..*’ di *message-bar* Arduino, seperti pada Gambar 3.28.



Gambar 3.28 Tampilan *done uploading*

Sumber: Dokumentasi Penulis

7. Selanjutnya pilih ‘*Serial Monitor*’ pada *toolbar* Arduino untuk melakukan pantauan data yang dikirim ke koordinator, dan data yang diterima dari koordinator menggunakan serial monitor, seperti pada Gambar 3.29.

```

14:10:01.024 -> RF radio initialized.
14:10:01.058 -> RF95 max message length = 251
14:10:01.134 -> RF95 radio initialized.
14:10:01.170 -> RF95 max message length = 251
14:10:15.953 -> Got message from : K
14:10:15.953 -> Received alert1 = 1
14:10:35.977 -> Got message from : K
14:10:35.977 -> Received alert1 = 1
14:10:50.993 -> Got message from : K
14:10:51.027 -> Received alert1 = 1
14:11:01.009 -> 1
14:11:01.009 -> Sending Rate = 1.13
14:11:01.042 -> Sending Notif = 0
14:11:01.042 -> Sending Data Pengendalian = 1
14:11:21.019 -> Got message from : K
14:11:21.052 -> Received alert1 = 1
14:11:36.047 -> Got message from : K
14:11:36.081 -> Received alert1 = 1
14:11:56.076 -> Got message from : K
14:11:56.111 -> Received alert1 = 1
14:12:01.070 -> 2
14:12:01.070 -> Sending Rate = 1.14
14:12:01.104 -> Sending Notif = 0
14:12:01.137 -> Sending Data Pengendalian = 1
14:12:11.165 -> Got message from : K
14:12:11.165 -> Received alert1 = 1
14:12:26.191 -> Got message from : K
14:12:26.191 -> Received alert1 = 1
14:12:41.198 -> Got message from : K
14:12:41.231 -> Received alert1 = 1
14:12:56.227 -> Got message from : K
14:12:56.227 -> Received alert1 = 1
14:13:01.155 -> 3
14:13:01.155 -> Sending Rate = 1.14
14:13:01.189 -> Sending Notif = 0
14:13:01.189 -> Sending Data Pengendalian = 1
14:13:16.257 -> Got message from : K
14:13:16.257 -> Received alert1 = 1

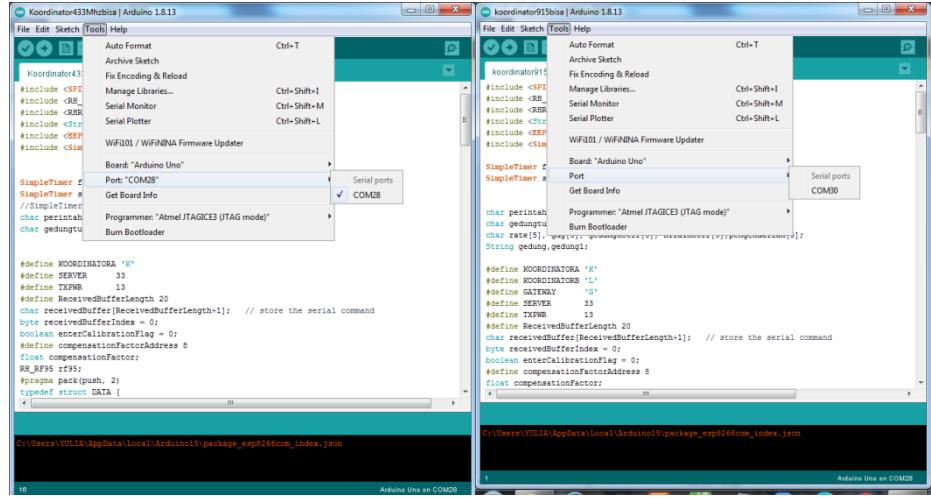
```

Gambar 3.29 Tampilan data *controlling* dan monitoring yang akan dikirim dan diterima dari sel gedung ke koordinator

Sumber : Dokumentasi Penulis

b. *Upload* program untuk koordinator

1. Membuka program koordinator yang telah dibuat pada menu *File- Open* memilih file pada lokasi penyimpanan.
2. Selanjutnya memilih port COM yang telah terpasang pada menu *Tools- Port*, seperti pada Gambar 3.30.



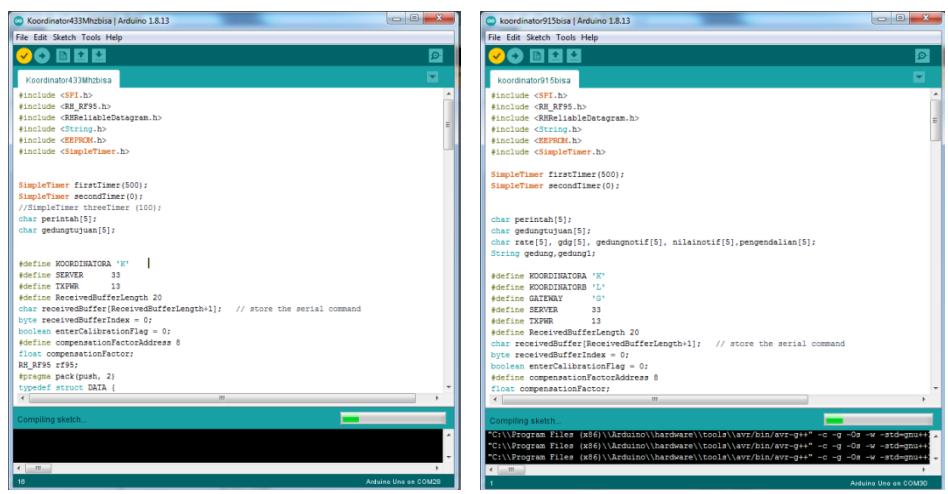
(a)

(b)

Gambar 3.30 Tampilan *port COM* pada koordinator

Sumber : Dokumentasi Penulis

3. Melakukan proses *Compile* dengan pilih ‘*verify*’ pada *toolbar* Arduino seperti Gambar 3.31.



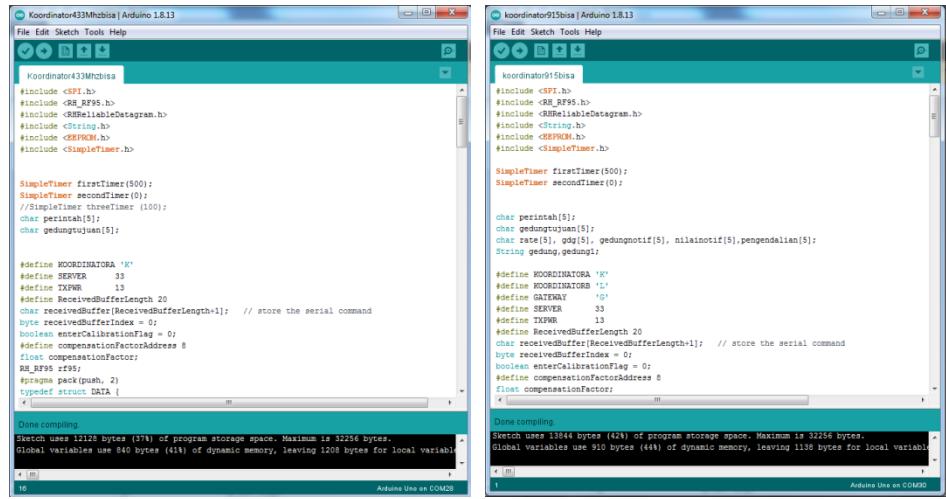
(a)

(b)

Gambar 3.31 Tampilan proses *compile* pada koordinator

Sumber : Dokumentasi Penulis

4. Apabila proses *Compile* sudah selesai ditandai dengan muncul ‘*Done compiling...*’ di *message bar* Arduino, seperti Gambar 3.32.

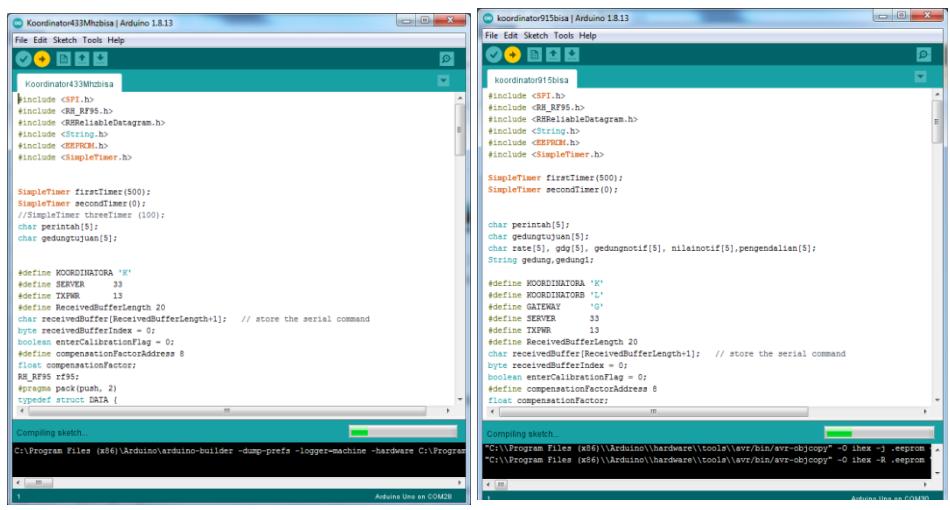


The image shows two side-by-side screenshots of the Arduino IDE. Both windows have the title 'Koordinator433Mhz bisa | Arduino 1.8.13'. In the left window (a), the status bar at the bottom says 'Done compiling.' In the right window (b), the status bar also says 'Done compiling.' Both windows display the same code for a coordinator node, which includes definitions for RFM22B pins, serial communication, and timers. The code is identical in both windows.

Gambar 3.32 Tampilan *done compiling* pada koordinator

Sumber : Dokumentasi Penulis

5. Kemudian melakukan proses *Upload* dengan pilih ‘*Upload*’ pada *toolbar* Arduino, seperti pada Gambar 3.33.



The image shows two side-by-side screenshots of the Arduino IDE. Both windows have the title 'Koordinator433Mhz bisa | Arduino 1.8.13'. In the left window (a), the status bar at the bottom says 'Compiling sketch...'. In the right window (b), the status bar also says 'Compiling sketch...', and below it, a command line window shows the upload process: 'C:\Program Files (x86)\Arduino\arduino-builder -dump-prefs -logger-machine -hardware C:\Program' and 'C:\Program Files (x86)\Arduino\arduino-builder -dump-prefs -logger-machine -hardware C:\Program'. This indicates that the sketch is being compiled and then uploaded to the Arduino Uno.

Gambar 3.33 Tampilan proses *upload* pada koordinator

Sumber : Dokumentasi Penulis

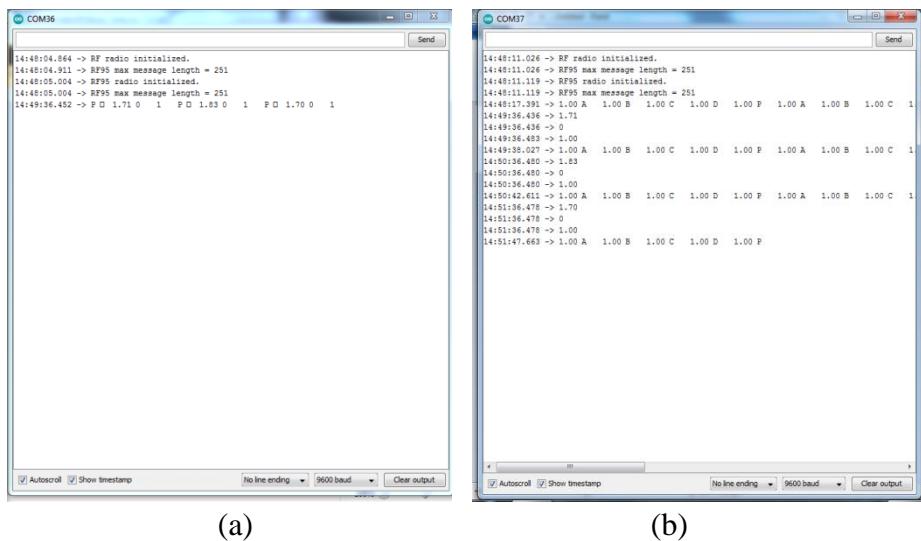
6. Apabila proses *Upload* sudah selesai ditandai dengan muncul ‘*Done uploading..*’ di *message-bar* Arduino, seperti pada Gambar 3.34.



Gambar 3.34 Tampilan *done uploading*

Sumber : Dokumentasi Penulis

7. Selanjutnya pilih ‘*Serial Monitor*’ pada *toolbar* Arduino untuk melakukan pantauan data yang diterima oleh LoRa Shield 915Mhz yang kemudian akan dikirimkan ke LoRa Shield 433Mhz menggunakan komunikasi serial, seperti pada Gambar 3.35.



(a)

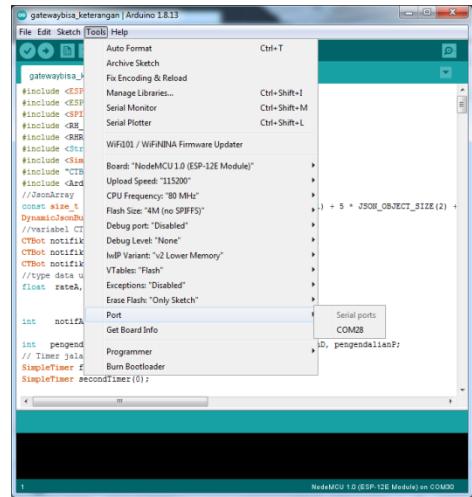
(b)

Gambar 3.35 Tampilan data *controlling* dan monitoring yang akan dikirim dan diterima ke sel gedung dan *gateway*

Sumber : Dokumentasi Penulis

c. *Upload program untuk Gateway*

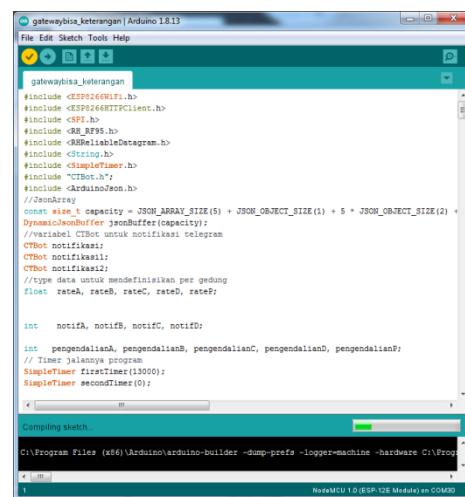
1. Membuka program sel gedung yang telah dibuat pada menu *File- Open* memilih file pada lokasi penyimpanan.
2. Selanjutnya memilih port COM yang telah terpasang pada menu *Tools- Port*, seperti pada Gambar 3.36.



Gambar 3.36 Tampilan port COM pada gateway

Sumber : Dokumentasi Penulis

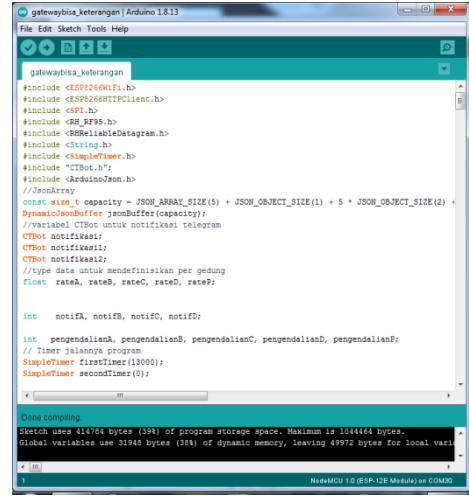
3. Melakukan proses *Compile* dengan pilih ‘*verify*’ pada toolbar Arduino seperti Gambar 3.37.



Gambar 3.37 Tampilan proses compile pada gateway

Sumber : Dokumentasi Penulis

4. Apabila proses *Compile* sudah selesai ditandai dengan muncul ‘*Done compiling...*’ di *message bar* Arduino, seperti Gambar 3.38

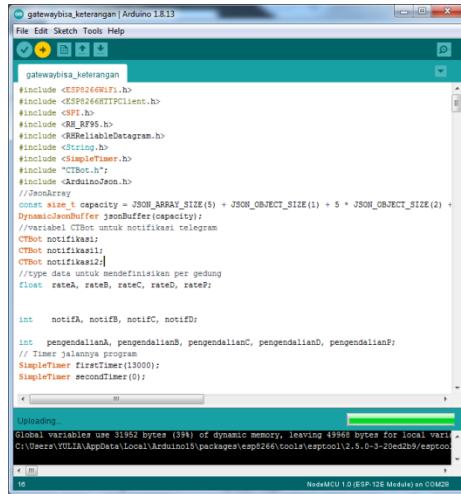


The screenshot shows the Arduino IDE interface with the sketch titled "gatewaybias\_keterangan". The code includes headers for ESP8266WiFi.h, WiFiClient.h, CFPT.h, RF24.h, and ReliableDatagram.h, along with simple timer and CTimer classes. The message bar at the bottom displays "Done compiling". Below the message bar, status information shows the sketch uses 414784 bytes (39%) of program storage space, leaving 1044444 bytes available. Global variables use 31948 bytes (38%) of dynamic memory, leaving 49972 bytes for local variables.

Gambar 3.38 Tampilan *done compiling* pada gateway

Sumber : Dokumentasi Penulis

5. Kemudian melakukan proses *Upload* dengan pilih ‘*Upload*’ pada *toolbar* Arduino, seperti pada Gambar 3.39.

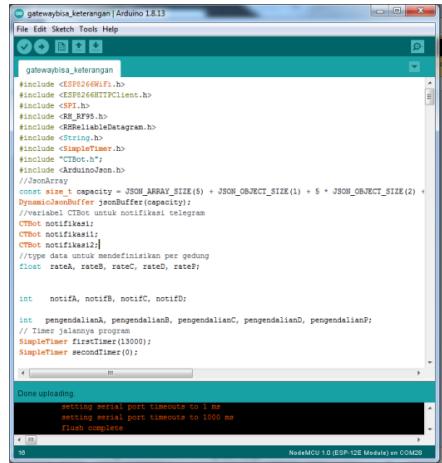


The screenshot shows the Arduino IDE interface with the sketch titled "gatewaybias\_keterangan". The code is identical to the one in Gambar 3.38. The message bar at the bottom displays "Uploading". Below the message bar, status information shows the sketch uses 31952 bytes (39%) of dynamic memory, leaving 49968 bytes for local variables. The path "C:\Users\YULIA\AppData\Local\Arduino15\packages\esp8266\tools\esp8266/tools/1.5.0-3-20ed2b9\esp8266" is visible.

Gambar 3.39 Tampilan proses *upload* pada gateway

Sumber : Dokumentasi Penulis

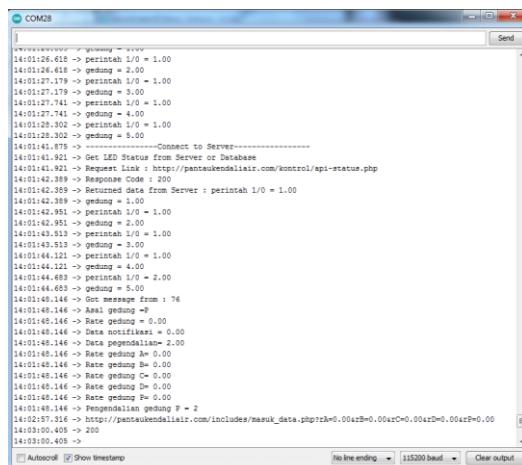
6. Apabila proses *Upload* sudah selesai ditandai dengan muncul ‘*Done uploading..*’ di *message-bar* Arduino, seperti pada Gambar 3.40.



Gambar 3.40 Tampilan *done uploading* pada gateway

Sumber: Dokumentasi Penulis

7. Selanjutnya pilih ‘*Serial Monitor*’ pada *toolbar* Arduino untuk melakukan pantauan data yang diterima oleh LoRa Bee yang kemudian akan dikirimkan ke Koordinator menggunakan media transmisi gelombang radio, dan pantauan data yang diterima LoRa Bee dari Koordinator yang selanjutnya akan di *upload* ke web, seperti pada Gambar 3.41.



Gambar 3.41 Tampilan data *controlling* dan monitoring pada *gateway*

Sumber: Dokumentasi Penulis

## **BAB IV**

### **PENGUJIAN DAN ANALISA**

Pada bab IV ini diuraikan mengenai hasil dan analisa pengujian dari sistem yang telah dirancang dan dibuat pada tugas akhir ini. Pengujian dilakukan untuk mengetahui bahwa sistem rancang bangun antara *node* pusat dan gedung, *node* koordinator dan *gateway* dapat berfungsi sesuai dengan perancangan sistem yang telah dibuat, serta bertujuan untuk mendapatkan data sehingga data tersebut dapat dianalisa.

#### **4.1 Metodologi Pengujian**

Metodologi pengujian merupakan rancangan pengujian perangkat sel pusat dan sel gedung, *node* koordinator dan *gateway* meliputi pengujian jarak cakupan wilayah sel gedung, koordinator dan *gateway*, pengujian pengiriman data sensor sel pusat dan sel gedung ke *node* koordinator, pengujian pengiriman data dari *node* koordinator ke *gateway*, pengujian pengiriman data dari *gateway* ke *database Web Server*, pengujian penerimaan data perintah dari *database Web Server* ke *gateway*, pengujian penerimaan data perintah dari *gateway* ke *node* koordinator, pengujian penerimaan data perintah dari *node* koordinator ke sel pusat atau sel gedung dan pengujian pengiriman notifikasi sel gedung ke *platform Telegram*.

Adapun penjelasan dari metodologi pengujian yang dilakukan sebagai berikut.

- a. Pengujian jarak cakupan wilayah sel gedung, *node* koordinator, dan *gateway*.

Pengujian jarak bertujuan untuk mengetahui sejauh mana LoRa *Shield* dan LoRa Bee dapat mengirim dan menerima data. Pengujian ini dilakukan pada beberapa lokasi yang berbeda dengan jarak antara sel gedung, Koordinator dan *gateway* bervariasi. Pengujian dinyatakan berhasil apabila LoRa *shield* pada sel gedung dengan LoRa *shield* pada koordinator dan LoRa Bee pada *gateway* dapat saling bertukar informasi.

- b. Pengujian pengiriman data sensor sel pusat dan gedung ke *node* koordinator.

Pengujian pengiriman data sensor sel pusat dan sel gedung ke *node* koordinator bertujuan untuk mengetahui data sensor *flowmeter* yang dikirim dari sel pusat dan sel gedung oleh LoRa *Shield* diterima oleh LoRa *Shield node* koordinator untuk dapat mengirim data informasi. Pengujian dikatakan berhasil ketika data yang dikirim sel pusat dan sel gedung ke *node* koordinator menerima semua data yang dikirim oleh sel pusat dan sel gedung.

- c. Pengujian pengiriman data dari *node* koordinator ke *gateway*.

Pengujian pengiriman data dari *node* koordinator ke *gateway* bertujuan untuk meneruskan data informasi yang di dapat LoRa *Shield* dan data yang diproses oleh mikrokontroler Arduino Uno *node* koordinator ke LoRa Bee *gateway*. Pengujian dikatakan berhasil ketika semua data sensor yang diterima *node* koordinator dari sel pusat dan sel gedung dapat dikirim ke *gateway*.

- d. Pengujian pengiriman data dari *gateway* ke *database Web Server*.

Pengujian pengiriman data dari *gateway* ke *database Web Server* bertujuan untuk mengirim data dari *gateway* yang didapatkan LoRa Bee ke *database Web Server* oleh *NodeMCU ESP8266*. Pengujian dikatakan berhasil ketika data yang dikirim oleh *NodeMCU ESP8266* pada *gateway* diterima oleh *database Web Server*.

- e. Pengujian penerimaan data perintah dari *database Web Server* ke *gateway*.

Pengujian penerimaan data perintah dari *database Web Server* ke *gateway* bertujuan untuk *gateway* menerima data yang berupa perintah dari *Web Server* yang kemudian data tersebut digunakan untuk melakukan pengendalian. Data perintah diterima diproses oleh *NodeMCU ESP8266*. Pengujian ini dikatakan berhasil ketika data yang di *input* dari *Web Server* dapat diterima oleh *gateway* yang digunakan untuk pengendalian.

- f. Pengujian penerimaan data perintah dari *gateway* ke *node* koordinator.

Pengujian penerimaan data perintah dari *gateway* ke *node* koordinator bertujuan untuk meneruskan data perintah yang di dapat dari komunikasi LoRa *Shield* pada *node* koordinator dengan LoRa Bee pada *gateway*. Data perintah yang didapatkan diproses oleh mikrokontroler Arhduino Uno yang kemudian ditunjukkan oleh alamat tujuan data tersebut dikirim. Perngujian ini dikatakan sukses ketika *node* koordinator menerima data perintah yang dikirim oleh *gateway*.

- g. Pengujian penerimaan data perintah dari *node* koordinator ke sel pusat atau sel gedung.

Pengujian penerimaan data perintah dari *node* koordinator ke sel pusat atau sel gedung bertujuan untuk sel pusat atau sel gedung menerima data yang dikirim oleh LoRa *Shield* pada *node* koordinator ke LoRa *Shield* sel pusat atau sel gedung yang kemudian data tersebut diproses oleh mikrokontroler Arduino Uno untuk mengendalikan katub selenoid terbuka atau tertutup. Pengujian ini dikatakan berhasil ketika data yang diterima sel pusat atau sel gedung diproses oleh mikrokontroler Arduino Uno dapat mengaktifkan atau menonaktifkan katub selenoid valve.

- h. Pengujian pengiriman notifikasi sel gedung ke *platform* Telegram.

Pengujian pengiriman notifikasi sel gedung ke *platform* Telegram bertujuan untuk mengirim notifikasi ke *platform* Telegram apabila terjadi kebocoran pada sel gedung tertentu dan mengirim notifikasi kondisi selenoid dengan kondisi katub menutup/*OFF*. Sel gedung akan mengirim notifikasi ke *node* koordinator kemudian ke *gateway* yang tersambung dengan *platform* Telegram. Pengujian ini dikatakan berhasil ketika *platform* menerima notifikasi dari *gateway* . Untuk notifikasi ke *platform* Telegram berhasil jika Telegram menerima notifikasi kondisi katub selenoid dalam keadaan *OFF*. Untuk notifikasi pengendalian kebocoran dikatakan berhasil jika *platform* Telegram menerima notifikasi kebocoran sesuai dengan waktu yang telah ditentukan yaitu lebih dari 10 menit. Jika sudah sesuai waktu yang disetting dan

dikatakan terjadi kebocoran maka sel gedung akan mengirim notifikasi ke *platform Telegram*.

i. Pengujian keakuratan data sensor *flowmeter*.

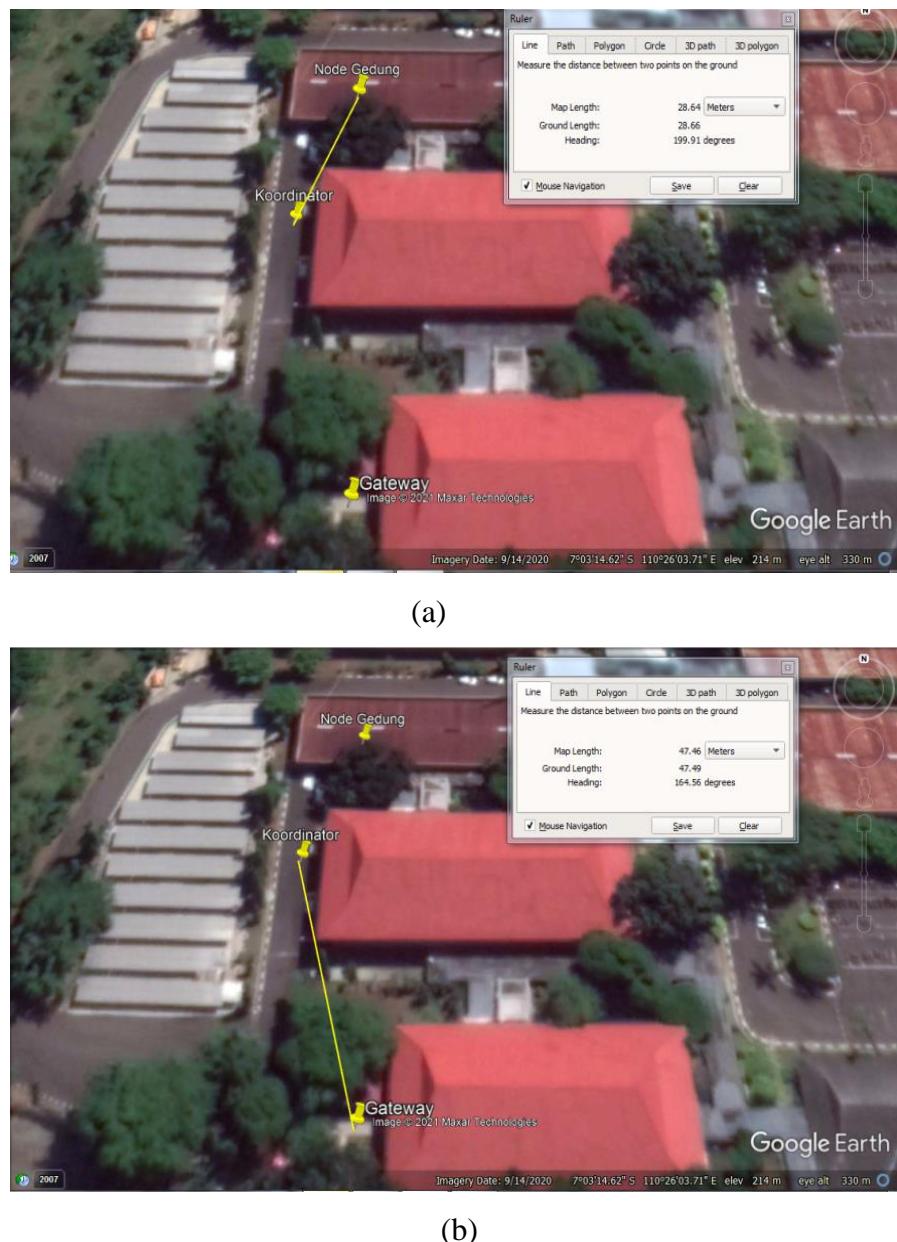
Pengujian keakuratan sensor *flowmeter* ini bertujuan untuk menguji keakuratan debit air pada *flowmeter* pada masing-masing sel. Debit akan dihitung oleh sensor *flowmeter* dengan rate yang berbeda-beda. Pengujian keakuratan debit air dihitung oleh sensor *flowmeter* antara sel pusat dengan sel gedung A,B,C dan D.

## 4.2 Hasil Pengujian dan Analisa

Dari tahapan pengujian sistem yang telah dilakukan di dapatkan hasil pengujian yang akan digunakan sebagai analisa.

### 4.2.1 Hasil pengujian jarak cakupan sel gedung, koordinator, dan *gateway*.

Pengujian jarak dilakukan sebanyak dua kali dan di tiga lokasi yang berbeda. Pengujian pertama dilakukan di Kampus Politeknik Negeri Semarang. Letak sel gedung berada di Lab Telkom Barat, letak Koordinator berada di Gedung SB, dan letak *gateway* berada di Gazebo Gedung SA. Jarak yang didapat antara sel gedung dengan koordinator adalah 28,66 meter, sedangkan jarak antara koordinator dengan *gateway* adalah 47,46 meter. Jadi jarak total yang dicapai dari *gateway* hingga sel gedung yaitu 76,12 meter. Letak sel gedung, koordinator, dan *gateway* dapat dilihat pada Gambar 4.1.



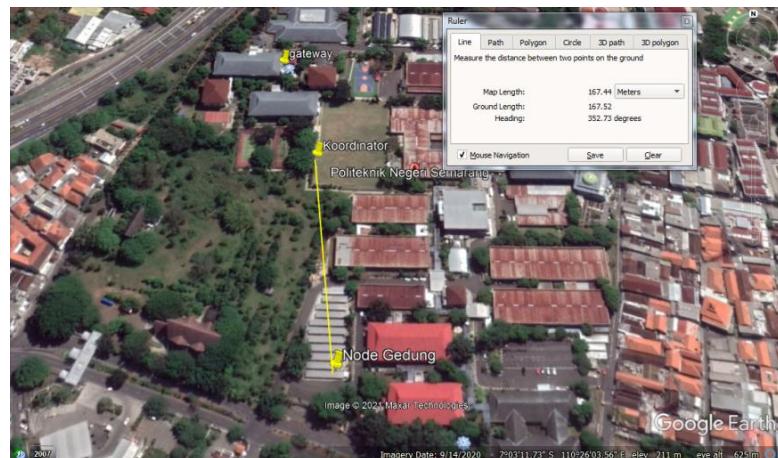
Gambar 4.1 Letak sel gedung, koordinator, dan *gateway* untuk pengujian pertama dilihat dari satelit

- (a) Jarak Sel Gedung dan Koordinator
- (b) Jarak Koordinator dan *Gateway*

Sumber: Dokumentasi Penulis

Pada pengujian kedua letak sel gedung berada di Parkiran Elektro, letak koordinator berada di Lapangan Hijau Politeknik Negeri Semarang, dan letak *gateway* berada di Gazebo Tata Niaga. Jarak yang didapat antara sel gedung dengan koordinator adalah 167,44

meter, sedangkan jarak antara koordinator dan *gateway* adalah 92,98 meter. Jadi jarak total yang dicapai dari *gateway* hingga sel gedung yaitu 260,42 meter. Letak sel gedung, koordinator, dan *gateway* dapat dilihat pada Gambar 4.2.



(a)



(b)

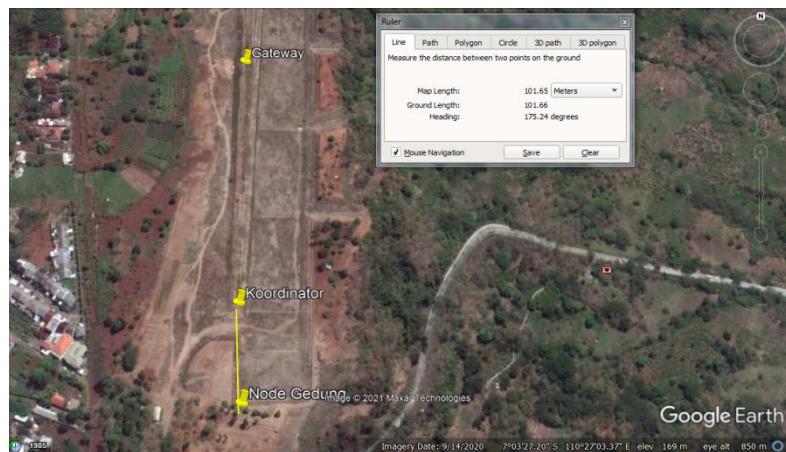
Gambar 4.2 Letak sel gedung, koordinator, dan *gateway* untuk pengujian kedua dilihat dari satelit  
pengujian ketiga letak sel gedung, letak koordinator, dan letak *gateway* berada di Lapangan Kodim Tembalang. Jarak yang didapat

- (a) Jarak Sel Gedung dan Koordinator
- (b) Jarak Koordinator dan *Gateway*

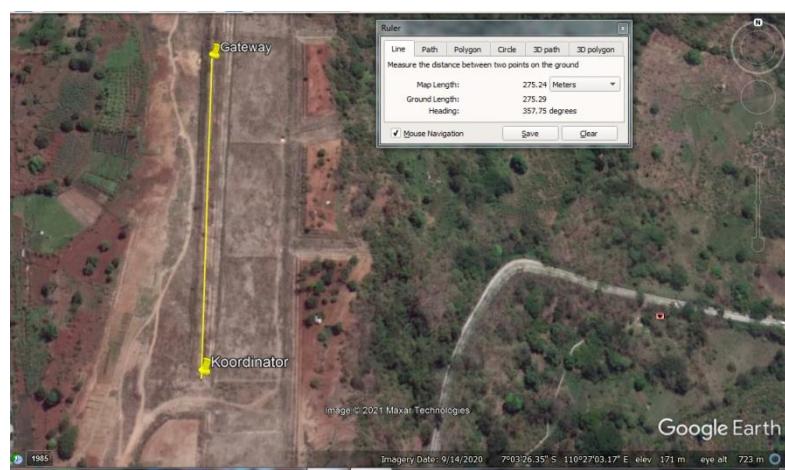
Sumber: Dokumentasi Penulis

Pada pengujian ketiga letak sel gedung, letak koordinator, dan letak *gateway* berada di Lapangan Kodim Tembalang. Jarak yang didapat antara sel gedung dengan koordinator adalah 101,65 meter,

sedangkan jarak antara koordinator dan *gateway* adalah 275,24 meter. Jadi jarak total yang dicapai dari *gateway* hingga sel gedung yaitu 376,89 meter. Letak sel gedung, koordinator, dan *gateway* dapat dilihat pada Gambar 4.3.



(a)



(b)

Gambar 4.3 Letak sel gedung, koordinator, dan *gateway* untuk pengujian ketiga dilihat dari satelit

(a) Jarak Sel Gedung dan Koordinator

(b) Jarak Koordinator dan *Gateway*

Sumber: Dokumentasi Penulis

#### 4.2.2 Pengujian pengiriman data sensor sel pusat dan gedung ke *node* koordinator.

Pengiriman data oleh sel pusat dan gedung merupakan data monitoring yang berasal dari perhitungan sensor *flowmeter* yang berada di masing-masing sel. Penggunaan air akan dihitung oleh sensor yang kemudian dikirim ke *node* koordinator 433 MHz yang kemudian diteruskan ke *node* koordinator 915 MHz.

Untuk hasil pengujian pengiriman data dari gedung ke *node* koordinator ditunjukkan Tabel 4.1.

Tabel 4.1 hasil pengujian data dari gedung ke *node* koordinator

No.	Sel	Data yang dikirim sel gedung ke koordinator 433 MHz dan koordinator 915 MHz	Data yang diterima koordinator 433 MHz dan koordinator 915 dari sel gedung
1.	Pusat	1. 0.00 Liter/Menit	P 0.00 0 1
		2. 0.00 Liter/Menit	P 0.00 0 1
		3. 0.07 Liter/Menit	P 0.07 0 1
		4. 5.62 Liter/Menit	P 5.62 0 1
		5. 4.56 Liter/Menit	P 4.56 0 1
		6. 4.62 Liter/Menit	P 4.62 0 1
		7. 4.28 Liter/Menit	P 4.28 0 1
		8. 4.28 Liter/Menit	P 4.28 0 1
2.	Gedung A	1. 0.00 Liter/Menit	A 0.00 0 1
		2. 0.95 Liter/Menit	A 0.95 0 1
		3. 0.04 Liter/Menit	A 0.04 0 1
		4. 1.08 Liter/Menit	A 1.08 0 1
		5. 2.71 Liter/Menit	A 2.71 0 1
		6. 2.69 Liter/Menit	A 2.69 0 1
		7. 2.70 Liter/Menit	A 2.70 0 1
		8. 2.72 Liter/Menit	A 2.72 0 1
3.	Gedung B	1. 0.00 Liter/Menit	B 0.00 0 1

		2. 2.00 Liter/Menit	B 2.00 0 1
		3. 2.55 Liter/Menit	B 2.55 0 1
		4. 2.55 Liter/Menit	B 2.55 0 1
		5. 2.54 Liter/Menit	B 2.54 0 1
		6. 2.54 Liter/Menit	B 2.54 0 1
		7. 2.52 Liter/Menit	B 2.52 0 1
		8. 2.53 Liter/Menit	B 2.55 0 1
4.	Gedung C	1. 0.00 Liter/Menit	C 0.00 0 1
		2. 1.24 Liter/Menit	C 1.24 0 1
		3. 4.47 Liter/Menit	C 4.47 0 1
		4. 4.55 Liter/Menit	C 4.55 0 1
		5. 4.61 Liter/Menit	C 4.61 0 1
		6. 4.60 Liter/Menit	C 4.60 0 1
		7. 4.69 Liter/Menit	C 4.69 0 1
		8. 4.74 Liter/Menit	C 4.74 0 1
5.	Gedung D	1. 0.00 Liter/Menit	D 0.00 0 1
		2. 1.80 Liter/Menit	D 1.80 0 1
		3. 2.65 Liter/Menit	D 2.65 0 1
		4. 2.67 Liter/Menit	D 2.67 0 1
		5. 2.70 Liter/Menit	D 2.70 0 1
		6. 2.73 Liter/Menit	D 2.73 0 1
		7. 2.74 Liter/Menit	D 2.74 0 1
		8. 2.73 Liter/Menit	D 2.73 0 1

Keterangan tabel 4.1 :

a. P 0.00 0 1

P = Sel pusat

0.0 = Jumlah debit air yang mengalir per menit (L/menit)

0 = Data notifikasi

1 = Data pengendalian

b. A 0.95 0 1

A = Sel gedung A

0.95 = Jumlah debit air yang mengalir per menit (L/menit)

0 = Data notifikasi

1 = Data pengendalian

c. B 2.00 0 1

C = Sel gedung B

2.00 = Jumlah debit air yang mengalir per menit (L/menit)

0 = Data notifikasi

1 = Data pengendalian

d. C 1.24 0 1

C = Sel gedung C

1.24 = Jumlah debit air yang mengalir per menit (L/menit)

0 = Data notifikasi

1 = Data pengendalian

e. D 1.80 0 1

D = Sel gedung D

1.80 = Jumlah debit air yang mengalir per menit (L/menit)

0 = Data notifikasi

1 = Data pengendalian

Tabel 4.1 adalah kumpulan data yang didapatkan ketika melakukan pengujian pada masing-masing sel yang terdiri dari 4 sel gedung dan 1 sel pusat. Pada kolom ketiga merupakan data yang didapatkan dari sensor *flowmeter* yang terdapat pada masing-masing sel. Terjadi penambahan atau pengurangan debit pada masing-masing sel tergantung pada kecepatan penggunaan air yang digunakan. . Data sensor yang terdapat pada tabel yang berisikan debit bernilai 0 menandakan bahwa keran air yang terdapat pada gedung tidak mengalir, sehingga *flowmeter* membaca debit bernilai 0 karena tidak ada air yang mengalir dalam sensor *flowmeter*. Pada kolom keempat adalah data yang diterima oleh *node* koordinator 433 MHz dan 915

MHz. Dari hasil percobaan data yang dikirim masing-masing sel ke *node* koordinator 433 MHz dan *node* koordinator 915 MHz adalah sama.

#### **4.2.3 Pengujian pengiriman data dari *node* koordinator ke *gateway*.**

Pengiriman data oleh koordinator merupakan pengiriman data *monitoring* dari masing-masing sel yang merupakan data perhitungan dari sensor *flowmeter*, data notifikasi, dan data pengendalian. Setelah data penggunaan air diterima oleh koordinator, data penggunaan air tersebut dikirimkan ke *gateway* untuk dikirimkan ke *Web Server*.

Untuk hasil pengujian pengiriman data dari *node* koordinator ke *gateway* ditunjukkan Tabel 4.2.

Tabel 4.2 Hasil pengujian pengiriman data dari *node* koordinator ke *gateway*

No	Data yang dikirim Koordinator ke <i>Gateway</i>	Data yang diterima <i>Gateway</i> dari Koordinator 915Mhz
1.	P 0.00 0 1	Asal gedung : P Rate gedung : 0.00 L/Menit Data notifikasi : 0 Data pengendalian : 1
2.	P 0.00 0 1	Asal gedung : P Rate gedung : 0.00 L/Menit Data notifikasi : 0 Data pengendalian : 1
3.	P 0.07 0 1	Asal gedung : P Rate gedung : 0.07 L/Menit Data notifikasi : 0 Data pengendalian : 1
4.	P 5.62 0 1	Asal gedung : P Rate gedung : 5.62 L/Menit

		Data notifikasi : 0 Data pengendalian : 1
5.	P 4.56 0 1	Asal gedung : P Rate gedung : 4.56 L/Menit Data notifikasi : 0 Data pengendalian : 1
6.	P 4.62 0 1	Asal gedung : P Rate gedung : 4.62 L/Menit Data notifikasi : 0 Data pengendalian : 1
7.	P 4.28 0 1	Asal gedung : P Rate gedung : 4.28 L/Menit Data notifikasi : 0 Data pengendalian : 1
8.	P 4.28 0 1	Asal gedung : P Rate gedung : 4.28 L/Menit Data notifikasi : 0 Data pengendalian : 1
9.	A 0.00 0 1	Asal gedung : A Rate gedung : 0.00 L/Menit Data notifikasi : 0 Data pengendalian : 1
10.	A 0.95 0 1	Asal gedung : A Rate gedung : 0.95 L/Menit Data notifikasi : 0 Data pengendalian : 1
11	A 0.04 0 1	Asal gedung : A Rate gedung : 0.04 L/Menit Data notifikasi : 0 Data pengendalian : 1
12.	A 1.08 0 1	Asal gedung : A Rate gedung : 1.08 L/Menit

		Data notifikasi : 0 Data pengendalian : 1
13.	A 2.71 0 1	Asal gedung : A Rate gedung : 2.71 L/Menit Data notifikasi : 0 Data pengendalian : 1
14.	A 2.69 0 1	Asal gedung : A Rate gedung : 2.69 L/Menit Data notifikasi : 0 Data pengendalian : 1
15.	A 2.70 0 1	Asal gedung : A Rate gedung : 2.70 L/Menit Data notifikasi : 0 Data pengendalian : 1
16.	A 2.72 0 1	Asal gedung : A Rate gedung : 2.72 L/Menit Data notifikasi : 0 Data pengendalian : 1
17.	B 0.00 0 1	Asal gedung : B Rate gedung : 0.00 L/Menit Data notifikasi : 0 Data pengendalian : 1
18.	B 2.00 0 1	-
19.	B 2.55 0 1	-
20.	B 2.55 0 1	Asal gedung : B Rate gedung : 2.55 L/Menit Data notifikasi : 0 Data pengendalian : 1
21.	B 2.54 0 1	Asal gedung : B Rate gedung : 2.54 L/Menit Data notifikasi : 0

		Data pengendalian : 1
22	B 2.54 0 1	Asal gedung : B Rate gedung : 2.54 L/Menit Data notifikasi : 0 Data pengendalian : 1
23.	B 2.52 0 1	Asal gedung : B Rate gedung : 2.52 L/Menit Data notifikasi : 0 Data pengendalian : 1
24	B 2.55 0 1	Asal gedung : B Rate gedung : 2.55 L/Menit Data notifikasi : 0 Data pengendalian : 1
25.	C 0.00 0 1	Asal gedung : C Rate gedung : 0.00 L/Menit Data notifikasi : 0 Data pengendalian : 1
26	C 1.24 0 1	Asal gedung : C Rate gedung : 1.24 L/Menit Data notifikasi : 0 Data pengendalian : 1
27.	C 4.47 0 1	Asal gedung : C Rate gedung : 4.47 L/Menit Data notifikasi : 0 Data pengendalian : 1
28.	C 4.55 0 1	Asal gedung : C Rate gedung : 4.55 L/Menit Data notifikasi : 0 Data pengendalian : 1
29.	C 4.61 0 1	Asal gedung : C Rate gedung : 4.61 L/Menit Data notifikasi : 0

		Data pengendalian : 1
30.	C 4.60 0 1	Asal gedung : C Rate gedung : 4.60 L/Menit Data notifikasi : 0 Data pengendalian : 1
31.	C 4.69 0 1	Asal gedung : C Rate gedung : 4.69 L/Menit Data notifikasi : 0 Data pengendalian : 1
32.	C 4.74 0 1	Asal gedung : C Rate gedung : 4.74 L/Menit Data notifikasi : 0 Data pengendalian : 1
33.	D 0.00 0 1	Asal gedung : D Rate gedung : 0.00 L/Menit Data notifikasi : 0 Data pengendalian : 1
34.	D 1.80 0 1	Asal gedung : D Rate gedung : 1.80 L/Menit Data notifikasi : 0 Data pengendalian : 1
35.	D 2.65 0 1	Asal gedung : D Rate gedung : 2.65 L/Menit Data notifikasi : 0 Data pengendalian : 1
36.	D 2.67 0 1	Asal gedung : D Rate gedung : 2.67 L/Menit Data notifikasi : 0 Data pengendalian : 1
37.	D 2.70 0 1	Asal gedung : D Rate gedung : 2.70 L/Menit Data notifikasi : 0

		Data pengendalian : 1
38.	D 2.73 0 1	Asal gedung : D Rate gedung : 2.73 L/Menit Data notifikasi : 0 Data pengendalian : 1
39.	D 2.74 0 1	Asal gedung : D Rate gedung : 2.74 L/Menit Data notifikasi : 0 Data pengendalian : 1
40.	D 2.73 0 1	Asal gedung : D Rate gedung : 2.73 L/Menit Data notifikasi : 0 Data pengendalian : 1

Tabel 4.2 adalah data hasil pengujian yang diterima dari sel pusat dan sel gedung ke *node* koordinator dan kemudian dikirimkan ke *gateway*. Pada tabel kolom kedua adalah data yang diterima *node* koordinator dari sensor *flowmeter* yang berada pada masing-masing sel. Data sensor *flowmeter* masing-masing sel mengirimkan data *rate* atau kecepatan air ke *node* koordinator setiap satu menit sekali dalam satuan Liter/Menit. Kemudian pada kolom ketiga adalah data yang diterima *gateway* yang dikirimkan oleh *node* koordinator. Data yang diterima *gateway* pada tabel tersebut terlihat ada yang terlewat atau data yang dikirimkan *node* koordinator tidak diterima oleh *gateway* dikarenakan waktu pengiriman bersamaan dengan data perintah dan atau informasi notifikasi yang mengakibatkan data terlewat atau tidak diterima di *gateway*.

#### 4.2.4 Pengujian pengiriman data dari *gateway* ke *database Web Server*.

Pengujian pengiriman data dari *gateway* ke *database Web Server* merupakan pengiriman data sensor *flowmeter* yang berasal dari sel gedung. Data sensor tersebut dikirimkan ke *database Web Server*

digunakan sebagai data monitoring penggunaan air. Untuk hasil pengujian pengiriman data dari *gateway* ke *database Web Server* dapat ditunjukkan pada Gambar 4.4.

```

COM28
11:20:19.692 -> http://pantaukendalair.com/includes/masuk_data.php?rB=1.36
11:20:22.851 -> 200
11:20:22.851 -> SELECT ROUND(volume, 2) vol FROM volume WHERE id_gedung = '2' ORDER BY :
11:20:41.621 -> Got message from : 76
11:20:41.621 -> Asal gedung =A
11:20:41.621 -> Rate gedung = 2.78
11:20:41.621 -> Data notifikasi = 0.00
11:20:41.621 -> Data pegendalian= 1.00
11:20:41.621 -> Rate gedung A= 2.78
11:20:41.621 -> Notif gedung A = 0
11:20:41.621 -> Pengendalian gedung A = 1
11:20:41.621 -> http://pantaukendalair.com/includes/masuk_data.php?rA=2.78
11:20:44.582 -> 200
11:20:44.582 -> SELECT ROUND(volume, 2) vol FROM volume WHERE id_gedung = '1' ORDER BY :
11:20:51.055 -> Got message from : 76
11:20:51.055 -> Asal gedung =C
11:20:51.055 -> Rate gedung = 2.03
11:20:51.055 -> Data notifikasi = 0.00
11:20:51.055 -> Data pegendalian= 1.00
11:20:51.055 -> Rate gedung C= 2.03
11:20:51.055 -> Notif gedung C = 0
11:20:51.055 -> Pengendalian gedung C = 1
11:20:51.055 -> http://pantaukendalair.com/includes/masuk_data.php?rC=2.03
11:20:54.186 -> 200
11:20:54.186 -> SELECT ROUND(volume, 2) vol FROM volume WHERE id_gedung = '3' ORDER BY :
11:21:01.702 -> Got message from : 76
11:21:01.702 -> Asal gedung =D
11:21:01.702 -> Rate gedung = 0.09
11:21:01.702 -> Data notifikasi = 0.00
11:21:01.702 -> Data pegendalian= 1.00
11:21:01.702 -> Rate gedung D= 0.09
11:21:01.702 -> Notif gedung D = 0
11:21:01.702 -> Pengendalian gedung D = 1
11:21:01.702 -> http://pantaukendalair.com/includes/masuk_data.php?rD=0.09
11:21:04.907 -> 200
11:21:04.907 -> SELECT ROUND(volume, 2) vol FROM volume WHERE id_gedung = '4' ORDER BY :
11:21:08.063 -> -----Connect to Server-----
11:21:08.063 -> Get LED Status from Server or Database

```

Penerimaan data dari  
node koordinator 915  
MHz Gedung A

Penerimaan data dari  
node koordinator 915  
MHz Gedung C

Penerimaan data dari  
node koordinator 915  
MHz Gedung D

Gambar 4.4 Gambar tampilan serial monitor pengiriman data dari *gateway* ke *database Web Server*

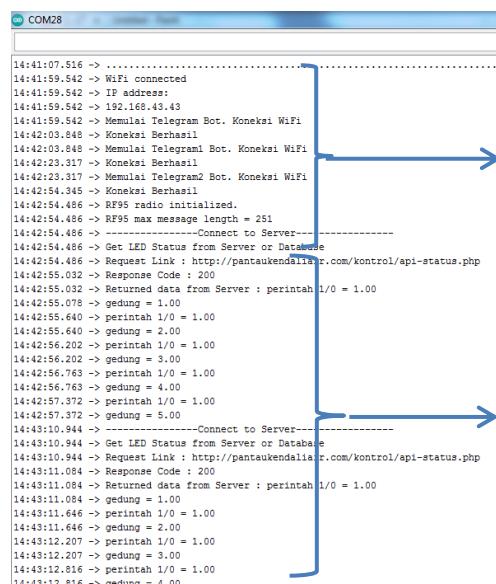
Sumber : Dokumentasi Penulis

Pada gambar 4.4 menunjukkan tampilan serial monitor pada *gateway*. Data yang diterima dari *gateway* akan diteruskan ke *database web server*. Data tersebut akan ditangkap dan diterima oleh *database web server* yang telah dikelompokkan dan diatur oleh admin *web server* tersebut. Tampilan tersebut berupa data sensor *flowmeter* masing-masing gedung dan data pengiriman notifikasi dari masing-masing gedung. Tampilan tersebut juga menampilkan data notifikasi perintah pengendalian dari masing-masing gedung yang digunakan untuk mengkonfirmasi keadaan katub solenoid

valve. Data yang dikirim oleh *gateway* berupa data-data sensor *flowmeter* dan data notifikasi pengendalian sel.

#### 4.2.5 Pengujian penerimaan data perintah dari *database Web Server* ke *gateway*.

Pengujian penerimaan data perintah dari *database Web Server* ke *gateway* merupakan permintaan *gateway* mengambil data dari *database*, data tersebut berupa perintah yang digunakan untuk melakukan pengendalian ke sel-sel gedung. Data tersebut diterima *gateway* yang selanjutnya akan dikirimkan ke koordinator dan dilanjutkan ke sel-sel gedung. Untuk hasil pengujian penerimaan data perintah dari *database Web Server* ke *gateway* dapat ditunjukkan pada Gambar 4.5.



```

COM28
14:41:07.516 -> .....
14:41:59.542 -> WiFi connected
14:41:59.542 -> IP address: 192.168.43.43
14:41:59.542 -> Memulai Telegram Bot. Koneksi WiFi
14:42:03.848 -> Koneksi Berhasil
14:42:03.848 -> Memulai Telegram Bot. Koneksi WiFi
14:42:23.317 -> Koneksi Berhasil
14:42:23.317 -> Memulai Telegram2 Bot. Koneksi WiFi
14:42:34.345 -> Koneksi Berhasil
14:42:34.345 -> RF95 radio initialized.
14:42:34.486 -> RF95 max message length = 251
14:42:34.486 -> _____Connect to Server_____
14:42:34.486 -> Get LED Status from Server or Database
14:42:34.486 -> Request Link : http://pantaukendaliir.com/kontrol/api-status.php
14:42:55.032 -> Response Code : 200
14:42:55.032 -> Returned data from Server : perintah 1/0 = 1.00
14:42:55.078 -> gedung = 1.00
14:42:55.640 -> perintah 1/0 = 1.00
14:42:55.640 -> gedung = 2.00
14:42:55.202 -> perintah 1/0 = 1.00
14:42:55.202 -> gedung = 3.00
14:42:55.763 -> perintah 1/0 = 1.00
14:42:55.763 -> gedung = 4.00
14:42:57.372 -> perintah 1/0 = 1.00
14:42:57.372 -> gedung = 5.00
14:43:10.944 -> _____Connect to Server_____
14:43:10.944 -> Get LED Status from Server or Database
14:43:10.944 -> Request Link : http://pantaukendaliir.com/kontrol/api-status.php
14:43:11.084 -> Response Code : 200
14:43:11.084 -> Returned data from Server : perintah 1/0 = 1.00
14:43:11.084 -> gedung = 1.00
14:43:11.646 -> perintah 1/0 = 1.00
14:43:11.646 -> gedung = 2.00
14:43:12.207 -> perintah 1/0 = 1.00
14:43:12.207 -> gedung = 3.00
14:43:12.816 -> perintah 1/0 = 1.00
14:43:12.816 -> gedung = 4.00

```

Memulai koneksi WiFi, Telegram, dan Inisialisasi LoRa

Request Link ke web dan mendapatkan data dari database berupa data perintah pengendalian

Gambar 4.5 Tampilan serial monitor pengujian penerimaan data perintah dari *database* ke *gateway*

Sumber : Dokumentasi Penulis

Keterangan:

Perintah 1 → ON

Perintah 2 → OFF

Gedung 1 → Gedung A

Gedung 2 → Gedung B

Gedung 3 → Gedung C

Gedung 4 → Gedung D

Gedung 5 → Pusat

Pada gambar 4.5 merupakan gambar tampilan serial monitor hasil pengujian dari data perintah dari *database web server*. *web server* pada hasil percobaan ini akan memberikan data perintah berupa *on* atau *off* untuk katub selenoid pada masing-masing sel yang dikirimkan ke *gateway* berupa perintah “1” atau “2”. Perintah “1” diberikan ketika untuk membuka katub selenoid yang tertutup. Sedangkan perintah “2” diberikan ketika untuk menutup katub selenoid yang terbuka. Perintah tersebut akan diterima oleh *gateway* yang kemudian akan dikirimkan ke masing-masing sel dengan melalui *node* koordinator.

#### **4.2.6 Pengujian penerimaan data perintah dari *gateway* ke *node* koordinator.**

Pengujian penerimaan data perintah dari *gateway* ke *node* koordinator merupakan pengiriman data perintah dari *gateway* yang akan diterima oleh koordinator, selanjutnya data perintah tersebut diteruskan ke sel gedung untuk mengendalikan katub solenoid valve. Untuk hasil pengujian penerimaan data perintah dari *gateway* ke *node* koordinator ditunjukkan Tabel 4.3.

Tabel 4.3 Hasil pengujian penerimaan data perintah dari *gateway* ke *node* koordinator

No	Data yang dikirim <i>gateway</i>	Data yang diterima Koordinator 915Mhz	Data yang diterima Koordinator 433Mhz
1.	11	1A	1A
2.	12	1B	1B
3.	13	1C	1C
4.	14	1D	1D
5.	15	1P	1P
6.	21	2A	2A
7.	22	2B	2B
8.	23	2C	2C
9.	24	2D	2D
10.	25	2P	2P

Pada tabel 4.3 merupakan pengambilan data yang diambil dari hasil pengujian *gateway* ke *node* koordinator. *Gateway* yang mendapat data berupa perintah dari *database web server* kemudian diolah dan dikirimkan ke *node* koordinator. Data perintah yang dikirim *gateway* berupa 11, 12, 13, 14 dan 15 serta 21, 22, 23, 24 dan 25. Data perintah yang mempunyai awalan “1” berarti menyatakan bahwa selenoid dalam kondisi *on* atau katub selenoid terbuka dan jika data perintah memiliki awalan “2” berarti menyatakan bahwa selenoid dalam kondisi *off* atau katub selenoid tertutup. Kemudian angka kedua menandakan simbol masing-masing sel. Angka “1” menunjukkan sel gedung A, angka “2” menunjukkan sel gedung B, angka “3” menunjukkan sel gedung C, angka “4” menunjukkan sel gedung “D” dan angka “5” menunjukkan sel pusat. Pada *node* koordinator 915 MHz angka-angka yang dikirimkan *gateway* dikonversi pada bit kedua menjadi 1A, 1B, 1C, 1D dan 1P serta 2A, 2B, 2C, 2D dan 2P. Kemudian data yang dikonversi oleh *node*

koordinator 915 MHz dikirimkan ke *node* koordinator 433 MHz dengan data yang sama pada *node* koordinator 915 MHz.

Keterangan :

- a. 11 / 1A = sel gedung A selenoid *ON*
- b. 12 / 1B = sel gedung B selenoid *ON*
- c. 13 / 1C = sel gedung C selenoid *ON*
- d. 14 / 1D = sel gedung D selenoid *ON*
- e. 15 / 1P = sel pusat selenoid *ON*
- f. 21 / 2A = sel gedung A selenoid *OFF*
- g. 22 / 2B = sel gedung B selenoid *OFF*
- h. 23 / 2C = sel gedung C selenoid *OFF*
- i. 24 / 2D = sel gedung D selenoid *OFF*
- j. 25 / 2P = sel pusat selenoid *OFF*

#### **4.2.7 Pengujian penerimaan data perintah dari *node* koordinator ke sel pusat atau sel gedung.**

Penerimaan data perintah dari *node* koordinator ke sel pusat atau sel gedung merupakan pengiriman data perintah ke sel pusat atau sel gedung yang digunakan untuk mengendalikan katub solenoid valve. Data perintah tersebut dikirim ke sel pusat atau sel gedung sesuai dengan inisialisasi tiap gedungnya. Untuk hasil pengujian pengiriman data perintah dari *node* koordinator ke sel pusat atau sel gedung dapat ditunjukkan Tabel 4.4.

Tabel 4.4 Pengujian penerimaan data perintah dari *node* koordinator ke sel pusat atau sel gedung

No	Data yang dikirim Koordinator	Data yang diterima Gedung A	Data yang diterima Gedung B	Data yang diterima GedungC	Data yang diterima GedungD	Data diterima Pusat
1.	1A1B1C1D1P	1A	1B	1C	1D	1P
2.	2A1B1C1D1P	2A	1B	1C	1D	1P
3.	1A2B1C1D1P	1A	2B	1C	1D	1P
4.	1A1B2C1D1P	1A	1B	2C	1D	1P
5.	1A1B1C2D1P	1A	1B	1C	2D	1P
6.	1A1B1C1D2P	1A	1B	1C	1D	2P
7.	2A2B2C2D2P	2A	2B	2C	2D	2P

Tabel 4.4 menampilkan informasi data pengujian pengiriman data dari *node* koordinator ke masing-masing sel. Pada kolom 2 pada tabel merupakan sekumpulan data yang diterima *node* koordinator untuk diteruskan ke masing- masing sel. Pada kolom 3 sampai 7 merupakan data perintah pengendalian yang didapatkan dari data yang dikirim *node* koordinator ke masing-masing gedung. Jika sel menerima data perintah “1” maka sel akan memberikan perintah kepada selenoid valve untuk kondisi katub membuka atau *ON*. jika sel menerima data perintah “2” maka sel akan memberikan perintah pada selenoid untuk kondisi katub menutup atau *OFF*.

Keterangan :

- a. 1A = Sel gedung A selenoid *ON* / katub selenoid membuka
- b. 1B = Sel gedung B selenoid *ON* / katub selenoid membuka
- c. 1C = Sel gedung C selenoid *ON* / katub selenoid membuka
- d. 1D = Sel gedung D selenoid *ON* / katub selenoid membuka
- e. 1P = Sel pusat selenoid *ON* / katub selenoid membuka
- f. 2A = Sel gedung A selenoid *OFF* / katub selenoid menutup
- g. 2B = Sel gedung B selenoid *OFF* / katub selenoid menutup
- h. 2C = Sel gedung C selenoid *OFF* / katub selenoid menutup

- i. 2D = Sel gedung D selenoid *OFF* / katub selenoid menutup
- j. 2P = Sel pusat selenoid *OFF* / katub selenoid menutup

#### **4.2.8 Pengujian pengiriman notifikasi sel gedung ke *platform Telegram*.**

Pengiriman data notifikasi dari masing-masing sel ke *platform Telegram* adalah komunikasi yang digunakan untuk memberikan kemudahan bagi pengguna jika terjadi *trouble* pada sistem dan agar pengguna dapat mengetahui status selenoid. Pengiriman data dilakukan masing-masing sel jika mengalami dua keadaan. Keadaan pertama, masing-masing sel mengirim notifikasi ke *platform Telegram* ketika katub selenoid menutup atau kondisi *OFF*. Keadaan kedua, jika terjadi kebocoran selama waktu 10 menit maka masing-masing sel akan mengirim perintah berupa informasi jika sel terjadi kebocoran pada *platform Telegram*. Tampilan notifikasi pada *platform Telegram* ditunjukkan pada gambar 4.6



Gambar 4.6 Tampilan notifikasi Telegram

Sumber : Dokumentasi Penulis

Gambar 4.6 Menunjukkan tampilan notifikasi pada *platform Telegram*. Notifikasi berasal dari data yang dikirim dari masing-masing sel ke *node* koordinator dan *gateway*, lalu *gateway* mengirimkan notifikasi tersebut ke *platform Telegram* jika data notifikasi kebocoran yang diterima adalah "1" dan data pengendalian yang diterima adalah "2". Jika data notifikasi "1" maka *gateway* akan mengirim notifikasi ke *Telegram* berisikan " Ada Kebocoran di Gedung A/B/C/D". Dan pada saat data pengendalian yang diterima

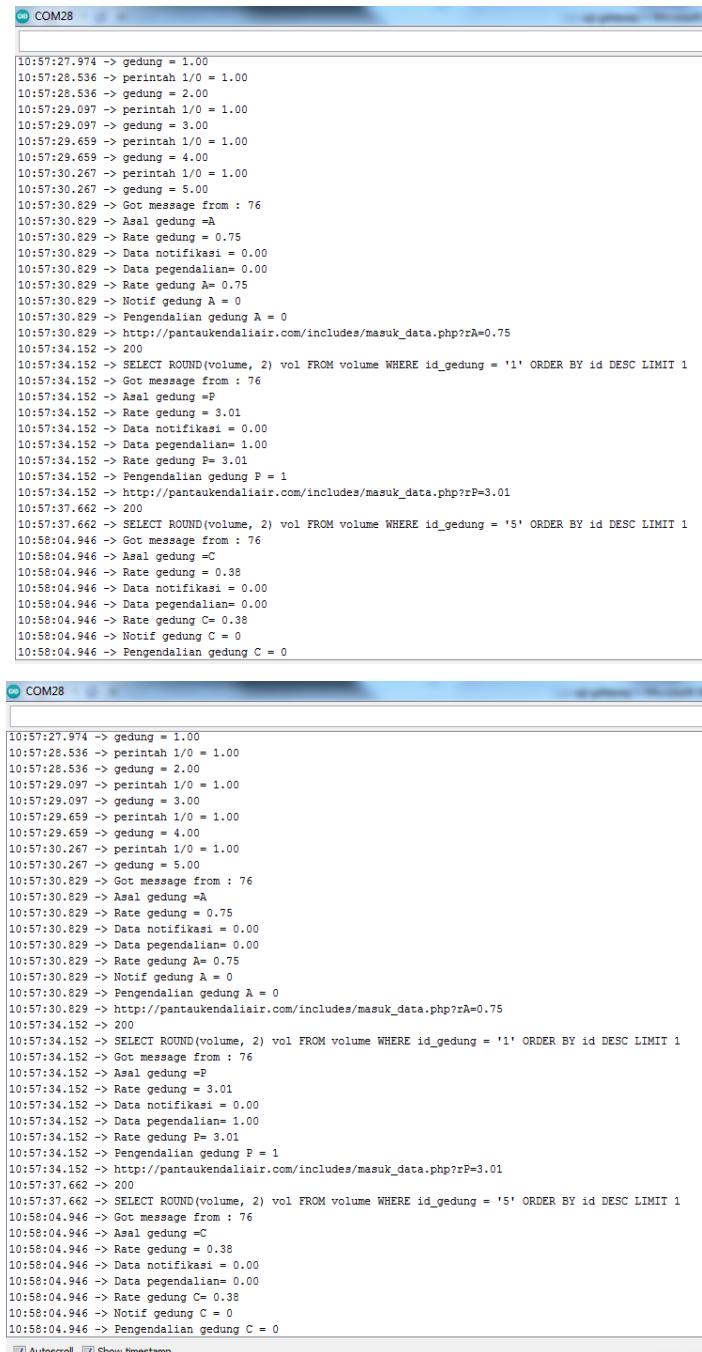
adalah “2” *gateway* mengirim notifikasi ke Telegram yang berupa konfirmasi "Selenoid Gedung A/B/C/D/P OFF(air mati)".

#### **4.2.9 Pengujian keakuratan data sensor *flowmeter*.**

Keakuratan data sensor *flowmeter* diuji untuk mengetahui seberapa akurat data air sesungguhnya dengan data perhitungan menggunakan sensor *flowmeter*. Data tersebut diambil dari masing-masing sel. Untuk pembanding pada pengujian ini digunakan antara sel pusat dengan sel gedung A,B,C dan D. Untuk hasil pengujian keakuratan data sensor *flowmeter* dapat ditunjukkan Tabel 4.5. dan gambar 4.7 adalah tampilan serial monitor pada *gateway* berupa data monitoring dari *node* sel.

Tabel 4.5 Hasil pengujian keakuratan data sensor *flowmeter*

No	Debit Gedung Pusat (L/min)	Debit Gedung A (L/min)	Debit Gedung B (L/min)	Debit Gedung C (L/min)	Debit Gedung D (L/min)	Perbandingan gedung pusat dengan gedung A,B,C,dan D (L/min)
1.	2.10	0.00	1.87	0.00	0.00	P (+0.23) > ABCD
2.	2.58	0.00	3.10	0.00	0.00	P (-0.52) < ABCD
3.	2.59	0.00	3.09	0.00	0.00	P (-0.5) < ABCD
4.	6.23	1.36	2.24	1.06	0.65	P (+0.92) > ABCD
5.	3.01	0.75	0.20	1.41	0.87	P (+0.22) > ABCD
6.	0.40	0.00	0.00	0.38	0.00	P (+0.02) > ABCD
7.	2.80	0.56	0.15	0.89	0.33	P (+0.87) > ABCD
8.	4.43	1.33	0.52	1.47	0.52	P (+0.59) > ABCD
9.	4.70	1.40	1.36	1.56	0.51	P (-0.13) < ABCD
10.	3.91	1.33	0.15	1.37	0.43	P (+0.63) > ABCD



```

COM28
10:57:27.974 -> gedung = 1.00
10:57:28.536 -> perintah 1/0 = 1.00
10:57:28.536 -> gedung = 2.00
10:57:29.097 -> perintah 1/0 = 1.00
10:57:29.097 -> gedung = 3.00
10:57:29.659 -> perintah 1/0 = 1.00
10:57:29.659 -> gedung = 4.00
10:57:30.267 -> perintah 1/0 = 1.00
10:57:30.267 -> gedung = 5.00
10:57:30.829 -> Got message from : 76
10:57:30.829 -> Asal gedung =A
10:57:30.829 -> Rate gedung = 0.75
10:57:30.829 -> Data notifikasi = 0.00
10:57:30.829 -> Data pegendalian= 0.00
10:57:30.829 -> Rate gedung A= 0.75
10:57:30.829 -> Notif gedung A = 0
10:57:30.829 -> Pengendalian gedung A = 0
10:57:30.829 -> http://pantaukendalialair.com/includes/masuk_data.php?rA=0.75
10:57:34.152 -> 200
10:57:34.152 -> SELECT ROUND(volume, 2) vol FROM volume WHERE id_gedung = '1' ORDER BY id DESC LIMIT 1
10:57:34.152 -> Got message from : 76
10:57:34.152 -> Asal gedung =P
10:57:34.152 -> Rate gedung = 3.01
10:57:34.152 -> Data notifikasi = 0.00
10:57:34.152 -> Data pegendalian= 1.00
10:57:34.152 -> Rate gedung P= 3.01
10:57:34.152 -> Pengendalian gedung P = 1
10:57:34.152 -> http://pantaukendalialair.com/includes/masuk_data.php?rP=3.01
10:57:37.662 -> 200
10:57:37.662 -> SELECT ROUND(volume, 2) vol FROM volume WHERE id_gedung = '5' ORDER BY id DESC LIMIT 1
10:58:04.946 -> Got message from : 76
10:58:04.946 -> Asal gedung =C
10:58:04.946 -> Rate gedung = 0.38
10:58:04.946 -> Data notifikasi = 0.00
10:58:04.946 -> Data pegendalian= 0.00
10:58:04.946 -> Rate gedung C= 0.38
10:58:04.946 -> Notif gedung C = 0
10:58:04.946 -> Pengendalian gedung C = 0

COM28
10:57:27.974 -> gedung = 1.00
10:57:28.536 -> perintah 1/0 = 1.00
10:57:28.536 -> gedung = 2.00
10:57:29.097 -> perintah 1/0 = 1.00
10:57:29.097 -> gedung = 3.00
10:57:29.659 -> perintah 1/0 = 1.00
10:57:29.659 -> gedung = 4.00
10:57:30.267 -> perintah 1/0 = 1.00
10:57:30.267 -> gedung = 5.00
10:57:30.829 -> Got message from : 76
10:57:30.829 -> Asal gedung =A
10:57:30.829 -> Rate gedung = 0.75
10:57:30.829 -> Data notifikasi = 0.00
10:57:30.829 -> Data pegendalian= 0.00
10:57:30.829 -> Rate gedung A= 0.75
10:57:30.829 -> Notif gedung A = 0
10:57:30.829 -> Pengendalian gedung A = 0
10:57:30.829 -> http://pantaukendalialair.com/includes/masuk_data.php?rA=0.75
10:57:34.152 -> 200
10:57:34.152 -> SELECT ROUND(volume, 2) vol FROM volume WHERE id_gedung = '1' ORDER BY id DESC LIMIT 1
10:57:34.152 -> Got message from : 76
10:57:34.152 -> Asal gedung =P
10:57:34.152 -> Rate gedung = 3.01
10:57:34.152 -> Data notifikasi = 0.00
10:57:34.152 -> Data pegendalian= 1.00
10:57:34.152 -> Rate gedung P= 3.01
10:57:34.152 -> Pengendalian gedung P = 1
10:57:34.152 -> http://pantaukendalialair.com/includes/masuk_data.php?rP=3.01
10:57:37.662 -> 200
10:57:37.662 -> SELECT ROUND(volume, 2) vol FROM volume WHERE id_gedung = '5' ORDER BY id DESC LIMIT 1
10:58:04.946 -> Got message from : 76
10:58:04.946 -> Asal gedung =C
10:58:04.946 -> Rate gedung = 0.38
10:58:04.946 -> Data notifikasi = 0.00
10:58:04.946 -> Data pegendalian= 0.00
10:58:04.946 -> Rate gedung C= 0.38
10:58:04.946 -> Notif gedung C = 0
10:58:04.946 -> Pengendalian gedung C = 0

```

Autoscroll  Show timestamp

Gambar 4.7 Tampilan serial monitor pengiriman keakuratan data sensor *flowmeter*

Sumber : Dokumentasi Penulis

Pada tabel 4.5 merupakan tabel hasil pengujian keakuratan data sensor *flowmeter* dan pada gambar 4.7 Menunjukan tampilan serial monitor pada *gateway* berupa data monitoring dari sel gedung pusat,

gedung A, gedung C, dan gedung D. Hasil keakuratan antara gedung pusat dan gedung A,B,C,D adalah gedung pusat lebih kecil daripada gedung A+ gedung B+ gedung C+ gedung D, dan gedung pusat lebih besar daripada gedung A+ gedung B+ gedung C+ gedung D. Hal tersebut dapat terjadi dikarenakan perangkat *flowmeter* yang digunakan kemungkinan berbeda *type/merk* yang menyebabkan terjadinya perbedaan akuransi debit air. Dan pada sel pusat menggunakan *flowmeter* dengan ukuran 3/4 inchi yang menyebabkan perbedaan tekanan air pada *flowmeter* dengan ukuran 1/2 inchi. Perbedaan tekanan juga mempengaruhi keakuratan perhitungan debit air. Selain itu, juga dipengaruhi dalam pengambilan data pada setiap gedungnya dan juga dikarenakan adanya data yang terlewat dan tidak sampai ke *gateway* sehingga dapat mempengaruhi keakuratan perhitungan jumlah debit air antara sel gedung pusat dengan sel gedung ABCD. Data yang terlewat dapat dikarenakan dalam proses pengiriman data terlalu banyak data yang dikirim sehingga menyebabkan data terlewat atau tidak diterima pada *gateway*.

## **BAB V**

### **PENUTUP**

#### **5.1 Kesimpulan**

Berdasarkan hasil pengujian dan analisa Tugas Akhir Rancang Bangun Model Monitoring dan Pengendalian Air Dalam Sistem *Smart Building* Berbasis IoT yang telah dilakukan, dapat diambil kesimpulan bahwa:

1. Data perintah yang diterima LoRa *Shield* pada sel gedung dari koordinator merupakan 2 bit data, bit pertama terdiri angka 1 atau 2. Angka 1 digunakan untuk menghidupkan solenoid valve (katub terbuka, air mengalir), dan angka 2 digunakan untuk mematikan solenoid valve (katub tertutup, air tidak mengalir). Selanjutnya bit kedua adalah huruf kapital yaitu huruf A untuk sel gedung A, huruf B untuk sel gedung B, huruf C untuk sel gedung C, huruf D untuk sel gedung D, huruf P untuk sel gedung P. huruf tersebut berfungsi sebagai tujuan alamat sel gedung agar data dapat dieksekusi.
2. Sel gedung akan mengirimkan data *flowmeter* yang digunakan untuk memonitoring penggunaan air. Data tersebut dikirimkan ke *node* koordinator, dan *gateway*. Data *flowmeter* yang dikirim adalah debit yang mengalir di setiap gedungnya, debit tersebut menggunakan satuan L/Menit.
3. *Node* koordinator dapat meneruskan data pengendalian dari *gateway* yang akan dikirimkan ke sel gedung untuk menghidupkan dan mematikan solenoid valve.
4. *Node* koordinator dapat meneruskan data *monitoring* dari sel gedung yang akan dikirimkan ke *gateway* yang digunakan sebagai data monitoring yang akan diolah di *Web Server*.
5. *Gateway* berhasil mengambil data pengendalian dari *database Web Server* kemudian data tersebut dikirim ke *node* koordinator, dimana di *node koordinator* data tersebut dikonversi menjadi data huruf yang digunakan sebagai identitas di setiap sel gedung.

6. *Gateway* dapat menerima data *monitoring* dari *node* koordinator untuk dikirimkan ke *database Web Server* untuk diolah menjadi grafik penggunaan air.
7. Jarak komunikasi data menggunakan LoRa *shield* dengan LoRa Bee antara sel gedung dengan *node* koordinator dapat mencapai 101,65 meter, dan jarak *node* koordinator dengan *gateway* dapat mencapai 275,24 meter dilokasi yang tanpa *obstacle*.

## 5.2 Saran

Rancang Bangun Model Monitoring dan Pengendalian Air Dalam Sistem *Smart Building* Berbasis IoT masih bisa dikembangkan lagi sehingga menjadi lebih baik. Adapun hal-hal yang dapat dikembangkan dari sistem ini, antara lain:

1. Sistem diharapkan dapat dikembangkan supaya menjangkau wilayah yang lebih luas.
2. Dapat dilakukan pendalam program agar data yang digunakan untuk pengendalian dan *monitoring* dapat dikirim dan diterima lebih maksimal lagi.

## DAFTAR PUSTAKA

- Putri, Adela dan Reza, Firma. 2019. Rancang Bangun Lampu Penerangan Taman Dikontrol Oleh *Lora Shield* dan Arduino Melalui Sistem *Wireless* dan Internet. Tugas Akhir. Semarang: Jurusan Teknik Elektro Politeknik Negeri Semarang
- uprianto, “Pengertian dan Prinsip Kerja *Solenoid Valve*”, 29 Oktober 2015 [Online]. Tersedia: “<http://blog.unnes.ac.id/antosupri/pengertian-dan-prinsip-kerja-solenoid-valve>” [Diakses pada tanggal 19 Agustus 2020]
- Dragino. 2017. LoRa Bee— *Wiki for Dragino Project.* [https://wiki.dragino.com/index.php?title=Lora\\_BEE](https://wiki.dragino.com/index.php?title=Lora_BEE) [Diakses pada tanggal 20 Agustus 2020]
- Nur Handayani, Dina dan Nanda Lisaryanti. 2019. Sistem Koordinator Pada Pengontrol Lampu Melalui Sistem *Wireless* dan Internet. Tugas Akhir. Semarang: Jurusan Teknik Elektro Politeknik Negeri Semarang
- ... ”Node MCU ESP8266”, [https://eprints.akakom.ac.id/4904/3/\\_143310003\\_BAB\\_II.pdf](https://eprints.akakom.ac.id/4904/3/_143310003_BAB_II.pdf) [Diakses pada tanggal 20 Agustus 2020]
- Setyawan, Niko. 2016. “Rancag Bangun Alat Ukur Volume Fluida Otomatis Menggunakan Flow Meter Berbasis Arduino Mega”. Yogyakarta: UIN Sunan Kalijaga
- Prasetyo, Galih. 2020. “Sistem *Monitorinug* dan Kendali Penggunaan Air Untuk Menyiram Tanaman Berbasis *IoT* Pada *Smart Agriculture System*. Tugas Akhir. Semarang: Jurusan Teknik Elektro Politeknik Negeri Semarang.
- Arduino, 2017. Arduino Uno Mikrokontroler ATMega 328. <http://www.labelektronika.com/2017/02/arduino-uno-mikrokontroler-atmega-328.html> [ Diakses pada tanggal 19 Agustus 2020]

- Dragino, 2017. LoRa Shield – *Wiki for Dragino Project.*  
[https://wiki.dragino.com/index.php?title=Lora\\_Shield](https://wiki.dragino.com/index.php?title=Lora_Shield) [Diakses pada tanggal 20Agustus 2020]
- Razor, Aldi, 2021. Cara kerja Arduino Uno dan bagaimana prinsip serta peranannya.  
<https://www.aldyrazor.com/2020/07/cara-kerja-arduino.html?m=1> [Diakses pada tanggal 20April 2021]



# **LAMPIRAN**

## Lampiran 1 Program Sel Gedung

Program pada sel Gedung P

```
#include <SPI.h>
#include <RH_RF95.h>
#include <RHReliableDatagram.h>
#include <String.h>
#include <SimpleTimer.h>

const int relay = 4;
int count = 0;
SimpleTimer firstTimer(5000);
SimpleTimer secondTimer(0);

#define KOORDINATORA 'K'
#define GEDUNG      'P'
#define TXPWR       13
float frequency = 433.0;
#define ReceivedBufferLength 20
char receivedBuffer[ReceivedBufferLength+1]; // store the serial command
byte receivedBufferIndex = 0;
boolean enterCalibrationFlag = 0;
#define compensationFactorAddress 8
float compensationFactor;
RH_RF95 rf95;

#pragma pack(push, 2)
typedef struct DATA {
    int data1, data2;
    float rate;
    int notif;
    int datapengendalian;  };

```

```

#pragma pack(pop)

DATA data;

const float FREQ = 433.0;

RH Datagram kirim(rf95, GEDUNG);
RH Datagram terima(rf95, GEDUNG);

// flowmeter

volatile int flow_frequency; // Measures flow sensor pulses
// Calculated litres/hour

float vol = 0.0,l_minute;

unsigned char flowsensor = 3; // Sensor Input
unsigned long currentTime;
unsigned long cloopTime;

void flow () // Interrupt function
{
    flow_frequency++;
}

void setup ()
{
    pinMode(flowsensor, INPUT);
    digitalWrite(flowsensor, HIGH); // Optional Internal Pull-Up
    Serial.begin(9600);
    attachInterrupt(digitalPinToInterrupt(flowsensor),  flow,  RISING); // Setup
    Interrupt
    currentTime = millis();
    cloopTime = currentTime;
    pinMode(relay, OUTPUT);
    digitalWrite(relay, HIGH);
    SetupLora();
}

```

```

void SetupLora(void)
{
{
    if (kirim.init())
    {
        if (!rf95.setFrequency(FREQ))
            Serial.println("Unable to set RF95 frequency");
        //if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
        //Serial.println("Invalid setModemConfig() option");
        rf95.setTxPower(TXPWR);
        Serial.println("RF radio initialized.");
    }
    else
        Serial.println("RF radio initialization failed.");
        Serial.print("RF95 max message length = ");
        Serial.println(rf95.maxMessageLength());
    }
}

{
    if (terima.init())
    {
        if (!rf95.setFrequency(FREQ))
            Serial.println("Unable to set RF95 frequency");
        //if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
        //Serial.println("Invalid setModemConfig() option");
        rf95.setTxPower(TXPWR);
        Serial.println("RF95 radio initialized.");
    }
    else
        Serial.println("RF95 radio initialization failed.");
        Serial.print("RF95 max message length = ");
        Serial.println(rf95.maxMessageLength());  }
}

```

```
}
```

```
void (*reset) (void) = 0;
void bacaSensor()
{
    if(secondTimer.isReady()){
        currentTime = millis(); // Every second, calculate and print litres/hour
        if(currentTime >= (cloopTime + 60000))
        {
            cloopTime = currentTime; // Updates cloopTime
            if(flow_frequency != 0)
            {
                // Pulse frequency (Hz) = 5.5Q, Q is flow rate in L/min.
                l_minute = (flow_frequency / 3.5); // (Pulse frequency x 60 min) / 7.5Q =
                // flowrate in L/hour
                l_minute = l_minute/60;
                //vol = vol +l_minute;
                flow_frequency = 0; // Reset Counter
                count++;
                data.rate = l_minute;
                if(count == 10){
                    data.notif = 1;
                }
                else if (count < 10){
                    data.notif = 0;
                }
                Serial.println(count);
                Serial.println("Sending Rate = " + String(data.rate));
                Serial.println("Sending Notif = "+String(data.notif));
                Serial.println("Sending Data Pengendalian = " +
                String(data.datapengendalian));
                if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))

```

```

Serial.print("Transmit failed");
rf95.waitPacketSent(100);
}

else if(flow_frequency == 0){
count = 0;
data.rate = 0;
data.notif = 0;
Serial.println("Sending Rate = " + String(data.rate));
Serial.println("Sending Notif = "+String(data.notif));
Serial.println("Sending      Data      Pengendalian      =      "      +
String(data.datapengendalian));
if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
Serial.print("Transmit failed");
rf95.waitPacketSent(100);
}

if(count == 10){
count = 0;
}
secondTimer.reset();
}

}

}

void loop(void)
{
bacaSensor();

if(firstTimer.isReady()){
uint8_t bufLen = sizeof(data);
char from;
if (terima.recvfrom((uint8_t *) &data, &bufLen, &from))
{

```

```

Serial.print("Got mesaage from : ");
Serial.println(from);
if (bufLen == sizeof(data) && from == 'K')
{
    Serial.println("Received alert1 = " + String(data.data1));
    int logic = data.data1;
    if (logic < 2)
    {
        digitalWrite(relay, HIGH);
    }
    else if ( logic > 1 )
    {
        digitalWrite(relay, LOW);
    }
    if(logic > 1){
        data.datapengendalian = 2;
    }
    else if(logic < 2){
        data.datapengendalian = 1;
    }
}
if(data.data1 > 1){
    Serial.println("Sending Rate = " + String(data.rate));
    Serial.println("Sending Notif = "+String(data.notif));
    Serial.println("Sending Data Pengendalian = " + String(data.datapengendalian));
    if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
        Serial.print("Transmit failed");
    rf95.waitPacketSent(100); } }
firstTimer.reset();
}
}

```

## Program pada sel Gedung A

```
#include <SPI.h>
#include <RH_RF95.h>
#include <RHReliableDatagram.h>
#include <String.h>
#include <SimpleTimer.h>

const int relay = 4;
int count = 0;
SimpleTimer firstTimer(5000);
SimpleTimer secondTimer(0);

#define KOORDINATORA 'K'
#define GEDUNG    'A'
#define TXPWR     13
float frequency = 433.0;
#define ReceivedBufferLength 20
char receivedBuffer[ReceivedBufferLength+1]; // store the serial command
byte receivedBufferIndex = 0;
boolean enterCalibrationFlag = 0;
#define compensationFactorAddress 8
float compensationFactor;
RH_RF95 rf95;

#pragma pack(push, 2)
typedef struct DATA {
int data1, data2;
float rate;
int notif;
int datapengendalian;  };
#pragma pack(pop)
```

```

DATA data;
const float FREQ = 433.0;
RH Datagram kirim(rf95, GEDUNG);
RH Datagram terima(rf95, GEDUNG);

// flowmeter
volatile int flow_frequency; // Measures flow sensor pulses
// Calculated litres/hour
float vol = 0.0,l_minute;
unsigned char flowsensor = 3; // Sensor Input
unsigned long currentTime;
unsigned long cloopTime;

void flow () // Interrupt function
{
    flow_frequency++;
}

void setup ()
{
    pinMode(flowsensor, INPUT);
    digitalWrite(flowsensor, HIGH); // Optional Internal Pull-Up
    Serial.begin(9600);
    attachInterrupt(digitalPinToInterrupt(flowsensor),  flow,  RISING); // Setup
    Interrupt
    currentTime = millis();
    cloopTime = currentTime;
    pinMode(relay, OUTPUT);
    digitalWrite(relay, HIGH);
    SetupLora();
}

```

```

void SetupLora(void)
{
{
    if (kirim.init())
    {
        if (!rf95.setFrequency(FREQ))
            Serial.println("Unable to set RF95 frequency");
        //if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
        //Serial.println("Invalid setModemConfig() option");
        rf95.setTxPower(TXPWR);
        Serial.println("RF radio initialized.");
    }
    else
        Serial.println("RF radio initialization failed.");
        Serial.print("RF95 max message length = ");
        Serial.println(rf95.maxMessageLength());
    }
}

{
    if (terima.init())
    {
        if (!rf95.setFrequency(FREQ))
            Serial.println("Unable to set RF95 frequency");
        //if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
        //Serial.println("Invalid setModemConfig() option");
        rf95.setTxPower(TXPWR);
        Serial.println("RF95 radio initialized.");
    }
    else
        Serial.println("RF95 radio initialization failed.");
        Serial.print("RF95 max message length = ");
        Serial.println(rf95.maxMessageLength());  }
}

```

```

void (*reset) (void) = 0;
void bacaSensor()
{
    if(secondTimer.isReady()){
        currentTime = millis(); // Every second, calculate and print litres/hour
        if(currentTime >= (cloopTime + 60000))
        {
            cloopTime = currentTime; // Updates cloopTime
            if(flow_frequency != 0)
            {
                // Pulse frequency (Hz) = 5.5Q, Q is flow rate in L/min.
                l_minute = (flow_frequency / 3.5); // (Pulse frequency x 60 min) / 7.5Q =
                flowrate in L/hour
                l_minute = l_minute/60;
                //vol = vol +l_minute;
                flow_frequency = 0; // Reset Counter
                count++;
                data.rate = l_minute;
                if(count == 10){
                    data.notif = 1;
                }
                else if (count < 10){
                    data.notif = 0;
                }
                Serial.println(count);
                Serial.println("Sending Rate = " + String(data.rate));
                Serial.println("Sending Notif = "+String(data.notif));
                Serial.println("Sending Data Pengendalian = " +
                String(data.datapengendalian));
                if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
                    Serial.print("Transmit failed");
            }
        }
    }
}

```

```

rf95.waitPacketSent(100);
}

else if(flow_frequency == 0){
count = 0;
data.rate = 0;
data.notif = 0;
Serial.println("Sending Rate = " + String(data.rate));
Serial.println("Sending Notif = "+String(data.notif));
Serial.println("Sending Data Pengendalian = " +
String(data.datapengendalian));
if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
Serial.print("Transmit failed");
rf95.waitPacketSent(100);
}

if(count == 10){
count = 0;
}
secondTimer.reset();
}
}

void loop(void)
{
bacaSensor();

if(firstTimer.isReady()){
uint8_t bufLen = sizeof(data);
char from;
if (terima.recvfrom((uint8_t *) &data, &bufLen, &from))
{
Serial.print("Got message from : ");

```

```

Serial.println(from);
if (bufLen == sizeof(data) && from == 'K')
{
    Serial.println("Received alert1 = " + String(data.data1));
    int logic = data.data1;
    if (logic < 2)
    {
        digitalWrite(relay, HIGH);
    }
    else if ( logic > 1 )
    {
        digitalWrite(relay, LOW);
    }
    if(logic > 1){
        data.datapengendalian = 2;
    }
    else if(logic < 2){
        data.datapengendalian = 1;
    }
}
if(data.data1 > 1){
    Serial.println("Sending Rate = " + String(data.rate));
    Serial.println("Sending Notif = "+String(data.notif));
    Serial.println("Sending      Data      Pengendalian      =      "      +
String(data.datapengendalian));
    if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
        Serial.print("Transmit failed");
    rf95.waitPacketSent(100); } }
firstTimer.reset();
}
}

```

## Program pada sel Gedung B

```
#include <SPI.h>
#include <RH_RF95.h>
#include <RHReliableDatagram.h>
#include <String.h>
#include <SimpleTimer.h>

const int relay = 4;
int count = 0;
SimpleTimer firstTimer(5000);
SimpleTimer secondTimer(0);

#define KOORDINATORA 'K'
#define GEDUNG    'B'
#define TXPWR     13
float frequency = 433.0;
#define ReceivedBufferLength 20
char receivedBuffer[ReceivedBufferLength+1]; // store the serial command
byte receivedBufferIndex = 0;
boolean enterCalibrationFlag = 0;
#define compensationFactorAddress 8
float compensationFactor;
RH_RF95 rf95;

#pragma pack(push, 2)
typedef struct DATA {
int data1, data2;
float rate;
int notif;
int datapengendalian;  };
#pragma pack(pop)
```

```

DATA data;
const float FREQ = 433.0;
RH Datagram kirim(rf95, GEDUNG);
RH Datagram terima(rf95, GEDUNG);

// flowmeter
volatile int flow_frequency; // Measures flow sensor pulses
// Calculated litres/hour
float vol = 0.0,l_minute;
unsigned char flowsensor = 3; // Sensor Input
unsigned long currentTime;
unsigned long cloopTime;

void flow () // Interrupt function
{
    flow_frequency++;
}

void setup ()
{
    pinMode(flowsensor, INPUT);
    digitalWrite(flowsensor, HIGH); // Optional Internal Pull-Up
    Serial.begin(9600);
    attachInterrupt(digitalPinToInterrupt(flowsensor),  flow,  RISING); // Setup
    Interrupt
    currentTime = millis();
    cloopTime = currentTime;
    pinMode(relay, OUTPUT);
    digitalWrite(relay, HIGH);
    SetupLora();
}

```

```

void SetupLora(void)
{
{
    if (kirim.init())
    {
        if (!rf95.setFrequency(FREQ))
            Serial.println("Unable to set RF95 frequency");
        //if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
        //Serial.println("Invalid setModemConfig() option");
        rf95.setTxPower(TXPWR);
        Serial.println("RF radio initialized.");
    }
    else
        Serial.println("RF radio initialization failed.");
        Serial.print("RF95 max message length = ");
        Serial.println(rf95.maxMessageLength());
    }
}

{
    if (terima.init())
    {
        if (!rf95.setFrequency(FREQ))
            Serial.println("Unable to set RF95 frequency");
        //if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
        //Serial.println("Invalid setModemConfig() option");
        rf95.setTxPower(TXPWR);
        Serial.println("RF95 radio initialized.");
    }
    else
        Serial.println("RF95 radio initialization failed.");
        Serial.print("RF95 max message length = ");
        Serial.println(rf95.maxMessageLength());  }
}

```

```

void (*reset) (void) = 0;
void bacaSensor()
{
    if(secondTimer.isReady()){
        currentTime = millis(); // Every second, calculate and print litres/hour
        if(currentTime >= (cloopTime + 60000))
        {
            cloopTime = currentTime; // Updates cloopTime
            if(flow_frequency != 0)
            {
                // Pulse frequency (Hz) = 5.5Q, Q is flow rate in L/min.
                l_minute = (flow_frequency / 3.5); // (Pulse frequency x 60 min) / 7.5Q =
                flowrate in L/hour
                l_minute = l_minute/60;
                //vol = vol +l_minute;
                flow_frequency = 0; // Reset Counter
                count++;
                data.rate = l_minute;
                if(count == 10){
                    data.notif = 1;
                }
                else if (count < 10){
                    data.notif = 0;
                }
                Serial.println(count);
                Serial.println("Sending Rate = " + String(data.rate));
                Serial.println("Sending Notif = "+String(data.notif));
                Serial.println("Sending Data Pengendalian = " +
                String(data.datapengendalian));
                if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
                    Serial.print("Transmit failed");
            }
        }
    }
}

```

```

rf95.waitPacketSent(100);
}

else if(flow_frequency == 0){
count = 0;
data.rate = 0;
data.notif = 0;
Serial.println("Sending Rate = " + String(data.rate));
Serial.println("Sending Notif = "+String(data.notif));
Serial.println("Sending Data Pengendalian = " +
String(data.datapengendalian));
if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
Serial.print("Transmit failed");
rf95.waitPacketSent(100);
}

if(count == 10){
count = 0;
}
secondTimer.reset();
}
}

void loop(void)
{
bacaSensor();

if(firstTimer.isReady()){
uint8_t bufLen = sizeof(data);
char from;
if (terima.recvfrom((uint8_t *) &data, &bufLen, &from))
{
Serial.print("Got mesaage from : ");

```

```

Serial.println(from);
if (bufLen == sizeof(data) && from == 'K')
{
    Serial.println("Received alert1 = " + String(data.data1));
    int logic = data.data1;
    if (logic < 2)
    {
        digitalWrite(relay, HIGH);
    }
    else if ( logic > 1 )
    {
        digitalWrite(relay, LOW);
    }
    if(logic > 1){
        data.datapengendalian = 2;
    }
    else if(logic < 2){
        data.datapengendalian = 1;
    }
}
if(data.data1 > 1){
    Serial.println("Sending Rate = " + String(data.rate));
    Serial.println("Sending Notif = "+String(data.notif));
    Serial.println("Sending      Data      Pengendalian      =      "      +
String(data.datapengendalian));
    if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
        Serial.print("Transmit failed");
    rf95.waitPacketSent(100); } }
firstTimer.reset();
}
}

```

## Program pada sel Gedung C

```
#include <SPI.h>
#include <RH_RF95.h>
#include <RHReliableDatagram.h>
#include <String.h>
#include <SimpleTimer.h>

const int relay = 4;
int count = 0;
SimpleTimer firstTimer(5000);
SimpleTimer secondTimer(0);

#define KOORDINATORA 'K'
#define GEDUNG    'C'
#define TXPWR     13
float frequency = 433.0;
#define ReceivedBufferLength 20
char receivedBuffer[ReceivedBufferLength+1]; // store the serial command
byte receivedBufferIndex = 0;
boolean enterCalibrationFlag = 0;
#define compensationFactorAddress 8
float compensationFactor;
RH_RF95 rf95;

#pragma pack(push, 2)
typedef struct DATA {
int data1, data2;
float rate;
int notif;
int datapengendalian;  };
#pragma pack(pop)
```

```

DATA data;
const float FREQ = 433.0;
RH Datagram kirim(rf95, GEDUNG);
RH Datagram terima(rf95, GEDUNG);

// flowmeter
volatile int flow_frequency; // Measures flow sensor pulses
// Calculated litres/hour
float vol = 0.0,l_minute;
unsigned char flowsensor = 3; // Sensor Input
unsigned long currentTime;
unsigned long cloopTime;

void flow () // Interrupt function
{
    flow_frequency++;
}

void setup ()
{
    pinMode(flowsensor, INPUT);
    digitalWrite(flowsensor, HIGH); // Optional Internal Pull-Up
    Serial.begin(9600);
    attachInterrupt(digitalPinToInterrupt(flowsensor),  flow,  RISING); // Setup
    Interrupt
    currentTime = millis();
    cloopTime = currentTime;
    pinMode(relay, OUTPUT);
    digitalWrite(relay, HIGH);
    SetupLora();
}

```

```

void SetupLora(void)
{
{
    if (kirim.init())
    {
        if (!rf95.setFrequency(FREQ))
            Serial.println("Unable to set RF95 frequency");
        //if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
        //Serial.println("Invalid setModemConfig() option");
        rf95.setTxPower(TXPWR);
        Serial.println("RF radio initialized.");
    }
    else
        Serial.println("RF radio initialization failed.");
        Serial.print("RF95 max message length = ");
        Serial.println(rf95.maxMessageLength());
    }
}

{
    if (terima.init())
    {
        if (!rf95.setFrequency(FREQ))
            Serial.println("Unable to set RF95 frequency");
        //if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
        //Serial.println("Invalid setModemConfig() option");
        rf95.setTxPower(TXPWR);
        Serial.println("RF95 radio initialized.");
    }
    else
        Serial.println("RF95 radio initialization failed.");
        Serial.print("RF95 max message length = ");
        Serial.println(rf95.maxMessageLength());  }
}

```

```

void (*reset) (void) = 0;
void bacaSensor()
{
    if(secondTimer.isReady()){
        currentTime = millis(); // Every second, calculate and print litres/hour
        if(currentTime >= (cloopTime + 60000))
        {
            cloopTime = currentTime; // Updates cloopTime
            if(flow_frequency != 0)
            {
                // Pulse frequency (Hz) = 5.5Q, Q is flow rate in L/min.
                l_minute = (flow_frequency / 3.5); // (Pulse frequency x 60 min) / 7.5Q =
                flowrate in L/hour
                l_minute = l_minute/60;
                //vol = vol +l_minute;
                flow_frequency = 0; // Reset Counter
                count++;
                data.rate = l_minute;
                if(count == 10){
                    data.notif = 1;
                }
                else if (count < 10){
                    data.notif = 0;
                }
                Serial.println(count);
                Serial.println("Sending Rate = " + String(data.rate));
                Serial.println("Sending Notif = "+String(data.notif));
                Serial.println("Sending Data Pengendalian = " +
                String(data.datapengendalian));
                if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
                    Serial.print("Transmit failed");
            }
        }
    }
}

```

```

rf95.waitPacketSent(100);
}

else if(flow_frequency == 0){
count = 0;
data.rate = 0;
data.notif = 0;
Serial.println("Sending Rate = " + String(data.rate));
Serial.println("Sending Notif = "+String(data.notif));
Serial.println("Sending Data Pengendalian = " +
String(data.datapengendalian));
if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
Serial.print("Transmit failed");
rf95.waitPacketSent(100);
}

if(count == 10){
count = 0;
}
secondTimer.reset();
}
}

void loop(void)
{
bacaSensor();

if(firstTimer.isReady()){
uint8_t bufLen = sizeof(data);
char from;
if (terima.recvfrom((uint8_t *) &data, &bufLen, &from))
{
Serial.print("Got message from : ");

```

```

Serial.println(from);
if (bufLen == sizeof(data) && from == 'K')
{
    Serial.println("Received alert1 = " + String(data.data1));
    int logic = data.data1;
    if (logic < 2)
    {
        digitalWrite(relay, HIGH);
    }
    else if ( logic > 1 )
    {
        digitalWrite(relay, LOW);
    }
    if(logic > 1){
        data.datapengendalian = 2;
    }
    else if(logic < 2){
        data.datapengendalian = 1;
    }
}
if(data.data1 > 1){
    Serial.println("Sending Rate = " + String(data.rate));
    Serial.println("Sending Notif = "+String(data.notif));
    Serial.println("Sending      Data      Pengendalian      =      "      +
String(data.datapengendalian));
    if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
        Serial.print("Transmit failed");
    rf95.waitPacketSent(100); } }
firstTimer.reset();
}
}

```

## Program pada sel Gedung D

```
#include <SPI.h>
#include <RH_RF95.h>
#include <RHReliableDatagram.h>
#include <String.h>
#include <SimpleTimer.h>

const int relay = 4;
int count = 0;
SimpleTimer firstTimer(5000);
SimpleTimer secondTimer(0);

#define KOORDINATORA 'K'
#define GEDUNG    'D'
#define TXPWR     13
float frequency = 433.0;
#define ReceivedBufferLength 20
char receivedBuffer[ReceivedBufferLength+1]; // store the serial command
byte receivedBufferIndex = 0;
boolean enterCalibrationFlag = 0;
#define compensationFactorAddress 8
float compensationFactor;
RH_RF95 rf95;

#pragma pack(push, 2)
typedef struct DATA {
int data1, data2;
float rate;
int notif;
int datapengendalian;  };
#pragma pack(pop)
```

```

DATA data;
const float FREQ = 433.0;
RH Datagram kirim(rf95, GEDUNG);
RH Datagram terima(rf95, GEDUNG);

// flowmeter
volatile int flow_frequency; // Measures flow sensor pulses
// Calculated litres/hour
float vol = 0.0,l_minute;
unsigned char flowsensor = 3; // Sensor Input
unsigned long currentTime;
unsigned long cloopTime;

void flow () // Interrupt function
{
    flow_frequency++;
}

void setup ()
{
    pinMode(flowsensor, INPUT);
    digitalWrite(flowsensor, HIGH); // Optional Internal Pull-Up
    Serial.begin(9600);
    attachInterrupt(digitalPinToInterrupt(flowsensor),  flow,  RISING); // Setup
    Interrupt
    currentTime = millis();
    cloopTime = currentTime;
    pinMode(relay, OUTPUT);
    digitalWrite(relay, HIGH);
    SetupLora();
}

```

```

void SetupLora(void)
{
{
    if (kirim.init())
    {
        if (!rf95.setFrequency(FREQ))
            Serial.println("Unable to set RF95 frequency");
        //if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
        //Serial.println("Invalid setModemConfig() option");
        rf95.setTxPower(TXPWR);
        Serial.println("RF radio initialized.");
    }
    else
        Serial.println("RF radio initialization failed.");
        Serial.print("RF95 max message length = ");
        Serial.println(rf95.maxMessageLength());
    }
}

{
    if (terima.init())
    {
        if (!rf95.setFrequency(FREQ))
            Serial.println("Unable to set RF95 frequency");
        //if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
        //Serial.println("Invalid setModemConfig() option");
        rf95.setTxPower(TXPWR);
        Serial.println("RF95 radio initialized.");
    }
    else
        Serial.println("RF95 radio initialization failed.");
        Serial.print("RF95 max message length = ");
        Serial.println(rf95.maxMessageLength());  }
}

```

```

void (*reset) (void) = 0;
void bacaSensor()
{
    if(secondTimer.isReady()){
        currentTime = millis(); // Every second, calculate and print litres/hour
        if(currentTime >= (cloopTime + 60000))
        {
            cloopTime = currentTime; // Updates cloopTime
            if(flow_frequency != 0)
            {
                // Pulse frequency (Hz) = 5.5Q, Q is flow rate in L/min.
                l_minute = (flow_frequency / 3.5); // (Pulse frequency x 60 min) / 7.5Q =
                flowrate in L/hour
                l_minute = l_minute/60;
                //vol = vol +l_minute;
                flow_frequency = 0; // Reset Counter
                count++;
                data.rate = l_minute;
                if(count == 10){
                    data.notif = 1;
                }
                else if (count < 10){
                    data.notif = 0;
                }
                Serial.println(count);
                Serial.println("Sending Rate = " + String(data.rate));
                Serial.println("Sending Notif = "+String(data.notif));
                Serial.println("Sending Data Pengendalian = " +
String(data.datapengendalian));
                if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
                    Serial.print("Transmit failed");
            }
        }
    }
}

```

```

rf95.waitPacketSent(100);
}

else if(flow_frequency == 0){
count = 0;
data.rate = 0;
data.notif = 0;
Serial.println("Sending Rate = " + String(data.rate));
Serial.println("Sending Notif = "+String(data.notif));
Serial.println("Sending Data Pengendalian = " +
String(data.datapengendalian));
if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
Serial.print("Transmit failed");
rf95.waitPacketSent(100);
}

if(count == 10){
count = 0;
}
secondTimer.reset();
}
}

void loop(void)
{
bacaSensor();

if(firstTimer.isReady()){
uint8_t bufLen = sizeof(data);
char from;
if (terima.recvfrom((uint8_t *) &data, &bufLen, &from))
{
Serial.print("Got message from : ");

```

```

Serial.println(from);
if (bufLen == sizeof(data) && from == 'K')
{
    Serial.println("Received alert1 = " + String(data.data1));
    int logic = data.data1;
    if (logic < 2)
    {
        digitalWrite(relay, HIGH);
    }
    else if ( logic > 1 )
    {
        digitalWrite(relay, LOW);
    }
    if(logic > 1){
        data.datapengendalian = 2;
    }
    else if(logic < 2){
        data.datapengendalian = 1;
    }
}
if(data.data1 > 1){
    Serial.println("Sending Rate = " + String(data.rate));
    Serial.println("Sending Notif = "+String(data.notif));
    Serial.println("Sending      Data      Pengendalian      =      "      +
String(data.datapengendalian));
    if (!kirim.sendto((uint8_t *) &data, sizeof(data), KOORDINATORA))
        Serial.print("Transmit failed");
    rf95.waitPacketSent(100); } }
firstTimer.reset();
}
}

```

## Lampiran 2 Program Node Koordinator

Program pada Koordinator 433Mhz

```
#include <SPI.h>
#include <RH_RF95.h>
#include <RHReliableDatagram.h>
#include <String.h>
#include <SimpleTimer.h>

SimpleTimer firstTimer(500);
SimpleTimer secondTimer(0);
//SimpleTimer threeTimer (100);

char perintah[5];
char gedungtujuan[5];

#define KOORDINATORA 'K'
#define TXPWR    13
#define ReceivedBufferLength 20
char receivedBuffer[ReceivedBufferLength+1]; // store the serial command
byte receivedBufferIndex = 0;
boolean enterCalibrationFlag = 0;
#define compensationFactorAddress 8
float compensationFactor;
RH_RF95 rf95;
#pragma pack(push, 2)
typedef struct DATA {
    int data1, data2;
    float rate;
    int notif;
    int datapengendalian;    };
#pragma pack(pop)
```

```

DATA data;

const float FREQ = 433.0;

RH Datagram kirim(rf95,KOORDINATORA );
RH Datagram terima(rf95,KOORDINATORA); // as we want

read

void setup()
{
    Serial.begin(9600);
    SetupLora();
}

void SetupLora(void)
{
{
    if (kirim.init())
    {
        if (!rf95.setFrequency(FREQ))
            Serial.println("Unable to set RF95 frequency");
        //if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
        //Serial.println("Invalid setModemConfig() option");
        rf95.setTxPower(TXPWR);
        Serial.println("RF radio initialized.");
    }
    else
        Serial.println("RF radio initialization failed.");
    Serial.print("RF95 max message length = ");
    Serial.println(rf95.maxMessageLength());
}
{
    if (terima.init())
    {
        if (!rf95.setFrequency(FREQ))
            Serial.println("Unable to set RF95 frequency");
}

```

```

//if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
//Serial.println("Invalid setModemConfig() option");
rf95.setTxPower(TXPWR);
Serial.println("RF95 radio initialized.");
}

else
Serial.println("RF95 radio initialization failed.");
Serial.print("RF95 max message length = ");
Serial.println(rf95.maxMessageLength());
}

}

void loop()
{
if(firstTimer.isReady()){
if (Serial.available()){

Serial.readBytes(perintah,5);//Read the serial data and store in var
Serial.readBytes(gedungtujuan,5);//Read the serial data and store in var
data.data1 = atoi(perintah);

#define GEDUNG gedungtujuan[0]
//Serial.print("kirim ke ");
// Serial.println(GEDUNG);
//Serial.println (data.data1);
//Serial.println("kirim perintah 1/0 = " + String(data.data1));
if (!kirim.sendto((uint8_t *) &data, sizeof(data), GEDUNG))
Serial.print("Transmit failed");

rf95.waitPacketSent(100);
}
}

if(secondTimer.isReady())
{
if (rf95.waitAvailableTimeout(1000)) // wait 100 mSec max for response
{

```

```

    uint8_t bufLen = sizeof(data);
    char from;
    if (terima.recvfrom((uint8_t *) &data, &bufLen, &from))
    {
        //Serial.print("Got message from : ");
        //Serial.println(from);
        //Serial.println("===== Received Rate = " + String(data.rate));
        //Serial.println("===== Received Volume = " + String(data.debit));
        String datagedung,rategedung,debitgedung;
        char rate[5];
        char gdg[5];
        String datanotif;
        char nilainotif[5];
        char pengendalian[5];
        String nilaipengendalian;
        datagedung = (from);
        rategedung = float (data.rate);
        datanotif = data.notif;
        nilaipengendalian =int (data.datapengendalian);
        datagedung.toCharArray(gdg,5);
        rategedung.toCharArray(rate,5);
        datanotif.toCharArray(nilainotif,5);
        nilaipengendalian.toCharArray(pengendalian,5);
        Serial.write(gdg,5);
        Serial.write(rate,5);
        Serial.write(nilainotif,5);
        Serial.write(pengendalian,5);    }
    secondTimer.reset();
}
}
}

```

## Program pada Koordinator 915Mhz

```
#include <SPI.h>
#include <RH_RF95.h>
#include <RHReliableDatagram.h>
#include <String.h>
#include <SimpleTimer.h>

SimpleTimer firstTimer(500);
SimpleTimer secondTimer(0);

char perintah[5];
char gedungtujuan[5];
char rate[5], gdg[5], gedungnotif[5], nilainotif[5],pengendalian[5];
String gedung,gedung1;

#define KOORDINATORA 'K'
#define KOORDINATORB 'L'
#define GATEWAY    'G'
#define TXPWR      13
#define ReceivedBufferLength 20
char receivedBuffer[ReceivedBufferLength+1]; // store the serial command
byte receivedBufferIndex = 0;
boolean enterCalibrationFlag = 0;
#define compensationFactorAddress 8
float compensationFactor;
#pragma pack(push, 2)
typedef struct DATA {
    float data1;
    float data2;
    float data3;
    float rate;
```

```

char gedung;
char notifgedung;
float notif;
float datapengendalian;
};

#pragma pack(pop)

DATA data;
RH_RF95 rf95;
const float FREQ = 915.0;
RHDatagram kirim(rf95,KOORDINATORB);
RHDatagram terima(rf95,KOORDINATORB); // as we want read

void setup(){
  Serial.begin(9600);
  SetupLora();
}

void SetupLora(void) {
{
  if (kirim.init()) {
    if (!rf95.setFrequency(FREQ))
      Serial.println("Unable to set RF95 frequency");
    //if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
    //Serial.println("Invalid setModemConfig() option");
    rf95.setTxPower(TXPWR);
    Serial.println("RF radio initialized.");
  }
  else
    Serial.println("RF radio initialization failed.");
  Serial.print("RF95 max message length = ");
  Serial.println(rf95.maxMessageLength());
}
{
  if (terima.init())
  {

```

```

if (!rf95.setFrequency(FREQ))
    Serial.println("Unable to set RF95 frequency");
//if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128))
//Serial.println("Invalid setModemConfig() option");
rf95.setTxPower(TXPWR);
Serial.println("RF95 radio initialized.");
}

else
Serial.println("RF95 radio initialization failed.");
Serial.print("RF95 max message length = ");
Serial.println(rf95.maxMessageLength());
}

}

void kirimdata(void)
{
if(Serial.available()){
int datanotif;
int nilaipengendalian;
Serial.readBytes(gdg,5);
Serial.readBytes(rate,5);
Serial.readBytes(nilainotif,5);
// Serial.readBytes(gedungnotif,5);
Serial.readBytes(pengendalian,5);
datanotif = atoi (nilainotif);
nilaipengendalian = atof (pengendalian);
data.gedung= gdg[0];
data.rate= atof (rate);
Serial.println(data.gedung);
Serial.println(data.rate);
data.notif = datanotif;
Serial.println(datanotif);
data.datapengendalian = nilaipengendalian;
}
}

```

```

    Serial.println(data.datapengendalian);

    if (!kirim.sendto((uint8_t *) &data, sizeof(data), GATEWAY))
        Serial.print("Transmit failed");

    rf95.waitPacketSent(100);
}

}

void terimadata(void)
{
    uint8_t bufLen = sizeof(data);

    char from;

    if (terima.recvfrom((uint8_t *) &data, &bufLen, &from))

    {
        //Serial.print("Got mesaage from : ");
        //Serial.println(from);

        //Serial.println("=====> Terima Perintah 1/0= " + String(data.data1));
        //Serial.println("=====> Gedung = " + String(data.data2)); //data angka urut
        abjad, ex 1=A, 2=B dst

        if (data.data2 == 1) {
            gedung = "A";
        }
        else if (data.data2 == 2) {
            gedung = "B";
        }
        else if (data.data2 == 3) {
            gedung = "C";
        }
        else if (data.data2 == 4) {
            gedung = "D";
        }
        else if (data.data2 == 5) {
            gedung = "P";
        }

        String nilai, ruang;

        ruang = gedung ;
        gedung.toCharArray(gedungtujuan,5);
        nilai = String(data.data1);
        nilai.toCharArray(perintah,5);
    }
}

```

```
Serial.write(perintah,5);
Serial.write(gedungtujuan,5);
} }

void loop()
{
if(secondTimer.isReady()){
kirimdata();
secondTimer.reset();
}
if(firstTimer.isReady()){
terimadata();
firstTimer.reset();
}
}
```

### Lampiran 3 Program *Gateway*

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <SPI.h>
#include <RH_RF95.h>
#include <RHReliableDatagram.h>
#include <String.h>
#include <SimpleTimer.h>
#include "CTBot.h";
#include <ArduinoJson.h>

//JsonArray
const size_t capacity = JSON_ARRAY_SIZE(5) + JSON_OBJECT_SIZE(1) + 5
* JSON_OBJECT_SIZE(2) + 90;
DynamicJsonBuffer jsonBuffer(capacity);

//variabel CTBot untuk notifikasi telegram
CTBot notifikasi;
CTBot notifikasi1;
CTBot notifikasi2;

//type data untuk mendefinisikan per gedung
float rateA, rateB, rateC, rateD, rateP;
int notifA, notifB, notifC, notifD;
int pengendalianA, pengendalianB, pengendalianC, pengendalianD,
pengendalianP;

// Timer jalannya program
SimpleTimer firstTimer(60000);
SimpleTimer secondTimer(0);
```

```

#define RFM95_CS    D2
#define RFM95_RST   D3
#define RFM95_INT   D1
#define KOORDINATORA 'K'
#define KOORDINATORB 'L'
#define GATEWAY     'G'
#define TXPWR       13
#define TXPWR       13

//type data buat pengiriman LoRa
#pragma pack(push, 2)
typedef struct DATA {
    float data1;
    float data2;
    float data3;
    float rate;
    char gedung;
    float notif;
    float datapengendalian;
};

#pragma pack(pop)
DATA data;
const float FREQ = 915.0;
RH_RF95 rf95(RFM95_CS, RFM95_INT);
RH Datagram gate(rf95, GATEWAY);

const char *ssid = "vivo 1904"; // Enter your wifi setting
const char *password = "gratisan";
String token = "1505598997:AAHRZzqNfD8MENU5s2hocvygukRd5e1Ghhs";
String token1 = "1669416128:AAG26schrWq2QO8QjgDRowBKLehincOINI8";
String token2="1542663897:AAHvn88GX0QFCCYXtPFlfmjSsQnCR3WkrWQ";
const int id = 1400163707;

```

```
const int id1 = 1363671361;
const int id2 = 864423087;

String host = "http://pantaukendaliair.com";
WiFiClient client;

void setup(void)
{
    Serial.begin(115200);
    pinMode(RFM95_RST, OUTPUT);
    digitalWrite(RFM95_RST, HIGH);
    connectWifi();
    delay(10);
    connectTelegram();
    delay(10);
    connectTelegram1();
    delay(10);
    connectTelegram2();
    digitalWrite(RFM95_RST, LOW); //Set pin reset Low
    delay(10);
    digitalWrite(RFM95_RST, HIGH); // Set pin reset High
    delay(10);
    SetupLora();
}

void connectWifi(void)
{
    WiFi.begin(ssid,password);
    while (WiFi.status() != WL_CONNECTED){
        delay(500);
        Serial.print(".");
    }
}
```

```
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

void connectTelegram(void)
{
    Serial.begin(115200);
    Serial.println("Memulai Telegram Bot. Koneksi WiFi");
    notifikasi.wifiConnect(ssid, password);
    notifikasi.setTelegramToken(token);

    if(notifikasi.testConnection())
    {
        Serial.println("Koneksi Berhasil");
    }
    else {
        Serial.println("Koneksi Gagal");
    }
}

void connectTelegram1(void)
{
    Serial.begin(115200);
    Serial.println("Memulai Telegram1 Bot. Koneksi WiFi");
    notifikasi1.wifiConnect(ssid, password);
    notifikasi1.setTelegramToken(token1);

    if(notifikasi1.testConnection())
    {
        Serial.println("Koneksi Berhasil");
    }
}
```

```

        }

    else {
        Serial.println("Koneksi Gagal");
    }
}

void connectTelegram2(void)
{
    Serial.begin(115200);
    Serial.println("Memulai Telegram2 Bot. Koneksi WiFi");
    notifikasi2.wifiConnect(ssid, password);
    notifikasi2.setTelegramToken(token2);

    if(notifikasi2.testConnection())
    {
        Serial.println("Koneksi Berhasil");
    }
    else {
        Serial.println("Koneksi Gagal");
    }
}

void SetupLora(void)
{
    if (gate.init())//starting LoRa
    {
        if (!rf95.setFrequency(FREQ)) //Starting set frequency
        Serial.println("Unable to set RF95 frequency");
        rf95.setTxPower(TXPWR);
        Serial.println("RF95 radio initialized.");
    }
    else

```

```

Serial.println("RF95 radio initialization failed.");
Serial.print("RF95 max message length = ");
Serial.println(rf95.maxMessageLength());
}

void loop (void)
{
if (firstTimer.isReady()) {

HTTPClient http; //--> Declare object of class HTTPClient

//-----Getting Data from MySQL Database
String GetAddress, LinkGet, getData, perintah,data1,data2;
GetAddress = "/kontrol/api-status.php";
LinkGet = host + GetAddress; //--> Make a Specify request destination

Serial.println("-----Connect to Server-----");
Serial.println("Get LED Status from Server or Database");
Serial.print("Request Link : ");
Serial.println(LinkGet);
http.begin(LinkGet); //--> Specify request destination
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
//Specify content-type header
int httpCodeGet = http.GET(); //--> Send the request
String payloadGet = http.getString(); //--> Get the response payload from server
Serial.print("Response Code : "); //--> If Response Code = 200 means
Successful connection, if -1 means connection failed. For more information see
here : https://en.wikipedia.org/wiki/List\_of\_HTTP\_status\_codes
Serial.println(httpCodeGet); //--> Print HTTP return code
Serial.print("Returned data from Server : ");

JsonObject& root = jsonBuffer.parseObject(payloadGet);

```

```

JsonArray& result = root["data"];

for (int i = 0; i < 5; i++) {
    perintah = int (result[i]["status"]);
    data1 = perintah[0];
    data2 = perintah[1];
    data.data1 = data1.toFloat();
    data.data2 = data2.toFloat();
    //Kirim data ke Node Koordinator 915Mhz
    Serial.println("perintah 1/0 = " + String(data.data1));
    Serial.println("gedung = " + String(data.data2));
    if (!gate.sendto((uint8_t *) &data, sizeof(data), KOORDINATORB))
        Serial.print("Transmit failed");
    rf95.waitPacketSent(100);
    delay(500);
}

firstTimer.reset();
}

if(secondTimer.isReady()){
    uint8_t bufLen = sizeof(data);
    uint8_t from;
    if (gate.recvfrom((uint8_t *) &data, &bufLen, &from))
    {
        Serial.print("Got message from : ");
        Serial.println(from);
        Serial.println("Asal gedung =" +String(data.gedung));
        Serial.println("Rate gedung = " +String(data.rate));
        Serial.println("Data notifikasi = " +String(data.notif));
        Serial.println("Data pegendalian= " +String(data.datapengendalian));
    }
}

```

```

if(data.gedung == 'A'){
    rateA = data.rate;
    notifA = data.notif;
    pengendalianA = data.datapengendalian;
    Serial.println("Rate gedung A= "+String(rateA));
    Serial.println("Notif gedung A = "+String(notifA));
    Serial.println("Pengendalian gedung A = "+String(pengendalianA));
    if (notifA == 1)
    {
        notifikasi.sendMessage(id,"Ada Kebocoran di Gedung A ");
        notifikasi1.sendMessage(id1,"Ada Kebocoran di Gedung A ");
        notifikasi2.sendMessage(id2,"Ada Kebocoran di Gedung A ");
    }
    else if (pengendalianA == 2){
        notifikasi.sendMessage(id,"Selenoid Gedung A OFF(Air Mati)");
        notifikasi1.sendMessage(id1,"Selenoid Gedung A OFF(Air Mati)");
        notifikasi2.sendMessage(id2,"Selenoid Gedung A OFF(Air Mati)");
    }
}

String postData =(String) "rA=" +rateA;
HTTPClient http; //Declare object of class HTTPClient
String GetAddress, LinkGet;
GetAddress = "/includes/masuk_data.php?";
LinkGet = host + GetAddress+postData;
Serial.println(LinkGet);
http.begin(LinkGet); //Specify request destination
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
//Specify content-type header
int httpCode = http.GET(); //Send the request
String payload = http.getString(); //Get the response payload
Serial.println(httpCode); //Print HTTP return code
Serial.println(payload); //Print request response payload
http.end(); //Close connection

```

```

}

else if(data.gedung == 'B'){

rateB = data.rate;

notifB = data.notif;

pengendalianB = data.datapengendalian;

Serial.println("Rate gedung B= "+String(rateB));

Serial.println("Notif gedung B = "+String(notifB));

Serial.println("Pengendalian gedung B = "+String(pengendalianB));

if (notifB == 1){

    notifikasi.sendMessage(id,"Ada Kebocoran di Gedung B ");

    notifikasi1.sendMessage(id1,"Ada Kebocoran di Gedung B ");

    notifikasi2.sendMessage(id2,"Ada Kebocoran di Gedung B ");

}

else if (pengendalianB == 2){

    notifikasi.sendMessage(id,"Selenoid Gedung B OFF(Air Mati)");

    notifikasi1.sendMessage(id1,"Selenoid Gedung B OFF(Air Mati)");

    notifikasi2.sendMessage(id2,"Selenoid Gedung B OFF(Air Mati");

}

String postData =(String) "rB=" +rateB;

HTTPClient http; //Declare object of class HTTPClient

String GetAddress, LinkGet;

GetAddress = "/includes/masuk_data.php?";

LinkGet = host + GetAddress+postData;

Serial.println(LinkGet);

http.begin(LinkGet); //Specify request destination

http.addHeader("Content-Type", "application/x-www-form-urlencoded");

//Specify content-type header

int httpCode = http.GET(); //Send the request

String payload = http.getString(); //Get the response payload

Serial.println(httpCode); //Print HTTP return code

Serial.println(payload); //Print request response payload

http.end(); //Close connection

```

```

        }

else if(data.gedung == 'C'){

    rateC = data.rate;
    notifC = data.notif;
    pengendalianC = data.datapengendalian;
    Serial.println("Rate gedung C= "+String(rateC));
    Serial.println("Notif gedung C = "+String(notifC));
    Serial.println("Pengendalian gedung C = "+String(pengendalianC));
    if (notifC == 1){

        notifikasi.sendMessage(id,"Ada Kebocoran di Gedung C ");
        notifikasi1.sendMessage(id1,"Ada Kebocoran di Gedung C ");
        notifikasi2.sendMessage(id2,"Ada Kebocoran di Gedung C ");
    }

else if (pengendalianC == 2){

    notifikasi.sendMessage(id,"Selenoid Gedung C OFF(Air Mati)");
    notifikasi1.sendMessage(id1,"Selenoid Gedung C OFF(Air Mati)");
    notifikasi2.sendMessage(id2,"Selenoid Gedung C OFF(Air Mati");
}

String postData =(String) "rC=" +rateC;
HTTPClient http; //Declare object of class HTTPClient
String GetAddress, LinkGet;
GetAddress = "/includes/masuk_data.php?";
LinkGet = host + GetAddress+postData;
Serial.println(LinkGet);
http.begin(LinkGet); //Specify request destination
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
//Specify content-type header

int httpCode = http.GET(); //Send the request
String payload = http.getString(); //Get the response payload
Serial.println(httpCode); //Print HTTP return code
Serial.println(payload); //Print request response payload

```

```

        http.end(); //Close connection
    }

else if(data.gedung == 'D'){

    rateD = data.rate;
    notifD = data.notif;
    pengendalianD = data.datapengendalian;
    Serial.println("Rate gedung D= "+String(rateD));
    Serial.println("Notif gedung D = "+String(notifD));
    Serial.println("Pengendalian gedung D = "+String(pengendalianD));
    if (notifD == 1){

        notifikasi.sendMessage(id,"Ada Kebocoran di Gedung D ");
        notifikasi1.sendMessage(id1,"Ada Kebocoran di Gedung D ");
        notifikasi2.sendMessage(id2,"Ada Kebocoran di Gedung D ");
    }

else if (pengendalianD == 2){

    notifikasi.sendMessage(id,"Selenoid Gedung D OFF(Air Mati)");
    notifikasi1.sendMessage(id1,"Selenoid Gedung D OFF(Air Mati)");
    notifikasi2.sendMessage(id2,"Selenoid Gedung D OFF(Air Mati");
}

String postData =(String) "rD=" +rateD;
HTTPClient http; //Declare object of class HTTPClient
String GetAddress, LinkGet;
GetAddress = "/includes/masuk_data.php?";
LinkGet = host + GetAddress+postData;
Serial.println(LinkGet);
http.begin(LinkGet); //Specify request destination
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
//Specify content-type header
int httpCode = http.GET(); //Send the request
String payload = http.getString(); //Get the response payload
Serial.println(httpCode); //Print HTTP return code
Serial.println(payload); //Print request response payload

```

```

        http.end(); //Close connection
    }

else if(data.gedung == 'P'){
    rateP = data.rate;
    pengendalianP = data.datapengendalian;
    Serial.println("Rate gedung P= "+String(rateP));
    Serial.println("Pengendalian gedung P = "+String(pengendalianP));
    if (pengendalianP == 2){
        notifikasi.sendMessage(id,"Selenoid Pusat OFF(Air Mati)");
        notifikasi1.sendMessage(id1,"Selenoid Pusat OFF(Air Mati)");
        notifikasi2.sendMessage(id2,"Selenoid Pusat OFF(Air Mati)");
    }
    String postData =(String) "rP=" +rateP;
    HTTPClient http; //Declare object of class HTTPClient
    String GetAddress, LinkGet;
    GetAddress = "/includes/masuk_data.php?";
    LinkGet = host + GetAddress+postData;
    Serial.println(LinkGet);
    http.begin(LinkGet); //Specify request destination
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");
    //Specify content-type header
    int httpCode = http.GET(); //Send the request
    String payload = http.getString(); //Get the response payload
    Serial.println(httpCode); //Print HTTP return code
    Serial.println(payload); //Print request response payload
    http.end(); //Close connection
}
}

secondTimer.reset();
}
}

```