



UNIVERSITAS INDONESIA

**IMPLEMENTASI ZIGBEE GATEWAY BERBASIS GUI MENGGUNAKAN
RASPBERRY PI**

TUGAS AKHIR

**LUQMAN SUNGKAR
1106088303**

**FAKULTAS ILMU KOMPUTER
PROGRAM STUDI SISTEM INFORMASI
DEPOK
JANUARI 2010**



UNIVERSITAS INDONESIA

**IMPLEMENTASI ZIGBEE GATEWAY BERBASIS GUI MENGGUNAKAN
RASPBERRY PI**

TUGAS AKHIR

**Diajukan sebagai salah satu syarat untuk memperoleh gelar
Sarjana Ilmu Komputer**

LUQMAN SUNGKAR

1106088303

**FAKULTAS ILMU KOMPUTER
PROGRAM STUDI SISTEM INFORMASI
DEPOK
JANUARI 2010**

HALAMAN PERSETUJUAN

Judul : Implementasi ZigBee *Gateway* Berbasis GUI Menggunakan Raspberry Pi
Nama : Luqman Sungkar
NPM : 1106088303

Laporan Tugas Akhir ini telah diperiksa dan disetujui.

XX Januari 2010

Prof. XXXX
Pembimbing Tugas Akhir

HALAMAN PERNYATAAN ORISINALITAS

**Tugas Akhir ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.**

Nama : Luqman Sungkar

NPM : 1106088303

Tanda Tangan :

Tanggal : XX Januari 2010

HALAMAN PENGESAHAN

Tugas Akhir ini diajukan oleh :

Nama : Luqman Sungkar
NPM : 1106088303
Program Studi : Sistem Informasi
Judul Tugas Akhir : Implementasi ZigBee *Gateway* Berbasis GUI Menggunakan Raspberry Pi

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Ilmu Komputer pada Program Studi Sistem Informasi, Fakultas Fakultas Ilmu Komputer, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Prof. XXXX ()

Penguji : Prof. XXX ()

Penguji : Prof. XXXX ()

Penguji : Prof. XXXXXX ()

@todo

Jangan lupa mengisi nama para penguji.

Ditetapkan di : Depok

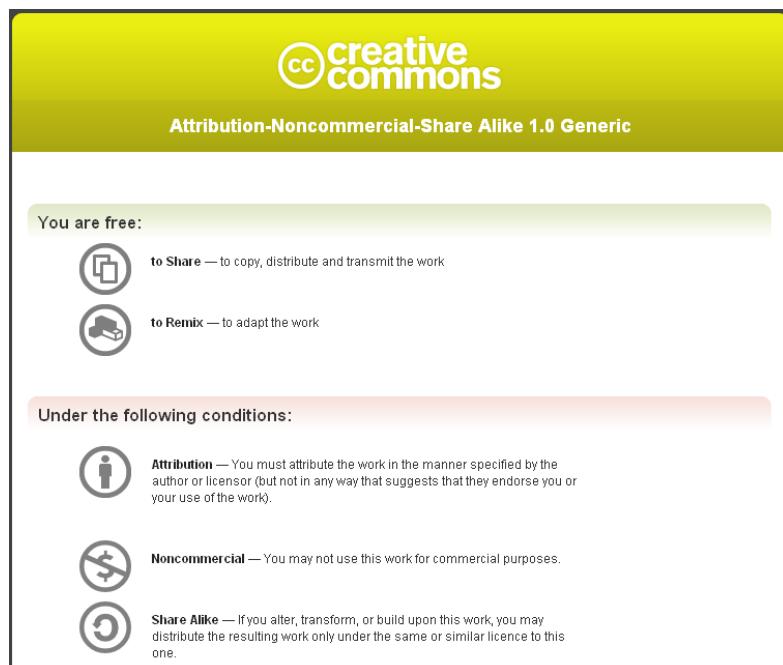
Tanggal : XX Januari 2010

KATA PENGANTAR

Template ini disediakan untuk orang-orang yang berencana menggunakan L^AT_EX untuk membuat dokumen tugas akhirnya. Mengapa L^AT_EX? Ada banyak hal mengapa menggunakan L^AT_EX, diantaranya:

1. L^AT_EX membuat kita jadi lebih fokus terhadap isi dokumen, bukan tampilan atau halaman.
2. L^AT_EX memudahkan dalam penulisan persamaan matematis.
3. Adanya automatis dalam penomoran caption, bab, subbab, subsubbab, referensi, dan rumus.
4. Adanya automatisasi dalam pembuatan daftar isi, daftar gambar, dan daftar tabel.
5. Adanya kemudahan dalam memberikan referensi dalam tulisan dengan menggunakan label. Cara ini dapat meminimalkan kesalahan pemberian referensi.

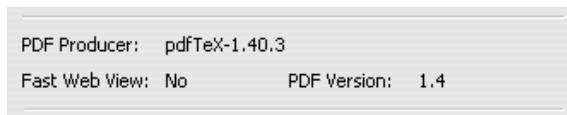
Template ini bebas digunakan dan didistribusikan sesuai dengan aturan *Creative Common License 1.0 Generic*, yang secara sederhana berisi:



Gambar 1: *Creative Common License 1.0 Generic*

Gambar 1 diambil dari http://creativecommons.org/licenses/by-nc-sa/1.0/deed.en_CA. Jika ingin mengentahui lebih lengkap mengenai *Creative Common License 1.0 Generic*, silahkan buka <http://creativecommons.org/licenses/by-nc-sa/1.0/legalcode>. Seluruh dokumen yang dibuat dengan menggunakan template ini sepenuhnya menjadi hak milik pembuat dokumen dan bebas didistribusikan sesuai dengan keperluan masing-masing. Lisensi hanya berlaku jika ada orang yang membuat template baru dengan menggunakan template ini sebagai dasarnya.

Dokumen ini dibuat dengan \LaTeX juga. Untuk meyakinkan Anda, coba lihat properti dari dokumen ini dan Anda akan menemukan bagian seperti Gambar 2. Dokumen ini dimaksudkan untuk memberikan gambaran kepada Anda seperti apa mudahnya menggunakan \LaTeX dan juga memperlihatkan betapa bagus dokumen yang dihasilkan. Seluruh url yang Anda temukan dapat Anda klik. Seluruh referensi yang ada juga dapat diklik. Untuk mengerti template yang disediakan, Anda tetap harus membuka kode \LaTeX dan bermain-main dengannya. Penjelasan dalam PDF ini masih bersifat gambaran dan tidak begitu mendetail, dapat dianggap sebagai pengantar singkat. Jika Anda merasa kesulitan dengan template ini, mungkin ada baiknya Anda belajar sedikit dasar-dasar \LaTeX .



Gambar 2: Dokumen Dibuat dengan PDFLatex

Semoga template ini dapat membantu orang-orang yang ingin mencoba menggunakan \LaTeX . Semoga template ini juga tidak berhenti disini dengan ada kontribusi dari para penggunanya. Kami juga ingin berterima kasih kepada Andreas Febrian, Lia Sadita, Fahrurrozi Rahman, Andre Tampubolon, dan Erik Dominikus atas kontribusinya dalam template ini.

Depok, 30 Desember 2009

Luqman Sungkar

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Luqman Sungkar
NPM : 1106088303
Program Studi : Sistem Informasi
Fakultas : Fakultas Ilmu Komputer
Jenis Karya : Tugas Akhir

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty Free Right)** atas karya ilmiah saya yang berjudul:

Implementasi ZigBee *Gateway* Berbasis GUI Menggunakan Raspberry Pi beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : XX Januari 2010
Yang menyatakan

(Luqman Sungkar)

ABSTRAK

Nama : Luqman Sungkar
Program Studi : Sistem Informasi
Judul : Implementasi ZigBee *Gateway* Berbasis GUI Menggunakan Raspberry Pi

@todo

Tuliskan abstrak laporan disini.

Kata Kunci:

@todo

Tuliskan kata kunci yang berhubungan dengan laporan disini

ABSTRACT

Name : Luqman Sungkar

Program : Sistem Informasi

Title : GUI Based ZigBee Gateway Implementation using Raspberry Pi

@todo

Write your abstract here.

Keywords:

@todo

Write up keywords about your report here.

DAFTAR ISI

HALAMAN JUDUL	i
LEMBAR PERSETUJUAN	ii
LEMBAR PERNYATAAN ORISINALITAS	iii
LEMBAR PENGESAHAN	iv
KATA PENGANTAR	v
LEMBAR PERSETUJUAN PUBLIKASI ILMIAH	vii
ABSTRAK	viii
Daftar Isi	x
Daftar Gambar	xiii
Daftar Tabel	xiv
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	3
1.4 Posisi Penelitian	3
1.5 Tahapan Pengerjaan	4
1.6 Ruang Lingkup Pengerjaan	5
1.7 Sistematika Penulisan Laporan	5
2 LANDASAN TEORI	7
2.1 <i>Internet of Things</i>	7
2.2 <i>Social internet of things</i>	8
2.3 ZigBee	9
2.4 Gateway	10
2.4.1 ZigBee Gateway	11
2.5 MQTT	12

2.5.1	Sistem <i>Publish-Subscribe</i>	12
2.5.2	Topik dalam MQTT	13
3	RANCANGAN IMPLEMENTASI	15
3.1	Gambaran Umum Sistem	15
3.1.1	Gambaran Umum Perangkat <i>Gateway</i>	17
3.2	Rancangan Implementasi <i>Gateway</i>	19
3.2.1	Perangkat yang digunakan	20
3.2.2	<i>Tools</i> yang digunakan	20
3.2.2.1	deCONZ	20
3.2.2.2	Mosquitto	22
3.2.2.3	Eclipse Paho	22
3.2.3	Rancangan Mekanisme Pengiriman Pesan	22
3.2.4	Rancangan Topik	23
3.2.5	Rancangan Translasi Pesan MQTT Menjadi Perintah REST	24
3.2.6	Rancangan Mekanisme Penggunaan Oleh Pengguna	27
3.2.6.1	Cara Pengguna Terhubung ke Perangkat <i>Gateway</i>	28
3.2.6.2	Cara Melakukan <i>Log In</i> Melalui Perangkat <i>Gate-</i> <i>way</i>	28
3.2.6.3	Cara Mendaftarkan Lampu atau <i>Group</i> ke <i>Platform</i>	28
3.3	Rancangan Pengujian	29
4	IMPLEMENTASI	31
4.1	Persiapan Alat	31
4.2	Implementasi <i>Gateway</i>	33
4.2.1	Implementasi MQTT <i>Client</i>	33
4.2.2	Translasi Pesan MQTT ke Pesan REST	36
4.3	Implementasi Halaman Kontrol Lampu	37
4.3.1	Halaman Utama Untuk Mengontrol Lampu	38
4.3.1.1	Melihat Daftar Lampu yang Terhubung	38
4.3.1.2	Mengubah Atribut Lampu	38
4.3.1.3	Membuat <i>Group</i> Lampu	40
4.3.1.4	<i>Log In</i> Pengguna	40
4.3.2	Implementasi Halaman Registrasi Lampu	41
4.4	Implementasi Perangkat <i>Gateway</i>	43
4.4.1	Konfigurasi <i>Start Up</i>	44
4.4.2	Konfigurasi <i>Access Point</i>	45

5 PENGUJIAN	47
5.1 Perangkat yang Digunakan	47
5.1.1 Hasil Pengujian	47
6 PENUTUP	67
6.1 Hasil Pengujian	67
6.2 Kesimpulan	67
LAMPIRAN	1
Lampiran 1	2

DAFTAR GAMBAR

1	<i>Creative Common License 1.0 Generic</i>	v
2	Dokumen Dibuat dengan PDFLatex	vi
1.1	Arsitektur <i>platform</i> yang dibuat	3
2.1	Arsitektur <i>Gateway ZigBee</i> [2]	11
2.2	Contoh topik dalam MQTT [8]	13
2.3	Contoh <i>single level wildcard</i> [8]	14
2.4	Contoh <i>multi level wildcard</i> [8]	14
3.1	Gambaran Umum <i>Platform</i> Internet of Things berbasis media sosial	15
3.2	Gambaran Umum Sistem	18
3.3	Gambaran Umum Gateway	19
3.4	Tampilan Aplikasi deCONZ berbasis GUI	21
3.5	Tampilan Aplikasi deCONZ berbasis web	21
3.6	Rancangan Mekanisme Pengiriman Pesan	23
4.1	Raspberry Pi Model B	31
4.2	Perangkat RaspBee yang digunakan	32
4.3	Perangkat RaspBee ketika dipasang pada Raspberry Pi	32
5.1	Lampu Philips Hue yang digunakan untuk pengujian	47

DAFTAR TABEL

3.1	Contoh pesan dan topik untuk menghidupkan lampu	24
3.2	Contoh pesan dan topik untuk mematikan lampu	25
3.3	Contoh pesan dan topik untuk mengubah tingkat kecerahan lampu .	25
3.4	Contoh pesan dan topik untuk mengubah warna lampu	25
3.5	Contoh pesan dan topik untuk mengubah tingkat kejemuhan lampu .	26
3.6	Contoh pesan dan topik untuk menghidupkan lampu di dalam suatu <i>group</i>	26
3.7	Contoh pesan dan topik untuk mematikan lampu di dalam suatu <i>group</i> .	26
3.8	Contoh pesan dan topik untuk mengubah tingkat kecerahan lampu di dalam suatu <i>group</i>	27
3.9	Contoh pesan dan topik untuk mengubah warna lampu di dalam suatu <i>group</i>	27
3.10	Contoh pesan dan topik untuk mengubah tingkat kejemuhan lampu di dalam suatu <i>group</i>	27
5.1	Hasil pengujian kasus uji 1	48
5.2	Hasil pengujian kasus uji 2	48
5.3	Hasil pengujian kasus uji 3	49
5.4	Hasil pengujian kasus uji 4	49
5.5	Hasil pengujian kasus uji 5	50
5.6	Hasil pengujian kasus uji 6	50
5.7	Hasil pengujian kasus uji 7	51
5.8	Hasil pengujian kasus uji 8	51
5.9	Hasil pengujian kasus uji 9	52
5.10	Hasil pengujian kasus uji 10	52
5.11	Hasil pengujian kasus uji 11	53
5.12	Hasil pengujian kasus uji 12	54
5.13	Hasil pengujian kasus uji 13	55
5.14	Hasil pengujian kasus uji 14	55
5.15	Hasil pengujian kasus uji 15	56
5.16	Hasil pengujian kasus uji 16	56
5.17	Hasil pengujian kasus uji 17	57
5.18	Hasil pengujian kasus uji 18	58

5.19 Hasil pengujian kasus uji 19	59
5.20 Hasil pengujian kasus uji 20	60
5.21 Hasil pengujian kasus uji 21	61
5.22 Hasil pengujian kasus uji 22	62
5.23 Hasil pengujian kasus uji 23	63
5.24 Hasil pengujian kasus uji 24	64
5.25 Hasil pengujian kasus uji 25	65
5.26 Hasil pengujian kasus uji 26	66

BAB 1

PENDAHULUAN

@todo

posisi penelitian, list apa aja yg di maksud buat end user

Pada bagian ini akan dijelaskan latar belakang dari pengerajan tugas akhir ini, tujuan penulisan, rumusan masalah, ruang lingkup dan batasan pengerajan, tahapan pengerajan, dan sistematika penulisan laporan.

1.1 Latar Belakang

Penemuan internet beberapa dekade lalu telah merubah cara manusia dalam hal bertukar informasi. Popularitas dari internet terus meningkat hingga pada akhirnya, menurut data dari cisco[5], jumlah perangkat yang terhubung ke internet pada tahun 2010 telah melewati jumlah dari populasi manusia yang ada di Bumi. Lebih jauh lagi, diperkirakan pada tahun 2015 ini jumlah perangkat yang terhubung ke internet akan menjadi dua kali lipat dari jumlah populasi manusia. Hal ini disebabkan oleh teknologi komputer terbenam (komputer yang tidak terlihat keberadaannya) yang akan memicu lahirnya era baru dalam sejarah internet, era *internet of things*.

Saat ini, berbagai perangkat yang menerapkan konsep *internet of things* telah bermunculan di pasaran. Beberapa *platform* yang bertujuan menghubungkan berbagai perangkat *internet of things*-pun mulai bermunculan di pasaran. Berbagai *platform* tersebut memiliki berbagai pendekatan, mulai dari yang menargetkan *end-user* dengan model otomasi, sampai yang menargetkan konsumen *enterprise*. Salah satu pendekatan yang belum banyak terlihat keberadaanya di pasaran adalah pendekatan dengan model sosial media dan menargetkan pada *end-user*. Oleh karena itu, penulis dan teman-teman penulis tertarik untuk mengembangkan sebuah *platform internet of things* yang menggunakan model sosial media dan ditargetkan untuk *end-user*.

Untuk membuat sebuah *platform* yang menargetkan pada *end-user*, diperlukan cara untuk menghubungkan berbagai perangkat yang dimiliki konsumen dengan cara yang mudah. Mekanisme pendaftaran ini harus bisa menangani baik perangkat yang sudah beredar dipasaran maupun yang nantinya akan beredar di pasaran. Konsumen juga mungkin akan menginginkan agar perangkat yang mereka miliki walaupun berbeda merek, namun bisa dikendalikan secara bersamaan. Untuk

itu, dibutuhkan sebuah *gateway* yang dapat menghubungkan berbagai perangkat berbeda ke *platform* yang diinginkan melalui internet.

Nisrina Luthfiyati dalam tugas akhirnya yang berjudul "Implementasi ZigBee *Coordinator* dengan REST *Interface*"[12] telah berhasil membuat sebuah *coordinator* untuk perangkat berbasis ZigBee yang dapat diakses melalui perintah REST. Fauziah Rahmawati kemudian dalam tugas akhirnya yang berjudul "Implementasi *Home Automation Gateway* untuk IOT *Cloud Service* berbasis ZigBee *Network*"[14] berhasil melanjutkan tugas akhir Nisrina Luthfiyati dan membuat sebuah *gateway* yang dapat menghubungkan perangkat berbasis ZigBee dengan *Home Automation Profile* ke internet. *Gateway* yang digunakan oleh Fauziah Rahmawati menerapkan konsep *publish-subscribe* dalam melakukan pengiriman data ke internet.

Namun, penulis merasa masih ada yang bisa ditingkatkan dari implementasi *gateway* yang telah dibuat oleh Fauziah, terutama jika akan digunakan pada *platform internet of things* yang akan dibuat oleh penulis dan teman-teman penulis yang menargetkan pada *end-user*. Implementasi *gateway* yang telah dibuat oleh Fauziah masih cukup rumit untuk digunakan dan masih membutuhkan suatu komputer terpisah yang terus menyala untuk bisa digunakan. Sedangkan jika ingin bisa digunakan oleh *end-user*, *gateway* harus bisa mendaftarkan perangkat yang dimiliki konsumen dengan cara yang mudah dan bisa terhubung dengan pengelola perangkat yang ada pada *platform* terkait. *Gateway* harus menyertakan identitas ketika terhubung ke *platform* sehingga dapat dibedakan antar *gateway* yang dimiliki oleh *user* berbeda. *Gateway* juga harus memiliki bentuk yang ringkas dan tidak memerlukan komputer tambahan sehingga tidak perlu ada komputer konsumen yang menyala terus-menerus. Oleh karena itu, penulis terpikir untuk membuat sebuah implementasi ZigBee *gateway* berbasis GUI (*Graphical User Interface*) dengan menggunakan Raspberry Pi untuk menyelesaikan masalah yang disebutkan sebelumnya.

1.2 Rumusan Masalah

Masalah yang ingin diselesaikan dalam tugas akhir ini adalah bagaimana membuat sebuah *gateway* yang dapat menghubungkan perangkat yang dimiliki konsumen dengan *platform internet of things* yang menggunakan pendekatan sosial media. *Gateway* harus bisa digunakan oleh *end-user* dengan mudah dan dapat berkomunikasi dengan *platform* yang diinginkan. Secara spesifik, rumusan masalah dalam tugas akhir ini dapat dituangkan ke dalam pertanyaan-pertanyaan berikut:

1. Bagaimana membuat ZigBee *gateway* berbasis GUI dengan menggunakan

Raspberry Pi?

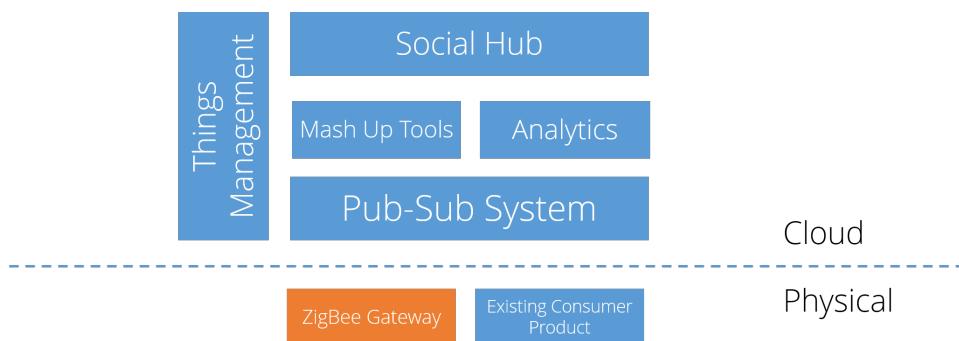
2. Bagaimana model komunikasi yang baik agar *gateway* dapat terhubung dengan *platform internet of things* berbasis sosial media?

1.3 Tujuan

Tujuan dari tugas akhir ini adalah menghasilkan sebuah perangkat *gateway* yang dapat mengirimkan informasi dari jaringan lampu ZigBee ke internet. Perangkat *gateway* yang dihasilkan dapat dioperasikan menggunakan GUI. *Gateway* juga dikembangkan dengan berpatokan pada *platform internet of things* yang berbasis sosial media.

1.4 Posisi Penelitian

Tugas akhir ini merupakan salah satu bagian dari satu topik besar, yaitu *Social Internet of Things platform* dengan pendekatan menggunakan *social media*. Penggerjaan topik besar ini dilakukan dalam satu kelompok yang dibimbing oleh Bapak Bob Hardian, seorang dosen di Fakultas Ilmu Komputer UI. Arsitektur *platform* yang akan dibuat bisa dilihat pada gambar 1.1.



Gambar 1.1: Arsitektur *platform* yang dibuat

Penjelasan lebih mendetail akan dijelaskan pada Bab 3. Pembagian topik lebih spesifik dari topik besar tersebut adalah:

1. *Social Hub* akan dikerjakan oleh Jouvy Alif Pradewo.
2. *Mash Up Tools* akan dikerjakan oleh Muhammad Redho Ayassa.
3. *Analytics* Bagian ini belum akan dikerjakan.
4. *Pub-Sub System* akan dikerjakan oleh Abdullah Izzuddin Alqassam.

5. *Things Management* akan dikerjakan oleh Prakoso Adi Nugroho.
6. *ZigBee Gateway* akan dikerjakan oleh penulis , Luqman Sungkar dalam tugas akhir ini.

1.5 Tahapan Pengerjaan

Pengerjaan dari tugas akhir ini akan dilakukan dengan tahapan sebagai berikut:

1. Studi Literatur

Sebelum memulai merancang dan melakukan implementasi, penulis harus memahami beberapa konsep yang berkaitan dengan apa yang ingin penulis buat. Beberapa konsep tersebut diantaranya, *internet of things*, ZigBee, *MQTT*, *social internet of things*, dan *gateway*. Penulis juga akan mempelajari implementasi *coordinator* yang telah dikerjakan oleh Nisrina Luthfiyati[12] dan implementasi *gateway* yang telah dikerjakan oleh Fauziah Rahmawati[14].

2. Analisis dan Perancangan

Setelah melakukan studi literatur, penulis akan menganalisis kebutuhan dari *gateway* yang akan dibuat. Dari hasil analisis tersebut, penulis akan merancang implementasi dari *gateway* tersebut. Hal yang akan ditentukan dalam perancangan ini mencakup skema komunikasi antara *gateway* dengan *platform* yang digunakan, skema komunikasi di dalam *gateway*, serta desain pesan yang akan dikirim antara *gateway* dengan *platform* yang digunakan.

3. Implementasi

Setelah berhasil melakukan perancangan, penulis akan melakukan implementasi sesuai rancangan yang dibuat. Hasil rancangan akan berupa perangkat *gateway* dan implementasi *software*. Pada tahap ini, penulis akan menggunakan *tools* dan SDK yang sudah ada untuk memudahkan proses implementasi

4. Uji Coba

Setelah melakukan implementasi, penulis harus menguji perangkat yang sudah dibuat untuk memastikan apakah perangkat berjalan sesuai dengan kebutuhan yang telah ditentukan.

5. Penarikan Kesimpulan

Setelah melakukan pengujian terhadap hasil implementasi, penulis akan melakukan analisis terhadap hasil pengujian, dan akhirnya dapat menarik kesimpulan dari seluruh kegiatan pengerjaan tugas akhir ini.

1.6 Ruang Lingkup Pengerjaan

Ruang lingkup implementasi *ZigBee gateway* untuk *platform internet of things* berbasis sosial media dengan target konsumen *end-user* adalah sebagai berikut:

1. Perangkat yang akan bisa dihubungkan dibatasi pada perangkat yang menggunakan implementasi ZigBee dengan *profile Light Link*.
2. Implementasi *ZigBee coordinator* yang digunakan adalah implementasi yang telah disediakan oleh pihak dresden, yaitu deCONZ.
3. Implementasi *gateway* akan berdasarkan pada implementasi *gateway* yang telah dibuat oleh Fauziah Rahmawati[14].
4. *Platform* yang akan digunakan sebagai acuan adalah *platform* yang disebutkan dalam posisi penelitian.
5. Perangkat yang digunakan adalah Raspberry Pi Model B dengan sistem operasi Raspbian.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan tugas akhir ini adalah sebagai berikut:

1. Bab 1 PENDAHULUAN

Bab ini berisi latar belakang pengerjaan tugas akhir, rumusan-rumusan masalah, ruang lingkup pengerjaan, tujuan dari tugas akhir, tahapan pengerjaan yang akan dijalani oleh penulis, dan sistematika dari penulisan laporan ini.

2. Bab 2 LANDASAN TEORI

Bab ini akan menjelaskan beberapa konsep yang diperlukan untuk mengerjakan tugas akhir ini. Konsep-konsep tersebut, diantaranya adalah *internet of things*, ZigBee yang meliputi definisi dan jenis perangkat, MQTT beserta mekanisme *publish-subscribe*, dan *gateway*.

3. Bab 3 RANCANGAN IMPLEMENTASI

Bab ini akan menjelaskan tentang rancangan dari implementasi *gateway* yang akan dibuat. Pada bab ini juga akan dijelaskan skema komunikasi antara *gateway* dengan *platform* yang akan digunakan.

4. Bab 4 IMPLEMENTASI

Bab ini akan menjelaskan tentang implementasi dari *gateway* meliputi konfigurasi *gateway*, dan bagaimana cara menggunakan *gateway* yang dibuat. Pada bab ini juga akan dijelaskan mengenai kode sumber yang diimplementasikan oleh penulis.

5. Bab 5 PENGUJIAN

Bab ini akan menjelaskan tentang mekanisme pengujian yang dilakukan penulis terhadap *gateway* yang dibuat. Pada bab ini juga hasil dari pengujian akan ditampilkan dan dilakukan analisis dari hasil pengujian tersebut.

6. Bab 6 KESIMPULAN DAN SARAN

Bab ini akan memberikan kesimpulan dari hasil implementasi *gateway* yang dilakukan oleh penulis. Penulis juga akan memberikan saran dan ide yang dapat digunakan untuk pengembangan selanjutnya.

BAB 2

LANDASAN TEORI

Bab ini akan menjelaskan beberapa teori dan konsep yang perlu dipahami untuk mengerjakan tugas akhir ini. Beberapa diantaranya adalah *internet of things* , Zig-Bee, *gateway*, MQTT, dan *social internet of things*.

2.1 *Internet of Things*

Internet of Things adalah suatu konsep dimana berbagai macam benda bisa terhubung ke internet. Benda-benda ini bisa bervariasi, mulai dari benda yang memang pada dasarnya sudah bisa terhubung ke internet seperti *smartphone*, hingga benda seperti lampu, kenop pintu,gembok, sensor-sensor, dan berbagai benda lain. Pada konsep ini, selain dapat terhubung ke internet, benda-benda tersebut juga dapat saling berkomunikasi tanpa bantuan manusia, sehingga membentuk suatu jaringan komunikasi antar benda. Implementasi konsep *internet of things* saat ini sudah sangat beragam, seperti pada bidang *home automation*, bidang kesehatan, bidang retail, dan bidang industri. Beberapa contoh produk populer yang sudah tersedia di pasaran saat ini sebagian besar fokus pada bidang *home automation*, seperti Philips Hue, Nest, WeMo, dan SmartThings.

Istilah *internet of things* pertama kali disebutkan pada tahun 1999 oleh Kevin Ashton dalam konteks manajemen rantai suplai dengan idenya untuk menggunakan RFID pada barang milik P&G [3]. Ia mengatakan bahwa komputer saat ini hampir seluruhnya bergantung pada manusia untuk mendapatkan informasi seperti melalui data yang diketikkan, mengambil gambar dari pemindai, menekan suatu tombol, dan lainnya. Masalahnya adalah, manusia memiliki waktu, perhatian, dan ketepatan yang terbatas. Bayangkan jika komputer bisa mengetahui segalanya tentang suatu hal tanpa bantuan sama sekali dari manusia. Kita bisa menghitung dan melacak segala hal, dan bisa mengurangi pengeluaran, dan kerugian secara besar-besaran. Kita bisa mencapai hal ini dengan membuat agar sebuah komputer dapat melihat, mendengar, dan membau lingkungan di sekitarnya dengan sendirinya, tanpa batasan data yang harus dimasukkan oleh manusia[3]. Saat ini, istilah *internet of things* tidak hanya terbatas pada konteks manajemen rantai suplai dan mempunyai kemungkinan implementasi yang lebih luas seperti pada bidang kesehatan, transportasi, pendidikan, manajemen energi, manajemen limbah, layanan publik, dan

lainnya.

Saat ini juga sudah ada beberapa definisi mengenai *internet of things*. RFID Group mendefinisikan *internet of things* sebagai jaringan berskala dunia yang terdiri dari objek yang saling terhubung dan memiliki alamat yang unik berdasarkan protokol komunikasi yang standar[10]. Sebuah paper oleh Jayavardhana Gubbi et al.[10] juga mendefinisikan *internet of things* sebagai perangkat yang dapat bergerak dan merasakan keadaan di sekitarnya yang saling terhubung dan dapat berbagi informasi antar berbagai *platform* yang berbeda melalui sebuah framework yang tunggal, yang memberikan gambaran menyeluruh yang dapat membuat munculnya aplikasi inovatif. Hal ini bisa dicapai dengan menggunakan sensor *ubiquitous* yang terus bekerja, analisa data, dan representasi informasi menggunakan *cloud computing*.

Singkatnya, walaupun definisi '*things*' (benda) saat ini sudah berubah, tapi tujuan dari konsep ini menurut Kevin Ashton masih tetap sama, yaitu membuat sebuah mesin agar dapat merasakan apa yang terjadi di sekitar mereka, mengambil informasi dari apa yang terjadi di sekitar mereka, mengolah informasi tersebut, mengirimkannya ke mesin lain, dan kemudian memberikan respons yang sesuai. Semua hal tersebut dilakukan tanpa ada bantuan dari manusia.

2.2 *Social internet of things*

Social internet of things adalah sebuah konsep yang mencoba menghubungkan konsep *internet of things* dengan konsep sosial. Terdapat beberapa pendekatan mengenai konsep *social internet of things* ini. Luigi Atzori et al dalam papernya [11] menawarkan sebuah konsep yang mencoba memberikan hubungan diantara benda-benda, bukan diantara pemiliknya. Hubungan antara benda dengan benda ini meniru hubungan antara manusia dengan manusia. Sebagai contoh, salah satu hubungan yang paling dasar antara manusia adalah hubungan orang tua dengan anak. Dalam papernya, Luigi Atzori et al menamakan hubungan ini sebagai *parental object relationship*, yaitu hubungan antara benda dengan benda lain yang dibuat dalam periode waktu yang sama dan dibuat oleh perusahaan yang sama. Contoh hubungan lain yang ditawarkan dalam paper ini adalah *co-location object relationship* dan *co-work object relationship* seperti yang dilakukan manusia dalam kehidupannya di suatu tempat yang sama atau ketika sedang dalam pekerjaan. Hubungan tersebut ditentukan ketika beberapa benda berada dalam lokasi yang dekat dalam waktu cukup lama atau secara periodik melakukan hubungan untuk melakukan suatu tugas tertentu. Hubungan lain yang cukup menarik yang disebutkan oleh Luigi Atzori et

al dalam papernya adalah *social object relationship*. Sama seperti hubungan yang dialami oleh manusia ketika berkenalan dengan manusia lain, pada hubungan ini suatu benda juga dapat bertukar profil sosial dengan benda lain. Melalui hal ini, sebuah benda dapat saling bertukar *best practice* untuk menyelesaikan suatu masalah yang sudah dialami oleh "teman"nya.

Contoh pendekatan lain adalah seperti yang dibuat oleh Dominique Guinard et al [7]. Dalam papernya, Dominique menawarkan sebuah konsep untuk memanfaatkan hubungan manusia yang terdapat pada layanan sosial media yang sudah ada di pasaran seperti facebook, twitter, dan lainnya. Melalui konsep ini, seorang pengguna dapat membagikan *resource* dari suatu benda yang dimilikinya ke temannya yang terdapat pada suatu media sosial yang sama. Pembagian *resource* ini bisa hanya membagikan status dari *resource* terkait saja (*read only*) atau juga bisa memberikan orang lain akses untuk melakukan kontrol atau merubah status dari benda terkait.

2.3 ZigBee

Zigbee adalah suatu standar yang mendefinisikan sekumpulan protokol komunikasi untuk jaringan nirkabel dengan *data rate* rendah dan jarak yang dekat [6]. ZigBee beroperasi di pita frekuensi 868 MHz, 915 MHz, dan 2.4GHz. *Data rate* maksimal dari ZigBee adalah 250 kilo bit per detik.

Walaupun memiliki *data rate* yang rendah, ZigBee memiliki kelebihan dibanding dengan protokol komunikasi lain, yaitu penggunaan daya yang rendah. Hal ini karena memang ZigBee ditargetkan untuk digunakan pada benda yang menggunakan baterai, di mana *data rate* yang dibutuhkan tidak terlalu tinggi, dan biaya yang murah dan daya tahan baterai yang lama adalah kebutuhan utama. Baterai yang digunakan oleh perangkat ZigBee dapat bertahan beberapa tahun sampai perlu diganti. Hal ini dapat dicapai karena pada kebanyakan situasi, total waktu yang digunakan oleh suatu perangkat nirkabel dalam melakukan aktivitas sangat terbatas, sehingga perangkat terkait akan menghabiskan sebagian besar waktunya dalam kondisi hemat daya atau yang dikenal dengan *sleep mode*[6].

Kelebihan lain dari ZigBee adalah topologi jaringannya yang berupa *mesh network*. Ini berarti pengiriman data dari satu perangkat ke perangkat lain bisa tidak dilakukan secara langsung, namun melalui perangkat-perangkat lain sebagai perantara. ZigBee juga memiliki karakteristik *Self-Forming* dan *Self-Healing*. *Self-Forming* berarti sebuah jaringan ZigBee dapat langsung terbentuk ketika perangkat telah aktif tanpa diperlukan supervisi tambahan, sedangkan *Self-Healing* berarti jika

salah satu perangkat ZigBee yang berada dalam rute pengiriman data mati karena suatu hal, jaringan ZigBee akan secara otomatis membuat rute pengiriman data yang baru.

Dalam implementasinya, ZigBee memiliki beberapa profil. Profil-profil ini mendefinisikan bagaimana suatu perangkat harus bekerja. Dalam profil ini disebutkan apa saja fungsi-fungsi yang harus ada ketika akan dilakukan implementasi, dan apa saja atribut-atribut yang harus dimiliki oleh suatu perangkat. Dengan mengikuti profil ini, setiap perangkat yang dibuat oleh perusahaan berbeda akan tetap memiliki perilaku yang sama. Beberapa contoh profil yang telah ditentukan adalah:

- ZigBee Home Automation 1.2
- Light Link 1.0
- Building Automation 1.0
- Retail Services
- Remote Control 2.0
- Health Care 1.0

Namun pada saat tulisan ini dibuat, ZigBee Alliance sebagai pengembang ZigBee telah mengumumkan versi terbaru dari ZigBee, yaitu ZigBee 3.0. Pada ZigBee versi terbaru ini, dilakukan unifikasi dari berbagai profil ZigBee yang telah disebutkan sebelumnya, sehingga hanya ada satu standar ZigBee. Dengan dilakukannya unifikasi ini, diharapkan dapat memudahkan pengembang perangkat untuk mengembangkan perangkat berbasis ZigBee yang mendukung komunikasi antar perangkat dengan jenis berbeda.

2.4 *Gateway*

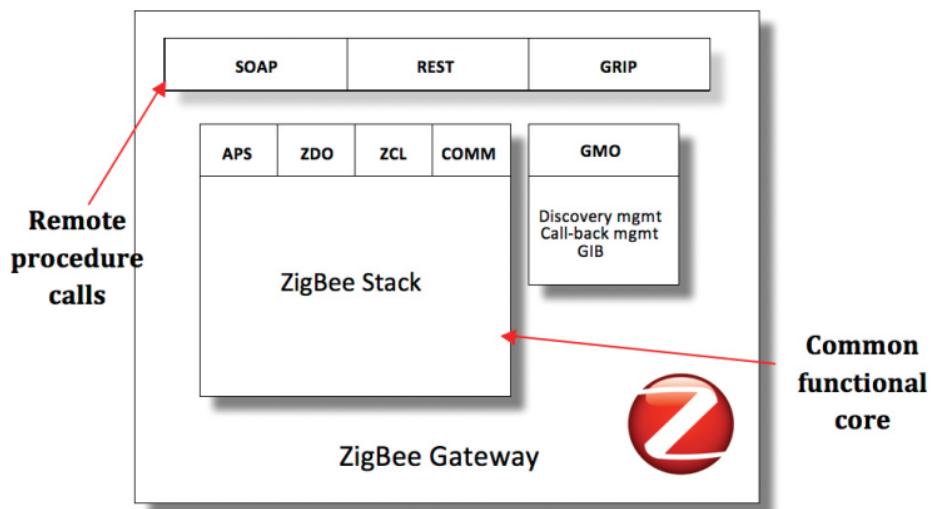
Gateway adalah sebuah entitas, baik berupa *hardware*, *software*, maupun gabungan keduanya, yang menyalurkan suatu data dari satu tempat ke tempat lainnya. *Gateway* bisa ada dalam banyak bentuk, seperti di antara beberapa LAN, diantara beberapa jenis jaringan yang berbeda, dan juga diantara beberapa aplikasi. *Gateway* biasanya beroperasi di atas *network layer* dan bisa melibatkan semua *layer*. Inti dari sebuah *gateway* adalah konversi atau pengubahan. Sebuah *gateway* akan mengubah pesan yang melewatinya hingga ke *layer* dimana *gateway* tersebut dioperasikan. Tipe *gateway* yang biasa digunakan adalah *application gateway* yang beroperasi di

application layer. Sebagai contoh, sebuah *gateway* pesan elektronik dapat mengubah pesan dari suatu format pesan elektronik ke format lainnya. [4].

2.4.1 ZigBee *Gateway*

Seperti dijelaskan sebelumnya, *gateway* berfungsi sebagai penyalur dan pengubah data yang dikirim dari satu tempat ke tempat lainnya. Berbagai macam *gateway* telah dikembangkan sesuai dengan kebutuhan. ZigBee Alliance sebagai pengembang dari protokol ZigBee juga telah mengeluarkan sebuah standar untuk membuat *gateway*. *Gateway* ini dapat digunakan untuk menghubungkan perangkat berbasis ZigBee dengan jaringan lain. Dalam dokumen yang dikeluarkan oleh ZigBee Alliance, dikatakan bahwa perangkat *gateway* ZigBee menyediakan jalur komunikasi menuju sebuah jaringan ZigBee menggunakan *IP based host application* (IPHA) dan sebaliknya dengan menyediakan suatu mekanisme dimana aplikasi eksternal bisa berinteraksi dengan suatu perangkat ZigBee untuk melakukan suatu kontrol atau mengambil suatu data [1].

ZigBee Alliance juga menyertakan desain arsitektur *gateway* dalam dokumen yang dikeluarkannya. Dalam desain arsitektur tersebut, sebuah *gateway* ZigBee dapat mengimplementasikan satu atau lebih protokol untuk melakukan *Remote Procedure Calls* (RPC). Protokol yang disarankan adalah SOAP, REST, dan GRIP. Pemilihan protokol tersebut dapat dilakukan sesuai kebutuhan pengembang [2]. Gambar 2.1 menunjukkan gambaran arsitektur *gateway* ZigBee yang disediakan oleh ZigBee Alliance.



Gambar 2.1: Arsitektur *Gateway* ZigBee[2]

Seperti terlihat pada gambar 2.1, implementasi *gateway* ZigBee dibagi menjadi dua lapis. Pada lapisan pertama, terdapat sekumpulan fungsi abstraksi yang tidak tergantung pada protokol dan digunakan untuk melakukan fungsi-fungsi terkait perangkat ZigBee dan pengaturan dari *gateway* itu sendiri. Pada lapisan kedua, terdapat sekumpulan protokol RPC yang dapat digunakan untuk berkomunikasi dengan pihak lain. Untuk pihak lain bisa berkomunikasi dengan *gateway*, akan tergantung pada protokol apa yang digunakan pada lapisan kedua ini. *Gateway* kemudian akan mengubah pesan yang diterima di lapisan kedua dan disampaikan kepada lapisan pertama untuk kemudian diproses.

2.5 MQTT

MQTT merupakan singkatan dari *Message Queueing Telemetry Transport*. MQTT adalah sebuah protokol pengiriman pesan yang sangat simpel dan ringan, dan di desain untuk perangkat yang memiliki keterbatasan dan *bandwidth* yang rendah. Hal ini membuat MQTT menjadi cocok untuk kebutuhan komunikasi antar mesin yang tidak memerlukan *bandwidth* yang tinggi namun perlu melakukan komunikasi yang cukup sering dan bisa diandalkan [13].

MQTT di desain agar segala kompleksitas mekanisme pengiriman pesan terdapat di sisi *broker*, sehingga dalam menggunakan *client* MQTT dapat dilakukan dengan mudah. MQTT menggunakan konsep *publish-subscribe* dalam melakukan kerjanya. Model *publish-subscribe* yang digunakan adalah berdasarkan topik. Dengan menggunakan topik yang disusun menjadi sebuah hirarki, dapat dimungkinkan untuk suatu perangkat melakukan langganan lebih dari satu topik sekaligus [9].

2.5.1 Sistem *Publish-Subscribe*

Konsep sistem *publish-subscribe* (pub-sub) adalah suatu model komunikasi di mana pihak yang tertarik untuk mendapatkan informasi mengenai suatu hal, dapat mendaftarkan ketertarikannya. Proses pendaftaran ketertarikan ini disebut proses *subscription*, dan oleh karena itu pihak yang melakukan proses ini disebut *subscriber*. Pihak yang akan memberikan informasi kemudian melakukan suatu proses yang disebut *publishing*, sehingga pihak yang melakukan proses ini disebut sebagai *publisher*. Selain kedua pihak tersebut, perlu ada suat pihak yang berperan menjadi penengah, dan bertanggung jawab untuk mengirimkan dan memastikan informasi yang dikirim oleh *publisher* dapat sampai ke *subscriber*. Pihak yang melakukan pekerjaan ini disebut sebagai *broker* [9].

Ada tiga tipe dasar dalam konsep sistem pub-sub ini, yaitu *topic-based* (berdasarkan topik), *type-based* (berdasarkan tipe), dan *content-based* (berdasarkan isi pesan). Pada tipe yang berdasarkan topik, proses *subscription* dan *publishing* hanya dapat dilakukan untuk suatu topik tertentu yang biasanya sudah diketahui terlebih dahulu ketika pengembangan sistem atau aplikasi. Untuk tipe *type-based*, pihak *subscriber* akan memberikan informasi tentang tipe data yang ingin ia *subscribe*, seperti misalnya data penggunaan listrik. Sedangkan untuk tipe yang terakhir yaitu *content-based*, subscriber akan melakukan proses *subscriber* dengan menjelaskan isi pesan seperti apakah yang ingin diterimanya [9].

2.5.2 Topik dalam MQTT

Topik dalam MQTT adalah sebuah *string* yang digunakan oleh *broker* MQTT untuk melakukan penyaringan pesan yang akan dikirim kepada *subscriber*. Topik dalam MQTT disusun menjadi suatu hirarki yang terbagi menjadi beberapa level. Setiap level dipisahkan dengan tanda *forward slash* (/). Setiap topik harus memuat minimal satu karakter dan bersifat *case sensitive* [8]. Gambar 2.2 menunjukkan contoh topik dalam MQTT.



Gambar 2.2: Contoh topik dalam MQTT [8]

Dalam membuat topik MQTT, kita juga dapat menggunakan *wildcard*. Terdapat dua macam *wildcard* dalam MQTT, yaitu *single level wildcard* yang di tandai dengan karakter tanda tambah (+) dan *multi level wildcard* yang ditandai dengan karakter tanda pagar (#). *Single level wildcard* dapat digunakan untuk merepresentasikan suatu topik dalam satu level. Gambar 2.3 menunjukkan contoh penggunaan *single level wildcard*. Sedangkan *multi level wildcard* dapat merepresentasikan lebih dari satu level topik, dan oleh karena itu karakter *multi level wildcard* hanya dapat digunakan sebagai karakter terakhir dalam suatu *string* topik dan harus di-dahului oleh karakter *forward slash* (/). Namun demikian, kita dapat menggunakan *multi level wildcard* sebagai satu-satunya karakter dalam *string* topik yang berarti *subscriber* akan menerima semua pesan yang dikirimkan ke *broker* [8]. Gambar 2.4 menunjukkan contoh penggunaan *multi level wildcard*.



Contoh topik yang akan diterima pesannya :

- ✓ siot/userA/kamarTidur/lampu
- ✓ siot/userA/Dapur/lampu
- ✗ siot/userA/Dapur/kulkas
- ✗ siot/userB/Dapur/lampu

Gambar 2.3: Contoh *single level wildcard* [8]



Contoh topik yang akan diterima pesannya :

- ✓ siot/userA/kamarTidur/lampu
- ✓ siot/userA/Rumah/Sensor/Temperatur
- ✓ siot/userA/sensor
- ✗ siot/userB/Dapur/lampu

Gambar 2.4: Contoh *multi level wildcard* [8]

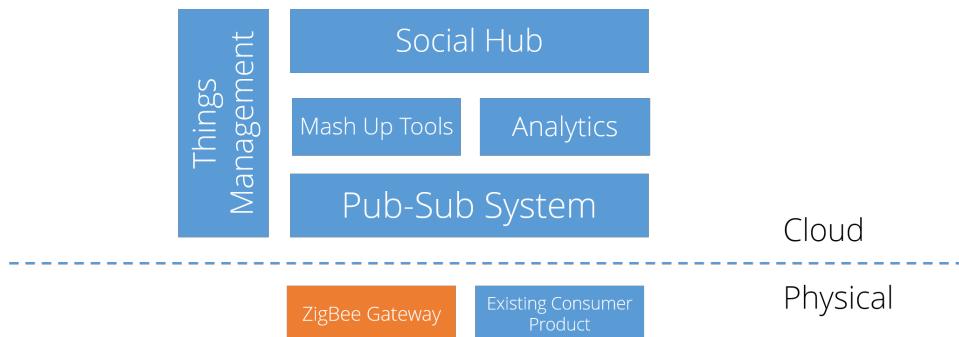
BAB 3

RANCANGAN IMPLEMENTASI

Bab ini menjelaskan rancangan sistem yang akan dibuat. Rancangan tersebut mencakup gambaran umum *platform* yang akan digunakan, gambaran umum *gateway* beserta rancangan implementasinya, dan rancangan pengujian.

3.1 Gambaran Umum Sistem

Dalam tugas akhir ini, akan diimplementasikan sebuah *gateway* yang akan menghubungkan jaringan lampu berbasis ZigBee dengan sebuah *platform internet of things* berbasis media sosial. Pada *platform* ini, seorang pengguna dapat menghubungkan berbagai perangkat yang dimilikinya ke internet, dan kemudian memberikan informasi terkait perangkat tersebut ke teman-temannya di media sosial. Selain membagikan informasi terkait perangkat, pengguna juga dapat membagikan akses kontrol suatu perangkat ke internet atau teman-temannya di media sosial. Gambaran keseluruhan sistem ini dapat dilihat pada gambar 3.1.



Gambar 3.1: Gambaran Umum *Platform* Internet of Things berbasis media sosial

Sistem ini terdiri dari beberapa bagian yang saling berhubungan. Sebagian besar sistem ini berada di *server* yang terhubung dengan jaringan internet. Penjelasan lebih detail mengenai bagian-bagian tersebut adalah sebagai berikut:

1. *Social Hub*

Bagian ini merupakan abstraksi teratas dalam keseluruhan sistem. Bagian ini sekaligus menjadi *user interface*, dimana pengguna *platform* nantinya akan mengakses sistem ini. Selain menjadi tempat pengguna mendaftar, bagian ini juga memiliki fungsi yang sangat penting, yaitu mengatur bagaimana

mekanisme pembagian hak akses dan hak kontrol diantara seorang pengguna dengan pengguna lainnya. Sebagai contoh, pengguna dapat memilih apakah hak akses informasi mengenai sebuah perangkat yang dimilikinya akan dibagikan ke seluruh pengguna internet, hanya teman-temannya saja, atau hanya orang-orang tertentu saja.

2. *Mash Up Tools*

Mash up tools merupakan salah satu cara bagi pengguna *platform* untuk berinteraksi dengan suatu perangkat. Melalui *mash up tools*, pengguna dapat melakukan otomasi terhadap sebuah perangkat jika memenuhi kondisi tertentu. Misalnya, seorang pengguna dapat mengatur agar lampu ruang keluarga yang dimilikinya akan otomatis menyala jika pengguna mendapat email dengan subjek tertentu.

3. *Analytics*

Dengan banyaknya perangkat yang nantinya akan terhubung ke *platform* ini, akan sangat banyak pula data yang melewati sistem ini. Data-data tersebut jika dilihat secara terpisah mungkin tidak memiliki nilai yang berarti. Namun, jika data seperti ini terdapat dalam jumlah yang sangat banyak, kita mungkin bisa mendapatkan wawasan tentang suatu hal yang terkait dengan data tersebut. Oleh karena itu, diperlukan suatu bagian yang dapat mengolah data yang banyak ini sehingga dapat memberikan suatu informasi yang berarti. Inilah peran dari bagian *analytics*.

4. *Pub-Sub System*

Seperti yang telah disebutkan sebelumnya, seiring dengan semakin banyaknya pengguna yang terdaftar dan perangkat yang terhubung, akan semakin banyak pula data yang akan melewati sistem ini. Data-data ini harus dikirimkan dari satu perangkat ke setiap pengguna yang berhak menerimanya, dan juga dari pengguna ke setiap perangkat yang harus menerimanya. Oleh karena itu, diperlukan suatu mekanisme pengaturan pengiriman pesan, yang dapat mengirimkan setiap data tersebut ke tujuan yang sesuai dan tetap menjaga kualitas kinerja dari sistem. Hal terkait mekanisme pengiriman pesan inilah yang ditangani oleh *pub-sub system*.

5. *Things Management*

Things management adalah bagian yang mengatur hubungan antara sistem ini dengan setiap perangkat yang terhubung. *Things management* mengatur mulai dari proses pendaftaran perangkat, hingga penyimpanan informasi

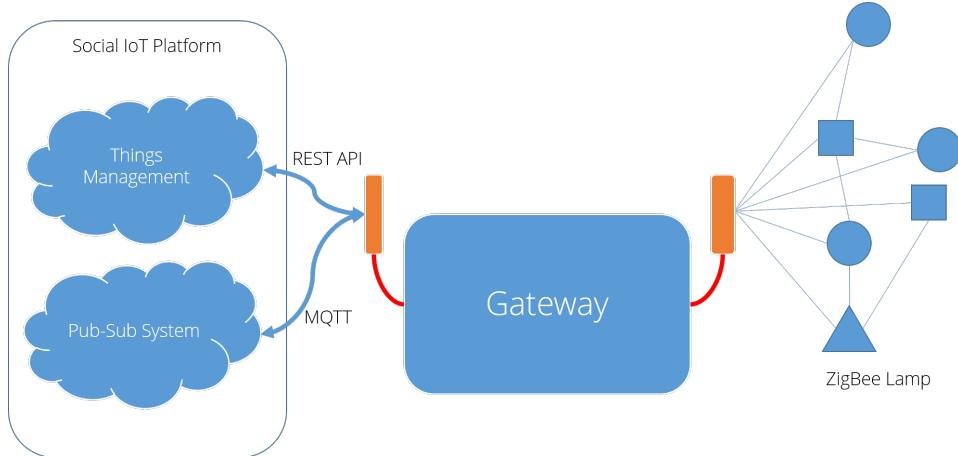
terkait perangkat. Things management ini juga yang mengatur bagaimana mekanisme komunikasi antara *platform* dengan perangkat. Perangkat yang terhubung dengan bagian ini terbagi menjadi dua macam. Yang pertama adalah perangkat yang sudah tersedia di pasaran saat ini. Untuk perangkat jenis ini, *things management* akan menggunakan API perangkat terkait sebagai jalur komunikasi dengan perangkat tersebut. Dengan demikian, perangkat yang sudah dimiliki pengguna dapat langsung dihubungkan dengan *platform* ini. Jenis kedua adalah perangkat yang akan dikembangkan di masa depan. Untuk perangkat jenis ini, *things management* akan menyediakan sebuah API yang dapat digunakan oleh pengembang perangkat cerdas, sehingga perangkat yang dikembangkannya dapat terhubung dengan *platform* ini.

6. *Gateway*

Bagian yang terakhir adalah *gateway*. Bagian ini berfungsi menghubungkan suatu perangkat dengan internet, khususnya *platform* ini. *Gateway* inilah yang akan penulis implementasikan dalam tugas akhir ini. Dalam sistem ini, *gateway* berhubungan dengan *things management* sebagai perangkat jenis kedua. Oleh karena itu, dalam melakukan implementasi *gateway* ini penulis akan mengacu pada API yang disediakan oleh *things management*. Meskipun di masa mendatang diharapkan *gateway* bisa menghubungkan berbagai jenis perangkat, namun untuk implementasi saat ini penulis akan fokus pada perangkat lampu yang menggunakan teknologi ZigBee. Selain lampu, *gateway* ini juga bisa mengontrol sebuah kumpulan lampu, atau biasa disebut sebagai *group*. Karena atribut yang dimiliki oleh lampu dan *group* sama, maka dalam tugas akhir ini sebuah *group* akan diibaratkan sebagai sebuah lampu.

3.1.1 Gambaran Umum Perangkat *Gateway*

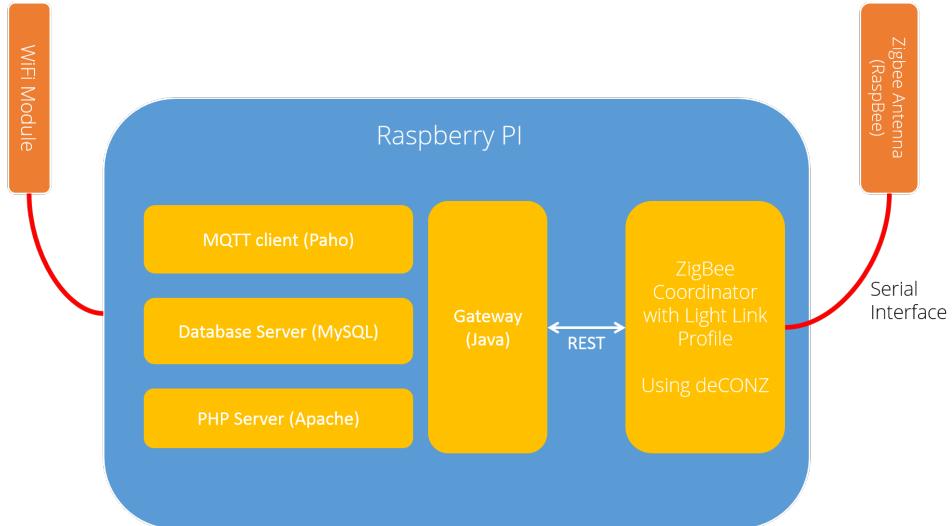
Dalam tugas akhir ini diimplementasikan sebuah perangkat *gateway* yang menghubungkan jaringan lampu berbasis ZigBee dengan internet. Jaringan lampu ini secara spesifik akan dihubungkan dengan *platform internet of things* berbasis sosial media dengan deskripsi seperti yang telah dijelaskan sebelumnya. Pertama kali, lampu akan didaftarkan ke bagian *things management* pada *platform*, kemudian setelah terdaftar, *gateway* akan secara periodik mengirimkan informasi mengenai lampu ke *pub-sub system* dan menerima perintah dari *pub-sub system*. Gambar 3.2 menunjukkan visualisasi hubungan perangkat *gateway* dengan *platform* dan lampu.



Gambar 3.2: Gambaran Umum Sistem

Sistem ini terdiri dari jaringan lampu ZigBee, sebuah perangkat *gateway*, dan *platform internet of things* yang berbasis sosial media. Di dalam *platform internet of things* terdapat *things management* yang bertugas sebagai portal manajemen perangkat, termasuk mendaftarkan dan menyimpan informasi terkait perangkat, dan sebuah *pub-sub system* yang bertugas untuk menyampaikan data yang dikirimkan ke dan dari *gateway*. *Gateway* akan menarik informasi mengenai keadaan lampu melalui jaringan ZigBee, kemudian jika lampu belum didaftarkan ke *platform*, maka pengguna dapat mendaftarkan lampu tersebut. Proses pendaftaran lampu ini menggunakan REST melalui API yang disediakan. Setelah didaftarkan, *gateway* kemudian akan mengirimkan informasi mengenai lampu terkait secara periodik ke *platform*. Informasi ini akan dikirimkan melalui protokol MQTT ke *pub-sub system* yang terdapat pada *platform*. Implementasi *pub-sub system* pada *platform* menggunakan Mosquitto. Selain itu, *gateway* juga dapat digunakan sebagai alat untuk mengontrol lampu-lampu yang terdapat dalam jaringan ZigBee.

Gambar 3.3 menunjukkan visualisasi yang lebih mendetail pada sistem internal perangkat *gateway*. Untuk bisa terhubung dengan jaringan ZigBee, dibutuhkan sebuah *coordinator*. Pada tugas akhir ini, implementasi ZigBee *coordinator* yang akan digunakan adalah implementasi yang dibuat oleh pihak Dresden Elektronik dengan nama deCONZ. deCONZ mengimplementasikan profile *light link* yang disediakan oleh pihak ZigBee Alliance, sehingga dapat digunakan untuk menghubungkan lampu ZigBee yang menggunakan profil yang sama. Perangkat yang digunakan untuk terhubung ke lampu ZigBee adalah sebuah *shield* Raspberry Pi bernama RaspBee yang juga dibuat oleh Dresden Elektronik dan dapat dihubungkan ke *port GPIO* yang terdapat pada Raspberry Pi. Secara *default*, deCONZ dapat berkomunikasi dengan RaspBee tanpa perlu dilakukan banyak konfigurasi.



Gambar 3.3: Gambaran Umum Gateway

Gateway pada tugas akhir ini akan bertugas mengirimkan informasi mengenai lampu ZigBee yang terhubung secara berkala ke *platform*, dan juga menerima perintah yang datang dari *platform* untuk kemudian disampaikan ke lampu terkait. Implementasi *gateway* dikerjakan menggunakan bahasa pemrograman Java. Ketika dijalankan, *gateway* akan membuat sebuah MQTT *client* yang diimplementasikan menggunakan *library* Eclipse Paho. MQTT *client* inilah yang akan digunakan untuk menerima perintah dari *platform* dan secara berkala mengirim informasi ke *platform*. Perangkat ini juga akan menyediakan sebuah aplikasi web yang dapat digunakan untuk mengontrol dan mengelola lampu yang terhubung. Melalui aplikasi web ini, pengguna dapat mendaftarkan lampu yang dimilikinya ke *things management* yang terdapat pada *platform*. Informasi terkait hasil pendaftaran ini akan disimpan pada sebuah *database server* dan akan digunakan selanjutnya oleh *gateway* untuk menentukan beberapa hal, seperti *id* lampu yang tersimpan di *things management*, perintah apa saja yang akan diterima oleh *gateway*, dan informasi apa saja yang akan dibagikan oleh *gateway*. Pada perangkat ini juga akan dipasang sebuah modul WiFi yang dapat digunakan untuk menghubungkan perangkat *wifi* ke internet dan untuk mengakses aplikasi web pada perangkat *gateway*.

3.2 Rancangan Implementasi *Gateway*

Pada subbab ini akan dijelaskan secara mendetail mengenai rancangan perangkat *gateway* yang dibuat. Penjelasan pada subbab ini mencakup perangkat dan *tools* yang digunakan, mekanisme pengiriman pesan beserta topik yang digunakan, dan rancangan cara penggunaan perangkat *gateway*.

3.2.1 Perangkat yang digunakan

Perangkat yang akan digunakan pada pengerjaan tugas akhir ini adalah Raspberry Pi. Raspberry Pi adalah sebuah komputer berukuran kecil, yang dikembangkan oleh yayasan Raspberry Pi. Walaupun berukuran kecil, Raspberry Pi merupakan sebuah komputer yang dapat berfungsi penuh seperti komputer biasa. OS yang tersedia untuk Raspberry Pi cukup banyak dan dapat diunduh langsung dari situs resmi Raspberry Pi. Raspberry Pi dipilih untuk pengerjaan tugas akhir ini karena memiliki kemampuan yang cukup untuk menjalankan berbagai kebutuhan pada implementasi perangkat *gateway* ini. Selain itu, Raspberry Pi memiliki harga yang cukup murah, hanya sekitar enam ratus ribu rupiah untuk tipe terbaru.

Untuk bisa terhubung dengan sebuah jaringan ZigBee, diperlukan juga sebuah perangkat yang memiliki kemampuan untuk menangkap sinyal ZigBee dan terhubung dengan jaringan ZigBee. Perangkat ini juga harus bisa digunakan pada perangkat Raspberry Pi. Perangkat yang dipilih untuk implementasi pada tugas akhir ini adalah RaspBee, sebuah *shield* untuk Raspberry Pi yang diproduksi oleh Dresden Elektronik. RaspBee dipilih untuk pengerjaan tugas akhir ini karena memiliki *tools* yang dapat digunakan untuk menghubungkan jaringan ZigBee dengan mudah, dan *tools* tersebut dapat dijalankan pada Raspberry Pi.

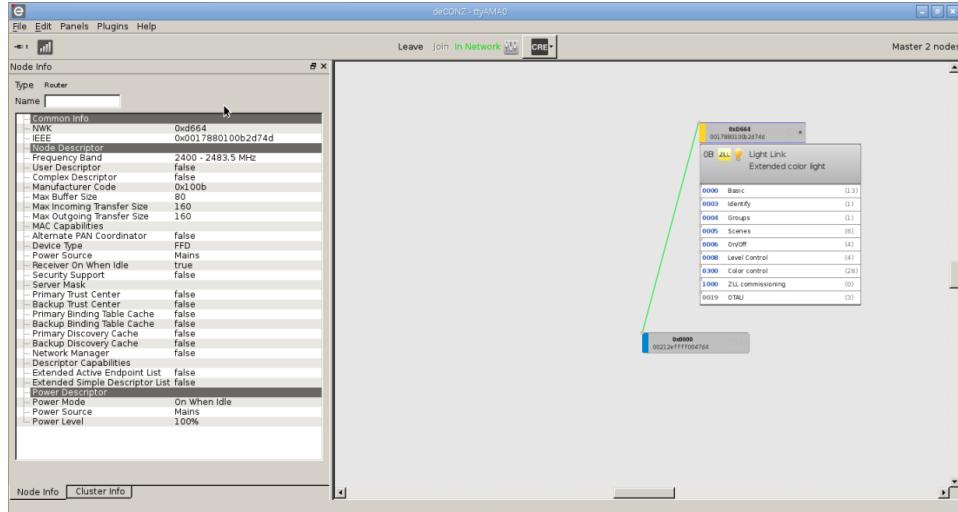
3.2.2 Tools yang digunakan

Implementasi perangkat *gateway* pada tugas akhir ini menggunakan beberapa *tools* yang sudah tersedia. *Tools* yang digunakan adalah deCONZ, Mosquitto, dan Eclipse Paho.

3.2.2.1 deCONZ

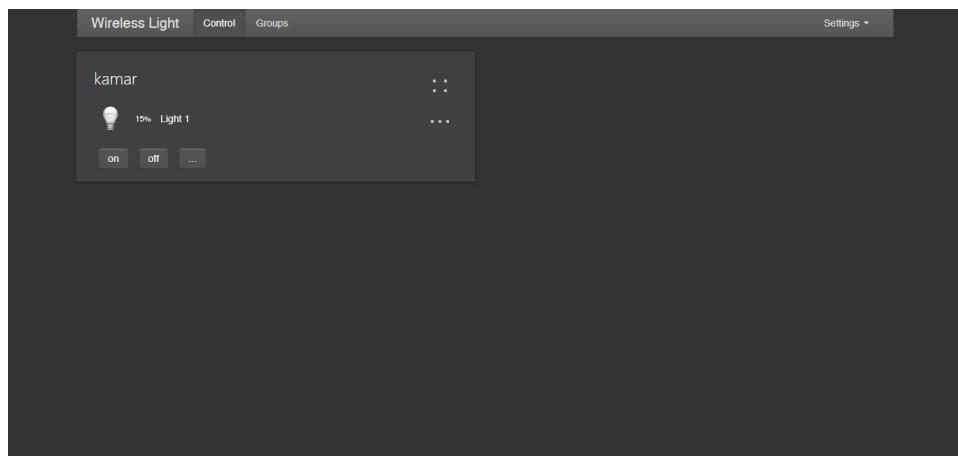
deCONZ merupakan paket aplikasi yang dibuat dan disebarluaskan secara gratis oleh Dresden Elektronik. Aplikasi ini dapat digunakan untuk menghubungkan beberapa perangkat yang diproduksi oleh Dresden Elektronik dengan jaringan perangkat berbasis ZigBee.

Terdapat tiga cara untuk menggunakan deCONZ. Yang pertama adalah dengan menggunakan aplikasi GUI yang termasuk dalam satu set aplikasi deCONZ. Untuk menggunakan aplikasi GUI ini, Raspberry Pi harus dihubungkan ke layar eksternal dan OS dalam Raspberry Pi harus menjalankan sebuah X server atau tampilan *desktop*. deCONZ dalam mode GUI ini bisa terhubung dengan semua perangkat berbasis ZigBee. Gambar 3.4 menunjukkan tampilan deCONZ ketika dijalankan dalam mode GUI



Gambar 3.4: Tampilan Aplikasi deCONZ berbasis GUI

Cara kedua adalah menggunakan aplikasi web yang juga sudah termasuk dalam set aplikasi deCONZ. Ketika dijalankan, aplikasi deCONZ akan sekaligus menjalankan sebuah *webserver* dimana aplikasi web deCONZ dijalankan. Untuk mengakses aplikasi web ini, kita dapat menggunakan browser dari desktop di Raspberry Pi , atau dari perangkat di luar Raspberry Pi menggunakan koneksi jaringan seperti Ethernet atau WiFi. Jika mengakses dari luar Raspberry Pi , maka kita perlu mengetahui alamat IP dari Raspberry Pi ketika sedang digunakan. Aplikasi deCONZ yang berbasis web ini hanya dapat digunakan untuk mengontrol perangkat berbasis ZigBee yang menggunakan profil *Light Link* atau *Home Automation* dengan jenis lampu. Gambar 3.5 menunjukkan tampilan aplikasi deCONZ berbasis web.



Gambar 3.5: Tampilan Aplikasi deCONZ berbasis web

Cara terakhir adalah menggunakan REST API. Sama seperti sebelumnya, ketika dijalankan deCONZ juga akan sekaligus menjalankan sebuah *webserver*, yang juga

menjadi tempat di mana REST API ini berjalan. Untuk bisa mengakses REST API ini, kita bisa menggunakan koneksi jaringan dari luar Raspberry Pi atau mengaksesnya langsung dari dalam Raspberry Pi. Untuk implementasi *gateway* pada tugas akhir ini, cara yang akan digunakan adalah melalui REST API.

3.2.2.2 Mosquitto

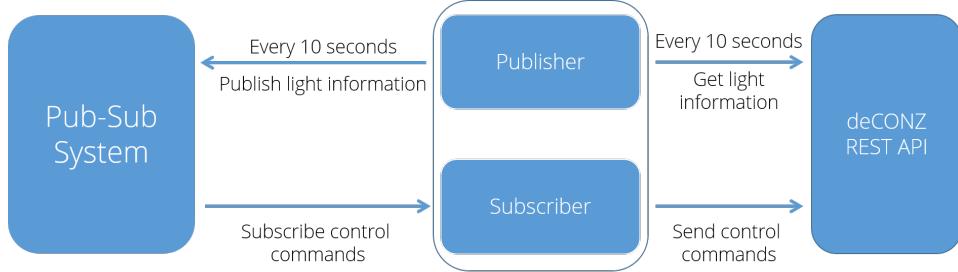
Mosquitto adalah sebuah broker *open source* yang mengimplementasikan protokol MQTT versi 3.1 dan 3.1.1. Mosquitto dapat diunduh melalui situs resminya, dan tersedia untuk banyak *platform*, mulai dari Windows, Mac, beberapa distro linux termasuk untuk Raspberry Pi, hingga iOS. Mosquitto juga sudah menyediakan *client* untuk melakukan proses *publishing* dan *subscribing* melalui *command line*. Pada tugas akhir ini, Mosquitto digunakan sebagai broker yang dipasang pada *pub-sub system* di *platform internet of things* yang dikembangkan oleh penulis dan teman-teman penulis.

3.2.2.3 Eclipse Paho

Paho merupakan sebuah implementasi *client MQTT* yang dikembangkan oleh Eclipse dan bersifat *open source*. Implementasi ini dibungkus menjadi suatu *library* yang dapat digunakan di beberapa bahasa dan dapat diunduh melalui situs resminya. Beberapa bahasa yang didukung oleh *library* Eclipse Paho saat ini adalah Java, Phyton, Javascript, C/C++, dan .Net. Pada tugas akhir ini, *library* yang digunakan adalah Eclipse Paho untuk bahasa Java, dan akan digunakan sebagai *client* yang akan berjalan pada perangkat *gateway*.

3.2.3 Rancangan Mekanisme Pengiriman Pesan

Ketika dijalankan, *gateway* akan membuat dua buah *client MQTT* menggunakan Eclipse paho. *Client* yang pertama akan berperan sebagai *publisher* dan *client* yang kedua sebagai *subscriber*. Setiap sepuluh detik, *gateway* akan mengambil informasi mengenai lampu yang terhubung ke *coordinator*, dan mengirimkan informasi tersebut ke *pub-sub system* yang terdapat pada *platform* melalui mekanisme *publishing*. *Subscriber* pada *gateway* akan selalu menunggu pesan yang datang dari *pub-sub system*. Jika pesan yang diterima berupa perintah, maka perintah tersebut akan diubah menjadi perintah REST yang akan dikirimkan ke deCONZ. Visualisasi mekanisme ini ditampilkan pada gambar 3.6.



Gambar 3.6: Rancangan Mekanisme Pengiriman Pesan

3.2.4 Rancangan Topik

Topik yang digunakan pada *platform internet of things* ini dibuat agar menjadi sebuah topik yang bersifat *generic*, sehingga topik ini tetap bisa digunakan apapun jenis perangkat yang nantinya akan terhubung. Pertama, untuk menandai bahwa topik ini digunakan untuk *platform* yang penulis dan teman-teman penulis buat, topik ini diawali dengan `sot/`. Selanjutnya, karena *platform* yang digunakan akan mengelola dua jenis perangkat yang berbeda (lihat subbab 3.1 bagian *things management*), maka selanjutnya ditambahkan `/d` untuk menandakan perangkat jenis pertama atau `/g` untuk menandakan atau perangkat jenis kedua. Karena perangkat *gateway* merupakan perangkat jenis kedua, maka akan digunakan `/g` dalam topik pada tugas akhir ini. Topik ini juga harus bisa mengatasi penggunaan oleh banyak pengguna, sehingga harus ada informasi mengenai identitas pengguna di dalam topik. Oleh karena itu, ditambahkan `/[idUser]` ke dalam topik.

Selanjutnya, *platform* juga memiliki kebutuhan untuk mengetahui jenis perangkat dalam suatu topik, maka ditambahkan `/[kategori]` ke dalam topik. Karena dalam tugas akhir ini perangkat yang digunakan hanya lampu, maka kategori yang digunakan adalah `ledlight`. Kemudian, *gateway* juga perlu mengetahui informasi tentang perangkat mana yang menjadi tujuan atau pengirim informasi di dalam topik, maka ditambahkan `[idPerangkat]` ke dalam topik. Terakhir, terdapat dua buah operasi terkait sebuah atribut dalam suatu perangkat yaitu untuk mengontrol, atau memberikan informasi mengenai perangkat. Oleh karena itu, ditambahkan `/[atribut]/(acc atau ctl)` pada bagian akhir dari topik. `acc` menandakan bahwa topik tersebut mengirimkan pesan yang berisi informasi mengenai perangkat ke *platform*, sedangkan `ctl` menandakan bahwa pesan yang dikirimkan berisi perintah yang ditujukan ke perangkat. Jika digabungkan, topik yang akan digunakan pada tugas akhir ini menjadi `sot/g/[idUser]/ledlight/[idPerangkat]/[atribut]/(acc atau ctl)`.

3.2.5 Rancangan Translasi Pesan MQTT Menjadi Perintah REST

@todo
name

Ketika *gateway* menerima pesan untuk mengontrol suatu lampu, *gateway* perlu menerjemahkan pesan tersebut menjadi sebuah perintah REST yang kemudian disampaikan ke deCONZ. Informasi yang dibutuhkan untuk mengontrol lampu atau *group* terdapat di dalam topik dan juga di dalam pesan yang dikirimkan. Selain itu, terdapat perbedaan id yang digunakan dalam *things management* di *platform* dan id yang digunakan deCONZ untuk menghubungi lampu, sehingga *gateway* juga perlu menyimpan informasi mengenai kedua id pada masing-masing lampu. Dalam bagian ini, id lampu yang disimpan pada *things management* akan dituliskan dengan *idPerangkat*, dan id lampu yang digunakan oleh deCONZ akan dituliskan dengan *idLokal*. Selain itu, untuk bisa mengakses API REST yang disediakan oleh deCONZ, diperlukan suatu kunci API. Pada tugas akhir ini, kunci yang digunakan adalah LUQMANGWTA. Berikut merupakan contoh pesan-pesan yang akan diterima oleh *gateway* dan akan disampaikan ke deCONZ.

1. Menghidupkan lampu

Berikut merupakan contoh topik dan pesan yang dikirimkan untuk menghidupkan lampu.

Tabel 3.1: Contoh pesan dan topik untuk menghidupkan lampu

Topik	sot/g/[idUser]/ledlight/[idPerangkat]/on/ctl
Isi pesan	true
Perintah REST	PUT http://localhost:8080/api/LUQMANGWTA/lights/[idLokal]/state
Data yang dikirimkan	{"on":true}
Respon deCONZ	[{"success": {"/lights/[idLokal]/state/on": true}}]

2. Mematikan Lampu

Berikut merupakan contoh topik dan pesan yang dikirimkan untuk mematikan lampu.

Tabel 3.2: Contoh pesan dan topik untuk mematikan lampu

Topik	sot/g/[idUser]/ledlight/[idPerangkat]/on/ctl
Isi pesan	false
Perintah REST	PUT http://localhost:8080/api/LUQMANGWTA/lights/[idLokal]/state
Data yang dikirimkan	{"on":false}
Respon deCONZ	[{"success": {"/lights/[idLokal]/state/on": false}}]

3. Mengubah Tingkat Kecerahan Lampu (*Brightness*)

Berikut merupakan contoh topik dan pesan yang dikirimkan untuk mengubah tingkat kecerahan lampu.

Tabel 3.3: Contoh pesan dan topik untuk mengubah tingkat kecerahan lampu

Topik	sot/g/[idUser]/ledlight/[idPerangkat]/bri/ctl
Isi pesan	100
Perintah REST	PUT http://localhost:8080/api/LUQMANGWTA/lights/[idLokal]/state
Data yang dikirimkan	{"bri":100}
Respon deCONZ	[{"success": {"/lights/[idLokal]/state/bri": 100}}]

4. Mengubah Warna Lampu (*Hue*)

Berikut merupakan contoh topik dan pesan yang dikirimkan untuk mengubah warna lampu.

Tabel 3.4: Contoh pesan dan topik untuk mengubah warna lampu

Topik	sot/g/[idUser]/ledlight/[idPerangkat]/hue/ctl
Isi pesan	45000
Perintah REST	PUT http://localhost:8080/api/LUQMANGWTA/lights/[idLokal]/state
Data yang dikirimkan	{"hue":45000}
Respon deCONZ	[{"success": {"/lights/[idLokal]/state/hue": 45000}}]

5. Mengubah Tingkat Kejemuhan Lampu (*Saturation*)

Berikut merupakan contoh topik dan pesan yang dikirimkan untuk mengubah warna lampu.

Tabel 3.5: Contoh pesan dan topik untuk mengubah tingkat kejemuhan lampu

Topik	sot/g/[idUser]/ledlight/[idPerangkat]/sat/ctl
Isi pesan	150
Perintah REST	PUT http://localhost:8080/api/LUQMANGWTA/lights/[idLokal]/state
Data yang dikirimkan	{"sat":246}
Respon deCONZ	[{"success": {"/lights/[idLokal]/state/sat":246}}]

6. Menghidupkan lampu di dalam suatu *group*

Berikut merupakan contoh topik dan pesan yang dikirimkan untuk menghidupkan lampu di dalam suatu *group*.

Tabel 3.6: Contoh pesan dan topik untuk menghidupkan lampu di dalam suatu *group*

Topik	sot/g/[idUser]/ledlight/[idPerangkat]/on/ctl
Isi pesan	true
Perintah REST	PUT http://localhost:8080/api/LUQMANGWTA/groups/[idLokal]/action
Data yang dikirimkan	{"on":true}
Respon deCONZ	[{"success": {"/groups/[idLokal]/action/on":true}}]

7. Mematikan Lampu di dalam suatu *group*

Berikut merupakan contoh topik dan pesan yang dikirimkan untuk mematikan lampu di dalam suatu *group*.

Tabel 3.7: Contoh pesan dan topik untuk mematikan lampu di dalam suatu *group*

Topik	sot/g/[idUser]/ledlight/[idPerangkat]/on/ctl
Isi pesan	false
Perintah REST	PUT http://localhost:8080/api/LUQMANGWTA/groups/[idLokal]/action
Data yang dikirimkan	{"on":false}
Respon deCONZ	[{"success": {"/groups/[idLokal]/action/on":false}}]

8. Mengubah Tingkat Kecerahan Lampu (*Brightness*) di dalam suatu *group*

Berikut merupakan contoh topik dan pesan yang dikirimkan untuk mengubah tingkat kecerahan lampu di dalam suatu *group*.

Tabel 3.8: Contoh pesan dan topik untuk mengubah tingkat kecerahan lampu di dalam suatu *group*

Topik	sot/g/[idUser]/ledlight/[idPerangkat]/bri/ctl
Isi pesan	100
Perintah REST	PUT http://localhost:8080/api/LUQMANGWTA/groups/[idLokal]/action
Data yang dikirimkan	{"bri":100}
Respon deCONZ	[{"success":{"/groups/[idLokal]/action/bri":100}}]

9. Mengubah Warna Lampu (*Hue*) di dalam suatu *group*

Berikut merupakan contoh topik dan pesan yang dikirimkan untuk mengubah warna lampu di dalam suatu *group*.

Tabel 3.9: Contoh pesan dan topik untuk mengubah warna lampu di dalam suatu *group*

Topik	sot/g/[idUser]/ledlight/[idPerangkat]/hue/ctl
Isi pesan	45000
Perintah REST	PUT http://localhost:8080/api/LUQMANGWTA/groups/[idLokal]/action
Data yang dikirimkan	{"hue":45000}
Respon deCONZ	[{"success":{"/groups/[idLokal]/action/hue":45000}}]

10. Mengubah Tingkat Kejemuhan Lampu (*Saturation*) di dalam suatu *group*

Berikut merupakan contoh topik dan pesan yang dikirimkan untuk mengubah warna lampu di dalam suatu *group*.

Tabel 3.10: Contoh pesan dan topik untuk mengubah tingkat kejemuhan lampu di dalam suatu *group*

Topik	sot/g/[idUser]/ledlight/[idPerangkat]/sat/ctl
Isi pesan	45000
Perintah REST	PUT http://localhost:8080/api/LUQMANGWTA/groups/[idLokal]/action
Data yang dikirimkan	{"sat":246}
Respon deCONZ	[{"success":{"/groups/[idLokal]/action/sat":246}}]

3.2.6 Rancangan Mekanisme Penggunaan Oleh Pengguna

Perangkat *gateway* yang dibuat dalam tugas akhir ini ditujukan agar tetap bisa digunakan tanpa harus terhubung ke *platform* yang dikerjakan oleh penulis dan teman-teman penulis. Oleh karena itu, dibutuhkan suatu cara untuk berinteraksi dengan

pengguna yang tidak terlalu rumit. Selain itu, diperlukan juga cara mendaftarkan perangkat yang terhubung ke perangkat *gateway* ke *things management* pada *platform*.

3.2.6.1 Cara Pengguna Terhubung ke Perangkat *Gateway*

Ketika perangkat *gateway* dinyalakan, perangkat *gateway* akan berperan sebagai WiFi *access point*. Pengguna kemudian dapat menggunakan komputer atau *smartphone* yang dimilikinya untuk terhubung ke *access point* yang dibuat oleh perangkat *gateway*. Ketika pengguna terhubung ke perangkat *gateway*, pengguna dapat membuka alamat 192.168.42.1/ta menggunakan *browser* yang dimiliki pengguna. Pada halaman itu, pengguna kemudian dapat memilih menu 'Pilih Access Point', dan pengguna akan diarahkan ke halaman yang menampilkan seluruh *access point* yang berada dalam jangkauan perangkat *gateway*. Pengguna kemudian dapat memilih *access point* yang memiliki koneksi internet dan menghubungkan perangkat *gateway* ke *access point tersebut*. Pengguna kemudian akan diminta untuk menghubungkan *smartphone* atau komputer yang digunakannya untuk dihubungkan ke *access point* yang sama. Halaman tersebut juga akan memberikan sebuah tautan beserta alamat yang perlu dibuka untuk bisa kembali mengakses halaman aplikasi web perangkat *gateway*.

3.2.6.2 Cara Melakukan *Log In* Melalui Perangkat *Gateway*

Untuk bisa mendaftarkan lampu atau *group* yang terhubung ke perangkat *gateway* ke *platform*, pengguna perlu melakukan *log in* terlebih dahulu. Untuk melakukan *log in*, pengguna harus sudah mendaftar terlebih dahulu di *platform*. Setelah mendaftar, pengguna cukup melakukan klik pada tombol *log in*, dan kemudian memasukkan *email* yang sudah didaftarkan di *platform*. Perangkat *gateway* kemudian akan mendapatkan id pengguna dari *platform*, yang kemudian akan disimpan di dalam *database* lokal untuk keperluan lain, seperti mendaftarkan lampu.

3.2.6.3 Cara Mendaftarkan Lampu atau *Group* ke *Platform*

Ketika membuka halaman web yang terdapat pada perangkat *gateway*, pengguna dapat melihat daftar lampu yang terhubung. Pengguna kemudian dapat memilih opsi untuk mendaftarkan suatu lampu ke *platform*. Ketika opsi tersebut dipilih, pengguna akan diminta untuk memilih atribut apa saja yang akan dibagikan ke *platform*, dan atribut apa saja yang diperbolehkan untuk dikontrol dari *platform*. Setelah didaftarkan, lampu tersebut akan mendapatkan sebuah id dari *things management*

pada platform. Id ini kemudian akan disimpan secara otomatis di dalam *database* lokal di perangkat *gateway*.

3.3 Rancangan Pengujian

Untuk memastikan perangkat yang dibuat berjalan dengan baik, perlu dilakukan suatu pengujian. Pengujian akan dilakukan dengan cara mencoba perangkat untuk menjalankan suatu kasus uji. Berikut adalah kasus uji yang akan diujikan pada perangkat yang dibuat:

1. Kasus Uji 1: Menjadi *access point*.
2. Kasus Uji 2: Menghubungkan perangkat *gateway* ke *access point* lain.
3. Kasus Uji 3: Menyalakan lampu melalui halaman web pada perangkat *gateway*.
4. Kasus Uji 4: Mematikan lampu melalui halaman web pada perangkat *gateway*.
5. Kasus Uji 5: Mengubah tingkat kecerahan lampu melalui halaman web pada perangkat *gateway*.
6. Kasus Uji 6: Mengubah warna lampu melalui halaman web pada perangkat *gateway*.
7. Kasus Uji 7: Mengubah tingkat kejemuhan lampu melalui halaman web pada perangkat *gateway*.
8. Kasus Uji 8: Membuat sebuah *group* melalui halaman web pada perangkat *gateway*.
9. Kasus Uji 9: Menyalakan lampu dalam suatu *group* melalui halaman web pada perangkat *gateway*.
10. Kasus Uji 10: Mematikan lampu dalam suatu *group* melalui halaman web pada perangkat *gateway*.
11. Kasus Uji 11: Mengubah tingkat kecerahan lampu dalam suatu *group* melalui halaman web pada perangkat *gateway*.
12. Kasus Uji 12: Mengubah warna lampu dalam suatu *group* melalui halaman web pada perangkat *gateway*.

13. Kasus Uji 13: Mengubah tingkat kejemuhan lampu dalam suatu *group* melalui halaman web pada perangkat *gateway*.
14. Kasus Uji 14: Mendaftarkan lampu ke *platform* melalui halaman web pada perangkat *gateway*.
15. Kasus Uji 15: Mendaftarkan *group* ke *platform* melalui halaman web pada perangkat *gateway*.
16. Kasus Uji 16: Mengirimkan informasi mengenai atribut yang didaftarkan secara berkala ke *pub-sub system*.
17. Kasus Uji 17: Menyalakan lampu melalui pesan yang dikirimkan dari *pub-sub system*.
18. Kasus Uji 18: Mematikan lampu melalui pesan yang dikirimkan dari *pub-sub system*.
19. Kasus Uji 19: Mengubah tingkat kecerahan lampu melalui pesan yang dikirimkan dari *pub-sub system*.
20. Kasus Uji 20: Mengubah warna lampu melalui pesan yang dikirimkan dari *pub-sub system*.
21. Kasus Uji 21: Mengubah tingkat kejemuhan lampu melalui pesan yang dikirimkan dari *pub-sub system*.
22. Kasus Uji 22: Menyalakan lampu dalam suatu *group* melalui pesan yang dikirimkan dari *pub-sub system*.
23. Kasus Uji 23: Mematikan lampu dalam suatu *group* melalui pesan yang dikirimkan dari *pub-sub system*.
24. Kasus Uji 24: Mengubah tingkat kecerahan lampu dalam suatu *group* melalui pesan yang dikirimkan dari *pub-sub system*.
25. Kasus Uji 25: Mengubah warna lampu dalam suatu *group* melalui pesan yang dikirimkan dari *pub-sub system*.
26. Kasus Uji 26: Mengubah tingkat kejemuhan lampu dalam suatu *group* melalui pesan yang dikirimkan dari *pub-sub system*.

BAB 4

IMPLEMENTASI

Bab ini membahas tentang implementasi perangkat *gateway*. Hal ini meliputi persiapan alat, implementasi *gateway*, dan implementasi halaman kontrol.

4.1 Persiapan Alat

Alat yang digunakan pada tugas akhir ini adalah sebuah Raspberry Pi, sebuah *shield* bernama RaspBee, dan WiFi dongle dengan merek Tenda. Tipe Raspberry Pi yang digunakan untuk tugas akhir ini adalah Raspberry Pi Model B dengan spesifikasi *processor* 700 MHz *single core*, 512 MB RAM, dan sebuah *micro SD* 8GB. Sistem operasi yang digunakan adalah Raspbian, sebuah modifikasi distro linux Debian yang dibuat untuk Raspberry Pi. Gambar 4.1 menunjukkan foto dari perangkat Raspberry Pi yang digunakan.



Gambar 4.1: Raspberry Pi Model B

Di dalam Raspberry Pi tersebut dipasang berbagai aplikasi dan *tools* yang dibutuhkan untuk penggeraan tugas akhir ini, seperti deCONZ, apache *server*, dan MySQL *server*. Berbagai aplikasi dan berkas yang diperlukan diatur agar otomatis berjalan ketika Raspberry Pi dinyalakan. Perangkat lain yang digunakan untuk menghubungkan Raspberry Pi dengan jaringan lampu ZigBee adalah RaspBee. RaspBee dapat dihubungkan dengan Raspberry Pi melalui port GPIO yang tersedia

pada Raspberry Pi. Gambar 4.2 menunjukkan foto perangkat RaspBee yang digunakan, dan gambar 4.3 menunjukkan foto perangkat RaspBee ketika dipasang pada Raspberry Pi yang digunakan.



Gambar 4.2: Perangkat RaspBee yang digunakan



Gambar 4.3: Perangkat RaspBee ketika dipasang pada Raspberry Pi

Selain Raspberry Pi dan RaspBee, dibutuhkan juga sebuah modul WiFi agar perangkat *gateway* bisa menjadi sebuah *access point* dan bisa terhubung ke *access point* lain. Hal ini dibutuhkan agar pengguna bisa terhubung ke perangkat *gateway* dan perangkat *gateway* bisa terhubung ke internet. Untuk ini, diperlukan sebuah modul WiFi yang bisa berperan sebagai *access point* dan di saat yang sama melakukan hubungan dengan *access point* lain. Karena keterbatasan

perangkat yang ada, dalam tugas akhir ini akan digunakan dua buah WiFi *dongle* dengan merek Tenda untuk menjadi sebuah modul WiFi yang diinginkan.

@todo

foto tenda

4.2 Implementasi *Gateway*

Implementasi *gateway* pada tugas akhir ini adalah implementasi terkait pengiriman data ke *platform* atau menerima data dari *platform* dan disampaikan ke lampu yang dituju. Hal ini meliputi implementasi MQTT *client* dan translasi pesan MQTT ke perintah REST untuk disampaikan ke deCONZ.

4.2.1 Implementasi MQTT *Client*

Pembuatan MQTT *client* pada tugas akhir ini dilakukan dengan menggunakan *library* Paho yang dibuat oleh Eclipse, dan dengan memodifikasi kode yang dibuat oleh Fauziah Rahmah dalam tugas akhirnya [14]. Selain itu, digunakan juga *library* JDBC yang dibuat oleh Oracle untuk bisa terhubung dengan *database* MySQL.

Langkah pertama yang dilakukan saat program dijalankan adalah membuat sebuah objek *gateway* sekaligus menjalankan sebuah *timer* untuk menjalankan sebuah fungsi di dalam objek tersebut setiap sepuluh detik. Hal ini dilakukan karena terdapat sebuah tugas yang harus dilakukan setiap suatu jangka waktu tertentu secara terus menerus, yaitu mengambil informasi mengenai lampu dan mengirimkannya ke *platform*. Berikut merupakan potongan kode untuk membuat objek *gateway* sekaligus menjalankan *timer*.

Kode 4.1: Membuat objek *gateway*

```
1 Timer timer = new Timer();
2 timer.schedule(new Gateway(), 0, 10000);
```

Ketika objek *gateway* dibuat, hal yang pertama dilakukan adalah mengambil id pengguna yang tersimpan di *database* lokal perangkat *gateway*. Setelah mendapat id pengguna, kemudian dibuat objek *client* MQTT yang berperan sebagai *publisher* dan *subscriber*. Ketika membuat objek *client*, parameter yang dibutuhkan adalah URI yang memberitahukan alamat dari *broker*. Dalam tugas akhir ini, alamat *broker* sekaligus merupakan alamat *platform*, yaitu 128.199.236.53, dan *port default* yang digunakan oleh MQTT adalah 1883. Berikut merupakan potongan kode untuk membuat objek *client* MQTT.

Kode 4.2: Membuat objek *client* MQTT

```

1 //mengambil user id
2 this.userID = getUserId();
3 try {
4 // membuat client sebagai publisher
5 clientPub = new MqttClient("tcp://128.199.236.53:1883", "Publisher"
6 );
7 System.out.println("Publisher_created");
8 // membuat client sebagai subscriber
9 clientSub = new MqttClient("tcp://128.199.236.53:1883", "Subscriber");
10 System.out.println("Subscriber_created");

```

Setelah objek *client* dibuat, *client* tersebut kemudian perlu dihubungkan dengan *broker*. Ketika dihubungkan dengan *broker*, *client* yang berperan sebagai *subscriber* akan sekaligus melakukan proses *subscribe* ke sebuah topik tertentu. Topik ini harus spesifik hanya untuk satu pengguna, namun dapat menerima semua pesan yang berkaitan dengan pengguna tersebut. Oleh karena itu, digunakanlah *single level wildcard* untuk kebutuhan tersebut. Berikut adalah potongan kode untuk menghubungkan *client* dengan *broker*.

Kode 4.3: Menghubungkan *client* dengan *broker*

```

1 connectMosquitto(clientSub);
2 connectMosquitto(clientPub);
3 ...
4 c.connect();
5 c.setCallback(this);
6 if(c.getClientId().equalsIgnoreCase("Subscriber")){
7 // client diatur agar bisa menerima pesan dari broker
8 c.setCallback(Gateway.this);
9 String topik = "sot/g/" + userID + "/+/+/ctl";
10 System.out.println(topik);
11 c.subscribe(topik);
12 System.out.println("Subscriber_connected");

```

Seperti yang telah dijelaskan sebelumnya, bahwa setiap sepuluh detik, *gateway* akan mengirimkan informasi ke *broker* mengenai kondisi lampu yang terhubung. Untuk melakukan hal ini, pertama *gateway* perlu melihat di *database* lokal mengenai informasi lampu yang terhubung dan informasi apa saja yang ingin dibagikan ke *platform*. Setelah itu, *client* yang berperan sebagai *publisher* akan melakukan *publishing* informasi-informasi tersebut. Proses *publishing* dilakukan dengan mekanisme yang dijelaskan sebelumnya, yaitu satu topik untuk

setiap atribut pada setiap lampu. Berikut adalah potongan kode yang melakukan *publishing* tersebut.

Kode 4.4: Publishing yang dilakukan setiap sepuluh detik

```

1 rs = doQuery("Select * from things");
2 try{
3 while(rs.next()){
4     String id = rs.getString(1);
5     String localId = rs.getString(4);
6     if (rs.getString(6).length() > 0) {
7         String access[] = rs.getString(6).split(",");
8         String url = "http://localhost:8080/api/" + apiKey + "/lights
9             /" + localId;
10        String jsonString = executeREST("GET", url, null);
11        if (jsonString != null && jsonString.charAt(0) != '[') {
12            JSONObject jsonObject = new JSONObject(jsonString);
13            JSONObject newJSON = jsonObject.getJSONObject("state");
14            jsonObject = new JSONObject(newJSON.toString());
15
16            for (String attr : access) {
17                if (attr.equals("on")) {
18                    publish("sot/g/" + userID + "/undefined/" + id + "/" +
19                        attr + "/acc", attr + ":" + (jsonObject.
20                            getBoolean(attr) == true ? "true" : "false
21                            "));
22                    System.out.println(attr + "_lampa_ke_" + localId +
23                        ":" + (jsonObject.getBoolean(attr) == true
24                            ? "true" : "false"));
25                } else{
26                    publish("sot/g/" + userID + "/undefined/" + id + "/" +
27                        attr + "/acc", attr + ":" + jsonObject.getInt(
28                            attr));
29                    System.out.println(attr + "_lampa_ke_" + localId +
30                        ":" + jsonObject.getInt(attr));
31                }
32            }
33        }
34    }
35 }
```

Setiap kali *broker* mengirim pesan ke *gateway*, sebuah *method* pada objek *gateway*, yaitu *method* `messageArrived` akan dipanggil. *Method* ini merupakan *method* yang dioverride dari *interface* `MqttCallback`. *Method* ini memiliki dua buah parameter, yaitu topik dengan tipe data `String` dan isi pesan dengan tipe data

`MqttMessage`. Pesan yang diterima kemudian perlu diubah menjadi `String` agar bisa diolah dengan mudah. Kemudian, topik dan isi pesan dikirimkan ke `method checkCommand` yang fungsinya akan dijelaskan pada bagian selanjutnya. Berikut merupakan potongan kode `method messageArrived`.

Kode 4.5: *Method messageArrived*

```

1 @Override
2 public void messageArrived(String topic, MqttMessage msg) {
3     System.out.println("Topic : " + topic + ", message : " + msg);
4     String command = msg.toString();
5     // mengecek isi pesan dari broker
6     checkCommand(topic, command);
7 }
```

4.2.2 Translasi Pesan MQTT ke Pesan REST

Seperti yang dijelaskan sebelumnya, bahwa perlu dilakukan translasi atau perubahan bentuk perintah, dari yang berupa topik dan pesan MQTT, menjadi sebuah perintah untuk mengeksekusi API REST yang disediakan oleh deCONZ. Perintah yang diterima dan diteruskan ke deCONZ dibatasi hanya sebatas atribut apa saja yang diperbolehkan oleh pengguna untuk dikontrol dari *platform*. Proses ini dilakukan dalam `method checkCommand` yang telah disebutkan dalam bagian sebelumnya. Pada `method` inilah dilakukan pengolahan topik dan pesan dari *broker* menjadi perintah REST.

Informasi yang disampaikan melalui topik adalah id perangkat atau lampu, dan atribut apa yang ingin di kontrol. Sedangkan untuk isi pesan, hanya berisi nilai baru dari atribut yang diinginkan. Id yang terdapat di dalam topik masih merupakan id yang disimpan oleh *things management*, sehingga perlu diubah ke id lokal yang disimpan di dalam *database*. Untuk mendapatkan id lokal, telah dibuat method bernama `getLocalId` yang menerima parameter id dari *things management* dan mengembalikan id lokal yang tersimpan di *database*. Selain itu, perlu diketahui juga apakah id yang dituju adalah sebuah lampu atau *group*. Setelah mendapatkan id lokal dan tipe dari id yang dituju, selanjutnya dipanggil fungsi API REST yang disediakan oleh deCONZ. Potongan kode berikut menampilkan isi dari `method checkCommand`.

Kode 4.6: *Method checkCommand*

```

1 void checkCommand(String topic, String command) {
2     String msg = "";
```

```

3   String[] topicArr = topic.split("/");
4   String localId = getLocalId(topicArr[4]);
5   String tipe = getTipe(topicArr[4]);
6   String attr = topicArr[5];
7   String allowed[] = getAttr(topicArr[4]).split(", ");
8   String url = "";
9   if (arrayContain(allowed, attr)) {
10     if (tipe.equals("Lampu")) {
11       url = "http://localhost:8080/api/" + apiKey + "/lights/" +
12         localId + "/state";
13     } else{
14       url = "http://localhost:8080/api/" + apiKey + "/groups/" +
15         localId + "/action";
16     }
17     if (Integer.parseInt(localId) > 0) {
18       msg = executeREST("PUT", url, "{\""+attr+"\":\""+command +
19         "\"}");
20     } else{
21       System.out.println("id_lampu_tidak_ditemukan");
22     }
23   }
}

```

4.3 Implementasi Halaman Kontrol Lampu

Pengguna juga dapat melakukan kontrol terhadap lampu yang terhubung dengan langsung melalui perangkat *gateway* atau tanpa harus terhubung ke internet. Untuk memenuhi hal ini, telah dibuat sebuah aplikasi berbasis web yang dijalankan di perangkat *gateway* secara lokal. Aplikasi ini diimplementasikan menggunakan bahasa pemrograman PHP dan berjalan di atas *server* Apache. Melalui aplikasi ini, pengguna dapat melakukan berbagai hal, seperti menghidupkan dan mematikan lampu, mengubah *brightness*, mengubah *hue*, dan mengubah *saturation*. Pengguna juga dapat membuat *group* yang terdiri dari lampu-lampu yang dimilikinya. Pengguna kemudian dapat melakukan *log in* menggunakan akun *platform* yang dimilikinya, dan kemudian dapat mendaftarkan lampu dan *group* yang ada ke *platform*. Secara umum, terdapat dua halaman pada aplikasi web ini, yaitu halaman utama untuk mengontrol lampu dan *group*, dan halaman untuk melakukan registrasi lampu atau *group*.

4.3.1 Halaman Utama Untuk Mengontrol Lampu

Pada halaman ini, pengguna dapat melakukan beberapa hal, yaitu melihat daftar lampu yang terhubung, menyalakan atau mematikan lampu, mengubah *brightness* lampu, mengubah warna lampu, dan mengubah *saturation* lampu. Pengguna juga dapat membuat *group* lampu, dan terakhir pengguna dapat *log in* menggunakan akun yang digunakan untuk mendaftar pada *platform*.

4.3.1.1 Melihat Daftar Lampu yang Terhubung

Untuk bisa melihat daftar lampu yang terhubung, halaman ini akan menggunakan API REST yang disediakan oleh deCONZ. Untuk memanggil API REST yang disediakan deCONZ, digunakan fungsi `curl` yang sudah disediakan oleh PHP, dan penulis membungkusnya menjadi sebuah fungsi bernama `kurl` yang menerima parameter (`String URI`, `String method`, `String data`). Hasil dari pemanggilan API ini adalah sebuah objek JSON yang berisi daftar id lampu jika ada yang terhubung, atau objek kosong jika tidak ada lampu yang terhubung. Potongan kode berikut menampilkan kode untuk mengambil informasi daftar lampu yang terhubung.

Kode 4.7: Mengambil informasi daftar lampu

```

1 $output = kurl('localhost:8080/api/' . $apiKey . "/lights", "GET");
2 $listLampu = array();
3 if (!empty($output)) {
4     $json_obj = json_decode($output);
5     $id_lampu = [];
6     foreach ($json_obj as $key => $value) {
7         array_push($id_lampu, $key);
8     }

```

4.3.1.2 Mengubah Atribut Lampu

Setelah mendapat info mengenai id lampu, kemudian API REST deCONZ digunakan lagi untuk mendapat informasi lebih detail mengenai setiap lampu. Informasi ini kemudian ditampilkan pada halaman utama aplikasi. Selain menampilkan informasi, halaman ini juga menampilkan beberapa elemen HTML untuk melakukan kontrol, seperti *button* untuk menghidupkan atau mematikan lampu, *range input* untuk mengubah *brightness*, dan *plugin Javascript* bernama JSColor untuk mengubah warna lampu. Masing-masing fungsi akan memanggil sebuah fungsi Javascript dan menggunakan fungsi AJAX untuk memanggil fungsi PHP yang

bertugas menghubungi API REST deCONZ. Potongan kode berikut ini merupakan fungsi PHP yang dipanggil oleh halaman utama untuk memanggil API REST deCONZ. Fungsi ini dibuat agar bisa menerima permintaan kontrol untuk semua atribut.

Kode 4.8: fungsi setAttribute

```

1 <?php
2 function setAttribute(){
3     global $gatewayBaseUrl, $apiKey;
4     $id = $_POST['id'];
5     $value = explode(":::", $_POST['value']);
6     $attr = explode(":::", $_POST['attr']);
7     $mode = $_POST['mode'];
8     $data_json = '{';
9     for ($i=0; $i < count($attr); $i++) {
10         $data_json .= '"' . $attr[$i] . ":" . $value[$i];
11         if ($i < count($attr)-1) {
12             $data_json .= ',';
13         }
14     }
15     $data_json .= '}';
16     $url = "";
17     if ($mode == 1) {
18         $url = $gatewayBaseUrl.$apiKey."/lights/".$id."/state";
19     }else{
20         $url = $gatewayBaseUrl.$apiKey."/groups/".$id."/action";
21     }
22     $result = kurl($url, "PUT", $data_json);
23     $hasilDecode = json_decode(substr($result, 1,-1));
24     if (array_key_exists('success', $hasilDecode)) {
25         $message = $hasilDecode->success;
26         $state = "";
27         foreach ($message as $key => $value) {
28             $state = $value;
29         }
30         $state = ($state == true ? "true" : "false");
31         echo "sukses:".$state;
32     }else{
33         echo $data_json;
34     }
35 }
```

4.3.1.3 Membuat *Group* Lampu

Untuk membuat *group*, digunakan API REST yang juga disediakan oleh deCONZ. Ketika memilih pilihan untuk membuat *group* baru, pengguna akan diminta memasukkan nama *group* dan memilih lampu yang mana saja yang akan dimasukkan ke dalam *group* tersebut. Setelah itu, sebuah fungsi PHP akan dipanggil untuk mendaftarkan *group* dengan informasi seperti yang diberikan oleh pengguna. Berikut adalah potongan kode untuk membuat *group* baru.

Kode 4.9: fungsi createGroup

```

1 <?php
2 function createGroup() {
3     global $gatewayBaseUrl,$apiKey;
4     $nama = $_POST['nama'];
5     $lampus = $_POST['group'];
6     if (!empty($lampus)) {
7         $konten = '{"name":"' . $nama . '"}';
8         $hasil = kurl($gatewayBaseUrl.$apiKey."/groups", "POST", $konten
9             );
10        $hasilDecode = json_decode(substr($hasil, 1,-1));
11        if (array_key_exists('success', $hasilDecode)) {
12            $id = $hasilDecode->success->id;
13            $konten = '{"lights":[';
14            foreach ($lampus as $lampa) {
15                $konten .= "'". $lampa . "', ";
16            }
17            $konten = rtrim($konten, ", ");
18            $konten.=']}';
19            $url = $gatewayBaseUrl.$apiKey."/groups/".$id;
20            echo kurl($url, "PUT", $konten);
21        }
22    }
23    header("Location:_index.php");
24    die();
}

```

4.3.1.4 *Log In* Pengguna

Untuk bisa mendaftarkan lampu dan *group* yang terhubung di perangkat *gateway* ke *platform*, diperlukan informasi id pengguna yang terdaftar di *platform*. Oleh karena itu, pengguna diharuskan melakukan *log in* sebelum mendaftarkan lampu atau *group*. Dalam melakukan *log in* ini, pengguna hanya perlu memasukkan email yang digunakan untuk mendaftar di *platform*. Setelah *log in* dan didapatkan id

pengguna yang terdaftar di *platform*, id ini akan disimpan di *database* lokal untuk keperluan pendaftaran lampu dan *group*. Informasi id ini juga akan digunakan oleh *gateway* untuk melakukan proses *subscribe* ke topik yang sesuai. Berikut ini adalah potongan kode untuk melakukan *log in*.

Kode 4.10: fungsi createGroup

```

1 <?php
2 function login(){
3     global $baseUrl;
4     $email = $_POST['email'];
5     $url = $baseUrl."user/check/".$email;
6     $hasil = kurl($url, "GET", "");
7     $hasilDecode = json_decode($hasil);
8     if (array_key_exists('user', $hasilDecode)) {
9         echo dbInsert("setting", array(
10             "kunci"=>"user_id",
11             "value"=>$hasilDecode->user));
12     }else{
13         echo "gagal";
14     }
15 }
```

4.3.2 Implementasi Halaman Registrasi Lampu

Pengguna dapat mendaftarkan lampu dan *group* yang terhubung ke perangkat *gateway* ke *platform*. Karena kesamaan atribut yang dimiliki oleh lampu dan *group*, ketika didaftarkan sebuah *group* akan dianggap sebagai lampu. Saat mendaftarkan lampu atau *group*, pengguna juga bisa sekaligus mendaftarkan atribut yang dimiliki oleh lampu atau *group*. Atribut ini adalah *On / Off*, *brightness*, *saturation*, dan *hue*. Ketika mendaftarkan, pengguna diminta memilih hak akses bagi setiap atribut. Terdapat dua hak akses, yaitu hak untuk membagikan informasi terkait suatu atribut ke *platform*, dan hak untuk memberikan hak kontrol suatu atribut dari *platform*. Untuk mendaftarkan atribut, disediakan sebuah fungsi yang bernama attrRegister yang menerima parameter hak akses, nama, tipe, deskripsi, nilai minimal , dan nilai maksimal untuk suatu atribut. Berikut adalah potongan kode untuk mendaftarkan atribut.

Kode 4.11: fungsi attrRegister

```

1 <?php
2 function attrRegister($array, $nama, $tipe, $description, $min,
3     $max) {
```

```

3   global $thingsID, $baseUrl;
4   $userID = getUserId();
5   $urlAtribut = $baseUrl."user/".$userID."/thing/".$thingsID."/"
6       property/register";
7   $konten = '{
8     "name": "'.$nama.'",
9     "access": "'.$(in_array('acc', $array) ? 'true' : 'false').'",
10    "control": "'.$(in_array('ctrl', $array) ? 'true' : 'false').'",
11    "valueType": "'.$tipe.'",
12    "description": "'.$description.'",
13    "min": "'.$min.'",
14    "max": "'.$max.''
15  }';
16  echo $konten."<br>";
17  $hasil = kurl($urlAtribut, "POST", $konten). "<br>";
18  echo $hasil;
19  return $hasil;
}

```

Setelah mendaftarkan lampu atau *group* beserta atributnya, akan didapatkan informasi id perangkat dari *platform*. Id ini kemudian disimpan di dalam *database* lokal untuk digunakan oleh *gateway*. Selain id, informasi mengenai atribut apa saja yang didaftarkan juga akan disimpan. Informasi ini digunakan oleh *gateway* dalam menentukan atribut mana saja yang akan dilakukan proses *subscribe* dan atribut mana yang akan dilakukan proses *publishing*. Seluruh hal ini dilakukan dalam sebuah fungsi bernama `registerThings`. Berikut adalah potongan kode ketika mendaftarkan lampu atau *group* serta ketika melakukan penyimpanan informasi ke *database*.

Kode 4.12: fungsi registerThings

```

1 <?php
2 function registerThings() {
3   ...
4   $thingsToken = kurl($baseUrl."user/".$userID."/thing/register",
5       POST", $kontenRequest);
6   $json_obj = json_decode($thingsToken);
7   if (strpos($json_obj->message, 'failed') !== false) {
8     echo "gagal";
9   }else{
10     $thingsID = $json_obj->id;
11     echo "things_ID: ".$thingsID."<br>";
12     $control = "";
13     $access = "";
}

```

```

13 $urlAtribut = $baseUrl."user/".$userID."/thing/".$thingsID."/property/register";
14 for ($i=0; $i < count($attr); $i++) {
15     if (!empty($attr[$i])) {
16         $hasil = "";
17         $attrName = "";
18         switch ($i) {
19             case 0:
20                 $hasil = attrRegister($attr[$i],"bri","INT","Tingkat_kecerahan_Lampu",0,255);
21                 $attrName = "bri";
22                 break;
23             ...
24         }
25         if (strpos($hasil,'added:') !== false) {
26             in_array('acc', $attr[$i]) ? $access .= $attrName.","
27                 : '';
28             in_array('ctrl', $attr[$i]) ? $control .= $attrName.","
29                 : '';
30         }
31     }
32     $access = rtrim($access, ", ");
33     $control = rtrim($control, ", ");
34
35 dbInsert('things',array(
36     'id'=>$thingsID,
37     'type'=> ($tipe == 1) ? 'Lampu' : 'Group',
38     'nama'=>$nama,
39     'local_id'=>$local_id,
40     'control'=>$control,
41     'access'=>$access,
42   ));
43 }
44 ...
45 }

```

4.4 Implementasi Perangkat Gateway

Agar perangkat *gateway* bisa berjalan seperti yang diinginkan, perlu dilakukan beberapa konfigurasi. Hal-hal yang perlu dikonfigurasi adalah pengaturan aplikasi yang berjalan ketika *start up*, dan pengaturan agar perangkat *gateway* bisa menjadi

sebuah *access point* dan dihubungkan ke *access point* lain.

4.4.1 Konfigurasi *Start Up*

Agar perangkat *gateway* bisa berjalan tanpa perlu dilakukan secara manual oleh pengguna, perlu dibuat konfigurasi agar beberapa *tools* atau aplikasi berjalan secara otomatis ketika perangkat menyala. *Tools* yang diperlukan agar berjalan secara otomatis ketika perangkat menyala adalah deCONZ. deCONZ diperlukan karena seluruh operasi terkait lampu akan menggunakan API REST yang disediakan oleh deCONZ.

Untuk bisa berjalan, deCONZ memerlukan sebuah X *server* yang sudah berjalan. Untuk menjalankan X *server*, secara *default* diperlukan pengguna untuk melakukan *log in* menggunakan GUI. Untuk *log in* menggunakan GUI pada Raspberry Pi, diperlukan sebuah layar eksternal atau penggunaan aplikasi lain. Karena perangkat *gateway* ini ditargetkan agar bisa berjalan tanpa perangkat tambahan, maka perlu dilakukan konfigurasi agar perangkat secara otomatis menjalankan sebuah X *server* ketika menyala, dan mengarahkan X *server* tersebut ke sebuah tty, sehingga tidak diperlukan sebuah layar. Setelah berhasil menjalankan X *server*, deCONZ secara otomatis dijalankan pada X *server* yang telah berjalan. Potongan kode berikut merupakan konfigurasi yang dibutuhkan untuk melakukan tersebut.

Kode 4.13: kode untuk menjalankan X *server* dan memulai deCONZ

```

1 #!/bin/bash
2
3 X :8 vt8 > /tmp/X8.log 2>&1 &
4 X_PID=$!
5
6 export DISPLAY=:8
7 deCONZ --auto-connect=1 --http-port=8080
8
9 kill $X_PID

```

Pada potongan kode diatas, dijalankan sebuah X *server* dan diarahkan pada tty8. Kemudian, log dari X *server* diarahkan ke file X8.log. Setelah itu, dilakukan pengeaturan agar deCONZ dijalankan pada X *server* yang telah dijalankan sebelumnya. Kode ini disimpan pada suatu file, yang kemudian akan dijalankan ketika *start up*. Untuk menjalankan file tersebut ketika *start up*, perlu dilakukan modifikasi file rc.local pada folder /etc. Pada file tersebut, ditambahkan baris /root/deconz > /root/deconz.out 2> /root/deconz.err &. Baris tersebut berarti ketika file dijalankan, keluaran dari aplikasi yang dijalankan akan ditaruh ke dalam file

deconz.out dan jika terdapat *error* akan ditaruh ke dalam file deconz.err.

4.4.2 Konfigurasi Access Point

Ketika pertama kali perangkat *gateway* dijalankan, perangkat ini akan berfungsi sebagai sebuah *access point*. Dengan ini, pengguna kemudian bisa terhubung ke perangkat *gateway* dengan menggunakan komputer atau *smartphone* yang dimilikinya. Untuk bisa menjadikan perangkat ini sebagai *access point*, dibutuhkan sebuah aplikasi dengan nama hostapd. Pada hostapd, dilakukan konfigurasi seperti pada potongan kode berikut.

Kode 4.14: Konfigurasi hostapd

```

1 interface=wlan0
2 ssid=[access_point_name]
3 hw_mode=g
4 channel=6
5 auth_algs=1
6 wmm_enabled=0

```

Setelah dilakukan konfigurasi diatas, hostapd dapat dijalankan dengan perintah service hostapd start. Untuk menjalankan hostapd secara otomatis setiap perangkat *gateway* dinyalakan, digunakan perintah update-rc.d hostapd enable. Setelah terhubung, pengguna dapat memilih *access point* lain yang berada di sekitarnya yang memiliki koneksi internet melalui suatu halaman web yang disediakan. Untuk melihat *access point* apa saja yang berada di sekitar perangkat, dapat digunakan perintah iwlist wlan1 scan | grep ESSID. Setelah ditampilkan, pengguna kemudian dapat melakukan klik pada *access point* yang dipilih. Perangkat *gateway* kemudian akan terhubung dengan *access point* yang dipilih. Potongan kode berikut menunjukkan implementasi untuk terhubung ke suatu *access point*.

Kode 4.15: Kode untuk terhubung ke suatu *access point*

```

1 if (isset($_GET['id'])) {
2     $id = $_GET['id'];
3     $bash = "sudo_iwconfig_wlan1_essid_". $id;
4     $output = shell_exec($bash);
5     $bash = "sudo_dhclient_wlan1";
6     $output = shell_exec($bash);
7 }

```

BAB 5

PENGUJIAN

Bab ini menjelaskan hasil pengujian yang dilakukan terhadap perangkat yang dibuat untuk menguji apakah hasil implementasi berjalan sesuai seperti yang diharapkan atau tidak. Penjelasan meliputi perangkat yang digunakan dan hasil pengujian.

5.1 Perangkat yang Digunakan

Perangkat yang digunakan dalam pengujian ini adalah dua buah lampu yang akan dikontrol. Masing-masing lampu ini adalah sebuah lampu berbasis ZigBee dengan merek Philips Hue. Philips Hue adalah sebuah lampu ZigBee yang mengimplementasikan profil ZigBee *Light Link*. Philips Hue merupakan sebuah *Color Dimmable Light*, sehingga dapat diubah warnanya dan tingkat kecerahannya. Gambar 5.1 menunjukkan foto kedua lampu yang digunakan.



Gambar 5.1: Lampu Philips Hue yang digunakan untuk pengujian

5.1.1 Hasil Pengujian

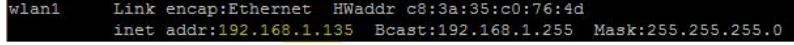
1. Kasus Uji 1: Menjadi *access point*.

Tabel 5.1: Hasil pengujian kasus uji 1

Perintah	Perangkat <i>gateway</i> dinyalakan
Prekondisi	-
Hasil	Perangkat <i>gateway</i> menyala dengan <i>access point</i> bernama "ZigBee_Gateway" muncul 
Ekspektasi	<i>Access point</i> bernama "ZigBee_Gateway" muncul

2. Kasus Uji 2: Menghubungkan perangkat *gateway* ke *access point* lain.

Tabel 5.2: Hasil pengujian kasus uji 2

Perintah	Membuka halaman 192.168.42.1/ta/wifi.php, dan memilih salah satu <i>access point</i> lain
Prekondisi	Perangkat yang digunakan untuk mengakses sudah terhubung ke perangkat <i>gateway</i>
Hasil	Perangkat <i>gateway</i> terhubung ke <i>access point</i> yang dipilih dan mendapatkan alamat IP 192.168.1.135. 
Ekspektasi	Perangkat <i>gateway</i> terhubung ke <i>access point</i> yang dipilih dan mendapatkan sebuah alamat IP

3. Kasus Uji 3: Menyalakan lampu melalui halaman web pada perangkat *gateway*

Tabel 5.3: Hasil pengujian kasus uji 3

Perintah	Tombol 'Nyalakan' ditekan
Prekondisi	Perangkat <i>gateway</i> sudah berjalan, lampu sudah terhubung dengan perangkat <i>gateway</i> dan dalam keadaan mati
Hasil	Lampu menyala 
Ekspektasi	Lampu menyala

4. Kasus Uji 4: Mematikan lampu melalui halaman web pada perangkat *gateway*

Tabel 5.4: Hasil pengujian kasus uji 4

Perintah	Tombol 'Matikan' ditekan
Prekondisi	Perangkat <i>gateway</i> sudah berjalan, lampu sudah terhubung dengan perangkat <i>gateway</i> dan dalam keadaan menyala
Hasil	Lampu mati 
Ekspektasi	Lampu mati

5. Kasus Uji 5: Mengubah tingkat kecerahan lampu melalui halaman web pada

perangkat *gateway*

Tabel 5.5: Hasil pengujian kasus uji 5

Perintah	<i>Slider</i> tingkat kecerahan digeser ke nilai yang lebih tinggi
Prekondisi	Perangkat <i>gateway</i> sudah berjalan, lampu sudah terhubung dengan perangkat <i>gateway</i> dan dalam keadaan menyala
Hasil	Lampu menjadi lebih terang 
Ekspektasi	Lampu menjadi lebih terang

6. Kasus Uji 6: Mengubah warna lampu melalui halaman web pada perangkat *gateway*

Tabel 5.6: Hasil pengujian kasus uji 6

Perintah	Mengubah warna pada <i>color picker</i> menjadi berwarna merah
Prekondisi	Perangkat <i>gateway</i> sudah berjalan, lampu sudah terhubung dengan perangkat <i>gateway</i> dan dalam keadaan menyala
Hasil	Lampu menjadi berwarna merah 
Ekspektasi	Lampu menjadi berwarna merah

7. Kasus Uji 7: Mengubah tingkat kejemuhan lampu melalui halaman web pada perangkat *gateway*

Tabel 5.7: Hasil pengujian kasus uji 7

Perintah	Menurunkan tingkat kejemuhan warna menggunakan <i>color picker</i>
Prekondisi	Perangkat <i>gateway</i> sudah berjalan, lampu sudah terhubung dengan perangkat <i>gateway</i> dan dalam keadaan menyala
Hasil	Lampu menjadi berwarna lebih muda 
Ekspektasi	Lampu menjadi berwarna lebih muda

8. Kasus Uji 8: Membuat sebuah *group* melalui halaman web pada perangkat *gateway*

Tabel 5.8: Hasil pengujian kasus uji 8

Perintah	Menekan tombol 'Buat Group', mengisi nama, kemudian memilih 'Light 1' dan 'Light 2', dan menekan tombol 'submit'
Prekondisi	Perangkat <i>gateway</i> sudah berjalan, lampu sudah terhubung dengan perangkat <i>gateway</i>
Hasil	Group terdaftar dengan anggota 'Light 1' dan 'Light 2'
Ekspektasi	Group terdaftar dengan anggota 'Light 1' dan 'Light 2'

9. Kasus Uji 9: Menyalakan lampu dalam suatu *group* melalui halaman web pada perangkat *gateway*

Tabel 5.9: Hasil pengujian kasus uji 9

Perintah	Tombol 'Nyalakan' pada bagian <i>groups</i> ditekan
Prekondisi	Perangkat <i>gateway</i> sudah berjalan, lampu sudah terhubung dengan perangkat <i>gateway</i> , <i>group</i> sudah dibuat, dan lampu dalam keadaan mati
Hasil	Kedua lampu menyala 
Ekspektasi	Kedua lampu menyala

10. Kasus Uji 10: Mematikan lampu dalam suatu *group* melalui halaman web pada perangkat *gateway*

Tabel 5.10: Hasil pengujian kasus uji 10

Perintah	Tombol 'Matikan' pada bagian <i>groups</i> ditekan
Prekondisi	Perangkat <i>gateway</i> sudah berjalan, lampu sudah terhubung dengan perangkat <i>gateway</i> , <i>group</i> sudah dibuat, dan lampu dalam keadaan menyala
Hasil	Kedua lampu mati 
Ekspektasi	Kedua lampu mati

11. Kasus Uji 11: Mengubah tingkat kecerahan lampu dalam suatu *group* melalui halaman web pada perangkat *gateway*

Tabel 5.11: Hasil pengujian kasus uji 11

Perintah	<i>Slider</i> tingkat kecerahan pada bagian <i>groups</i> digeser ke nilai yang lebih tinggi
Prekondisi	Perangkat <i>gateway</i> sudah berjalan, lampu sudah terhubung dengan perangkat <i>gateway</i> , <i>group</i> sudah dibuat, dan lampu dalam keadaan menyala
Hasil	Kedua lampu menjadi lebih terang 
Ekspektasi	Kedua lampu menjadi lebih terang

12. Kasus Uji 12: Mengubah warna lampu dalam suatu *group* melalui halaman web pada perangkat *gateway*

Tabel 5.12: Hasil pengujian kasus uji 12

Perintah	Mengubah warna pada <i>color picker</i> menjadi berwarna hijau
Prekondisi	Perangkat <i>gateway</i> sudah berjalan, lampu sudah terhubung dengan perangkat <i>gateway</i> , <i>group</i> sudah dibuat, dan lampu dalam keadaan menyala
Hasil	Kedua lampu menjadi berwarna hijau 
Ekspektasi	Kedua lampu menjadi berwarna hijau

13. Kasus Uji 13: Mengubah tingkat kejemuhan lampu dalam suatu *group* melalui halaman web pada perangkat *gateway*

Tabel 5.13: Hasil pengujian kasus uji 13

Perintah	Mengubah tingkat kejemuhan warna menggunakan <i>color picker</i> pada bagian <i>groups</i>
Prekondisi	Perangkat <i>gateway</i> sudah berjalan, lampu sudah terhubung dengan perangkat <i>gateway</i> , <i>group</i> sudah dibuat, dan lampu dalam keadaan menyala
Hasil	Kedua lampu menjadi berwarna berwarna lebih muda 
Ekspektasi	Kedua lampu menjadi berwarna berwarna lebih muda

14. Kasus Uji 14: Mendaftarkan lampu ke *platform* melalui halaman web pada perangkat *gateway*.

Tabel 5.14: Hasil pengujian kasus uji 14

Perintah	Menekan tombol 'Daftarkan Light2', memilih hak akses untuk setiap atribut, dan menekan tombol 'Daftar'
Prekondisi	Perangkat <i>gateway</i> sudah terhubung ke <i>access point</i> dengan koneksi internet, lampu sudah terhubung dengan perangkat <i>gateway</i>
Hasil	Lampu beserta atribut yang dipilih terdaftar dan mendapat id 557d23cb0fcf33af4e209d93
Ekspektasi	Lampu beserta atribut yang dipilih terdaftar dan mendapat sebuah id

15. Kasus Uji 15: Mendaftarkan *group* ke *platform* melalui halaman web pada perangkat *gateway*

Tabel 5.15: Hasil pengujian kasus uji 15

Perintah	Menekan tombol 'Daftarkan Lab Nok', memilih hak akses untuk setiap atribut, dan menekan tombol 'Daftar'
Prekondisi	Perangkat <i>gateway</i> sudah terhubung ke <i>access point</i> dengan koneksi internet, <i>group</i> sudah dibuat
Hasil	<i>Group</i> beserta atribut yang dipilih terdaftar dan mendapat id 557d10c80fcf33af4e209d8e
Ekspektasi	<i>Group</i> beserta atribut yang dipilih terdaftar dan mendapat sebuah id

16. Kasus Uji 16: Mengirimkan informasi mengenai atribut yang didaftarkan secara berkala ke *pub-sub system*

Tabel 5.16: Hasil pengujian kasus uji 16

Perintah	mosquitto_sub -h 128.199.236.53 -t sot/g/557903af5ad821942ef0af51/+//+/acc
Prekondisi	Perangkat <i>gateway</i> sudah terhubung ke <i>access point</i> dengan koneksi internet, lampu sudah terhubung ke perangkat <i>gateway</i>
Hasil	Mendapatkan informasi mengenai atribut yang didaftarkan setiap sepuluh detik <pre>C:\Users\Luqman>mosquitto_sub -h 128.199.236.53 -t sot/g/557903af5ad821942ef0af51/+//+/acc wkukwk hue: 65535 sat: 150 bri: 20 bri: 20 hue: 65535 sat: 150 on: false hue: 65535 sat: 150</pre>
Ekspektasi	Mendapatkan informasi mengenai atribut yang didaftarkan setiap sepuluh detik

17. Kasus Uji 17: Menyalakan lampu melalui pesan yang dikirimkan dari *pub-sub system*.

Tabel 5.17: Hasil pengujian kasus uji 17

Topik	sot/g/557903af5ad821942ef0af51/ledlight /557b693c1463cee933d9df01/on/ctl
Prekondisi	Lampu sudah didaftarkan dengan atribut terkait diberikan hak kontrol, lampu dalam keadaan mati
Isi pesan	true
Hasil	Lampu menyala 
Ekspektasi	Lampu menyala

18. Kasus Uji 18: Mematikan lampu melalui pesan yang dikirimkan dari *pub-sub system*.

Tabel 5.18: Hasil pengujian kasus uji 18

Topik	sot/g/557903af5ad821942ef0af51/ledlight /557b693c1463cee933d9df01/on/ctl
Prekondisi	Lampu sudah didaftarkan dengan atribut terkait diberikan hak kontrol, lampu dalam keadaan menyala
Isi pesan	false
Hasil	Lampu mati 
Ekspektasi	Lampu mati

19. Kasus Uji 19: Mengubah tingkat kecerahan lampu melalui pesan yang dikirimkan dari *pub-sub system*.

Tabel 5.19: Hasil pengujian kasus uji 19

Topik	sot/g/557903af5ad821942ef0af51/ledlight /557b693c1463cee933d9df01/bri/ctl
Prekondisi	Lampu sudah didaftarkan dengan atribut terkait diberikan hak kontrol, lampu dalam keadaan menyala
Isi pesan	200
Hasil	Lampu menjadi lebih terang 
Ekspektasi	Lampu menjadi lebih terang

20. Kasus Uji 20: Mengubah warna lampu melalui pesan yang dikirimkan dari *pub-sub system*.

Tabel 5.20: Hasil pengujian kasus uji 20

Topik	sot/g/557903af5ad821942ef0af51/ledlight /557b693c1463cee933d9df01/hue/ctl
Prekondisi	Lampu sudah didaftarkan dengan atribut terkait diberikan hak kontrol, lampu dalam keadaan menyala
Isi pesan	45000
Hasil	Lampu menjadi berwarna biru 
Ekspektasi	Lampu menjadi berwarna biru

21. Kasus Uji 21: Mengubah tingkat kejemuhan lampu melalui pesan yang dikirimkan dari *pub-sub system*.

Tabel 5.21: Hasil pengujian kasus uji 21

Topik	sot/g/557903af5ad821942ef0af51/ledlight /557b693c1463cee933d9df01/sat/ctl
Prekondisi	Lampu sudah didaftarkan dengan atribut terkait diberikan hak kontrol, lampu dalam keadaan menyala
Isi pesan	150
Hasil	Lampu menjadi berwarna lebih muda 
Ekspektasi	Lampu menjadi berwarna lebih muda

22. Kasus Uji 22: Menyalakan lampu dalam suatu *group* melalui pesan yang dikirimkan dari *pub-sub system*.

Tabel 5.22: Hasil pengujian kasus uji 22

Topik	sot/g/557903af5ad821942ef0af51/ledlight /557d10c80fcf33af4e209d8e/on/ctl
Prekondisi	<i>Group</i> sudah didaftarkan dengan atribut terkait diberikan hak kontrol, lampu anggota <i>group</i> dalam keadaan mati
Isi pesan	true
Hasil	Kedua lampu menyala 
Ekspektasi	Kedua lampu menyala

23. Kasus Uji 23: Mematikan lampu dalam suatu *group* melalui pesan yang dikirimkan dari *pub-sub system*.

Tabel 5.23: Hasil pengujian kasus uji 23

Topik	sot/g/557903af5ad821942ef0af51/ledlight /557d10c80fcf33af4e209d8e/on/ctl
Prekondisi	<i>Group</i> sudah didaftarkan dengan atribut terkait diberikan hak kontrol, lampu anggota <i>group</i> dalam keadaan menyala
Isi pesan	false
Hasil	Kedua lampu mati 
Ekspektasi	Kedua lampu mati

24. Kasus Uji 24: Mengubah tingkat kecerahan lampu dalam suatu *group* melalui pesan yang dikirimkan dari *pub-sub system*.

Tabel 5.24: Hasil pengujian kasus uji 24

Topik	sot/g/557903af5ad821942ef0af51/ledlight /557d10c80fcf33af4e209d8e/bri/ctl
Prekondisi	<i>Group</i> sudah didaftarkan dengan atribut terkait diberikan hak kontrol, lampu anggota <i>group</i> dalam keadaan menyala
Isi pesan	200
Hasil	Kedua lampu menjadi lebih terang 
Ekspektasi	Kedua lampu menjadi lebih terang

25. Kasus Uji 25: Mengubah warna lampu dalam suatu *group* melalui pesan yang dikirimkan dari *pub-sub system*.

Tabel 5.25: Hasil pengujian kasus uji 25

Topik	sot/g/557903af5ad821942ef0af51/ledlight /557d10c80fcf33af4e209d8e/hue/ctl
Prekondisi	<i>Group</i> sudah didaftarkan dengan atribut terkait diberikan hak kontrol, lampu anggota <i>group</i> dalam keadaan menyala
Isi pesan	65535
Hasil	Kedua lampu menjadi berwarna merah 
Ekspektasi	Kedua lampu menjadi berwarna merah

26. Kasus Uji 26: Mengubah tingkat kejemuhan lampu dalam suatu *group* melalui pesan yang dikirimkan dari *pub-sub system*.

Tabel 5.26: Hasil pengujian kasus uji 26

Topik	sot/g/557903af5ad821942ef0af51/ledlight /557d10c80fcf33af4e209d8e/sat/ctl
Prekondisi	<i>Group</i> sudah didaftarkan dengan atribut terkait diberikan hak kontrol, lampu anggota <i>group</i> dalam keadaan menyala
Isi pesan	150
Hasil	Kedua lampu menjadi berwarna lebih muda 
Ekspektasi	Kedua lampu menjadi berwarna lebih muda

BAB 6

PENUTUP

Pada bab ini dijelaskan mengenai hasil pengujian, kesimpulan dari implementasi perangkat *gateway* yang telah dilakukan, serta saran dari penulis untuk pengembangan selanjutnya.

6.1 Hasil Pengujian

Dari hasil pengujian, dapat dilihat bahwa perangkat *gateway* sudah dapat beroperasi dan digunakan sesuai dengan tujuan. Perangkat *gateway* dapat berperan sebagai sebuah *access point* secara otomatis setelah perangkat dinyalakan. Perangkat *gateway* juga kemudian dapat dihubungkan dengan *access point* lainnya melalui halaman aplikasi web yang disediakan. Operasi untuk mengontrol lampu melalui halaman aplikasi web yang disediakan juga dapat dilakukan dengan sukses. Setelah terhubung ke internet, pengguna juga dapat mendaftarkan lampu atau *group* ke *platform* dengan sukses. Perintah yang dikirimkan oleh *pub-sub system* ke topik yang sesuai juga berhasil diterima dan dikerjakan dengan sukses. Terakhir, perangkat *gateway* berhasil mengirimkan informasi mengenai kondisi setiap lampu atau *group* terdaftar ke *pub-sub system* setiap jangka waktu tertentu.

6.2 Kesimpulan

Dalam tugas akhir ini, penulis berhasil menghubungkan sebuah jaringan lampu berbasis ZigBee dengan sebuah *platform internet of things* berbasis sosial media. Dalam tugas akhir ini, digunakan Raspberry Pi sebagai perangkat untuk menjalankan semua operasi terkait sehingga tidak diperlukan lagi komputer tambahan. Berdasarkan implementasi yang telah dilakukan, penulis dapat mengambil beberapa kesimpulan, yaitu:

1. sesuatu

BIBLIOGRAPHY

- [1] ZigBee Alliance. *Network Device Gateway Specification*. ZigBee Alliance. 2011.
- [2] ZigBee Alliance. *Understanding ZigBee Gateway*. ZigBee Alliance. 2010.
- [3] Kevin Ashton. “That ‘internet of things’ thing”. In: *RFID Journal* 22.7 (2009), pp. 97–114.
- [4] Steve Smith Burnie Blakeley Deborah J. Boyd. *Introduction to Networking Technologies*. IBM Redbooks, 1994.
- [5] Cisco. *The Internet of Things*. URL: <http://share.cisco.com/internet-of-things.html>.
- [6] Shahin Farahani. *ZigBee wireless networks and transceivers*. Newnes, 2011.
- [7] Dominique Guinard, Mathias Fischer, and Vlad Trifa. “Sharing using social networks in a composable web of things”. In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*. IEEE. 2010, pp. 702–707.
- [8] HiveMQ. *MQTT Essentials Part 5: MQTT Topics and Best Practice*. Accessed: 2015-05-20. URL: <http://www.hivemq.com/mqtt-essentials-part-5-mqtt-topics-best-practices>.
- [9] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. “MQTT-SA publish/subscribe protocol for Wireless Sensor Networks”. In: *Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on*. IEEE. 2008, pp. 791–798.
- [10] S.Marusic dan M. Palaniswami J. Gubby R.Buyya. “Internet of Things (IoT): A vision, architectural elements, and future directions”. In: *Elsevier* (2013).
- [11] Giacomo Morabito Luigi Atzori Antonio Iera. “SIoT: Giving a Social Structure to the Internet of Things”. In: *IEEE Communications Letters* () .
- [12] Nisrina Luthfiyati. *Implementasi zigbee coordinator dengan REST*.
- [13] MQTT. *Frequently Asked Question*. Accessed: 2015-05-19. URL: mqtt.org/faq.
- [14] Fauziah Rahmawati. *Implementasi Home Automation Gateway untuk IOT Cloud Service berbasis ZigBee Network*.

LAMPIRAN

LAMPIRAN 1