



UNIVERSITAS INDONESIA

**Pengembangan dan Uji Kinerja Sistem Kendali
Elektronik Melalui Ratron**

SKRIPSI

**DEAN IQBAL DIVANDA
0706275965**

**FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KOMPUTER
DEPOK
JANUARI 2012**



UNIVERSITAS INDONESIA

**Pengembangan dan Uji Kinerja Sistem Kendali
Elektronik Melalui Ratron**

SKRIPSI

Diajukan sebagai salah satu syarat memperoleh gelar Sarjana Teknik

**DEAN IQBAL DIVANDA
0706275965**

**FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KOMPUTER
DEPOK
JANUARI 2012**

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan benar.

Nama : Dean Iqbal Divanda

NPM : 0706275965

Tanda Tangan : 

Tanggal : 6 Januari 2012

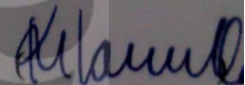
HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :
Nama : Dean Iqbal Divanda
NPM : 0706275965
Program Studi : Teknik Komputer
Judul Skripsi : Pengembangan dan Uji Kinerja Sistem Kendali
Elektronik Melalui Ratron

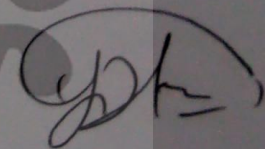
Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Prof. Dr.-Ing. Ir. Kalamullah Ramli, M.Eng.



Penguji : I Gde Dharma ST. MT.



Penguji : Yan Maraden ST. Msc.



Ditetapkan di : Depok
Tanggal : 6 Januari 2012

KATA PENGANTAR

Puji syukur saya panjatkan kehadiran Allah SWT, karena atas segala rahmat dan penyertaan-Nya saya dapat menyelesaikan skripsi ini. Saya menyadari bahwa skripsi ini tidak akan terselesaikan tanpa bantuan dari berbagai pihak. Oleh sebab itu, saya mengucapkan terima kasih kepada:

1. Prof. Dr. –Ing. Kalamullah Ramli, M.Eng, selaku dosen pembimbing yang membantu memberikan arahan dan nasihat, baik selama penulisan skripsi maupun selama masa studi di Teknik Komputer.
2. Kedua orang tua saya yang tidak pernah lelah dalam memberikan dukungan serta doa untuk kebaikan saya.
3. Teman-teman di Departemen Teknik Elektro Universitas Indonesia, khususnya program studi Teknik Komputer angkatan 2007, yang telah memberikan pengalaman dan kenangan terbaik selama masa studi.

Semoga Allah SWT berkenan membalas kebaikan semua pihak yang telah membantu dan menjadikan skripsi ini dapat bermanfaat bagi perkembangan ilmu pengetahuan.

Depok, 6 Januari 2012

Dean Iqbal Divanda

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama : Dean Iqbal Divanda
NPM : 0706275965
Program studi : Teknik Komputer
Departemen : Teknik Elektro
Fakultas : Teknik
Jenis karya : Skripsi

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

Pengembangan dan Uji Kinerja Sistem Kendali Elektronik Melalui Ratron

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non Eksklusif ini, Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 6 Januari 2012

Yang menyatakan



Dean Iqbal Divanda

ABSTRAK

Nama : Dean Iqbal Divanda
Program Studi : Teknik Komputer
Judul : Pengembangan dan Uji Kinerja Sistem Kendali Elektronik Melalui Ratron

Latar belakang pembuatan proyek ini adalah untuk memaksimalkan kemampuan dari teknologi surat elektronik (ratron) dan mikrokontroler. Aplikasi *Email Electronic Control* (EEC) atau Ratron Pengendali Elektronika (RPE) digunakan untuk melakukan interaksi antara ratron sebagai *user interface* yang berhubungan langsung dengan manusia dengan mikrokontroler sebagai perangkat tingkat rendahnya. Komunikasi antara ratron sebagai pengendali utama terhadap elektronika dapat menggunakan bahasa pemrograman JAVA dan C. Salah satu tujuan utama dalam proyek kali ini yaitu pengujian penggunaan IMAP IDLE sebagai media bantu notifikasi ratron baru, berjalan dengan baik dengan hasil menyalanya lampu setelah dilakukan pengujian. Analisa mengenai memori yang digunakan menyatakan bahwa penggunaan memori yang digunakan hanya sekitar 7 MB. Ini dinyatakan efisien berdasarkan referensi dalam (IBM, 2008). Aplikasi ini masih dinyatakan sebagai *single user and single command*. Penggunaan terbesar *method* dalam program ini digunakan untuk pengendalian ratron. Kecepatan rata-rata akses dari HP ke komputer adalah 26,064 detik dan dari komputer ke komputer adalah 11,9385 detik. JavaMail tidak dapat digunakan didalam sistem keamanan jaringan yang tinggi seperti didalam *proxy* kampus jadi sebagai alternatif penggunaan dapat digunakan internet langsung menggunakan *modem*. Perintah-perintah lain yang lebih kompleks seperti *multiple user and multiple command* dibutuhkan penelitian lebih lanjut. Konsep pengendalian elektronika melalui ratron yang sudah terbukti berhasil, dapat dikembangkan lebih lanjut untuk aplikasi *Smart Home*, automasi pabrik, dan seterusnya.

Kata Kunci: *ratron, sistem kendali, sistem terintegrasi, mikrokontroler, JAVA, IMAP, IDLE.*

ABSTRACT

Name : Dean Iqbal Divanda
Study Program : Computer Engineering
Title : Development and Performance Analysis of Email Based Control System

The main background of this project is to maximize the capabilities of email and microcontroller technology. User Applications of Email Electronic Control (EEC) or *Ratron Pengendali Elektronik* (RPE) is used to perform the interaction between email as user interface for humans that been used to direct contact with a microcontroller as the low level device. Communication between email as the main controller with the electronics device was uses the programming language such as Java and C. One of the main objectives in this project is to test the use of the IMAP IDLE notifications for new email. Our tests state that this application works appropriately. The result shows that LED/Lamp is activated after sending the email. The memory's application used for heap size is only about 7 MB. It is considered efficient as reffered to related research (IBM, 2008). Average speed for access/running time from HP to Computer is 26,064 second and from Computer to Computer is 11,9385 second. The biggest use of methods in this program are to control email. JavaMail can not be used in systems in such high network security environment like in the campus proxy so the use of the direct internet using a modem for is an alternative. This application can perform some simple commands for single user and single commamd. Further research work is require to improve it is capability.

Keyword: *email, control system, embedded system, microcontroller, JAVA, IMAP, IDLE*

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PERNYATAAN ORISINALITAS.....	ii
LEMBAR PENGESAHAN	iii
KATA PENGANTAR.....	iv
LEMBAR PERSETUJUAN PUBLIKASI KARYA ILMIAH	v
ABSTRAK	vi
ABSTRACT.....	vii
DAFTAR ISI.....	viii
DAFTAR GAMBAR	x
DAFTAR TABEL.....	xii
1 PENDAHULUAN	1
1.1 Latar belakang	1
1.2 Tujuan dan sasaran	2
1.3 Kelayakan Teknis dan Metodologi.....	2
1.4 Prospek dan Pemanfaatan Hasil Riset	3
2 KOMPONEN PENDUKUNG SISTEM KENDALI RATRON	5
2.1 Mikrokontroler.....	5
2.1.1 Mikrokontroler Adruino UNO.....	6
2.2 Embedded System	7
2.3 LED	8
2.4 USB	9
2.5 JAVA	10
2.5.1 JavaMail.....	11
2.5.2 Java IMAP IDLE	11
2.6 E-mail / Ratron	12
2.6.1 E-Mail / Ratron Server	13
2.6.2 IMAP.....	14
2.6.3 SMTP.....	15
3 PERANCANGAN SISTEM KENDALI ELEKTRONIK MELALUI RATRON.....	16
3.1 Ratron Pengendali Elektronik.....	16
3.2 Fase 1 : Pengiriman Ratron ke Ratron Server	17
3.2.1 Spesifikasi Fase 1	18
3.2.2 Cara Kerja Fase 1	18
3.3 Fase 2 : Modifikasi Ratron Server	18
3.3.1 Spesifikasi Fase 2	19
3.3.2 Cara Kerja Fase 2	19
3.4 Fase 3 : Penghubung Antara Ratron dan Mikrokontroler.....	20
3.4.1 Spesifikasi Fase 3	22
3.4.2 Cara Kerja Fase 3	22
3.5 Fase 4 : Pemrograman Mikrokontroler dan Peralatan Elektronik	23

3.5.1 Spesifikasi Fase 4	25
3.5.2 Cara Kerja Fase 4	26
4 IMPLEMENTASI DAN ANALISIS SISTEM KENDALI ELEKTRONIKA MELALUI RATRON.....	27
4.1 Implementasi Fase 4	27
4.1.1 Perangkat Keras	27
4.1.2 Perangkat Lunak	29
4.2 Implementasi Fase 3	32
4.2.1 Tatap Muka Aplikasi	32
4.2.2 Notifikasi Ratron	33
4.2.3 Pengambilan Ratron	34
4.2.4 Komunikasi Data Serial	39
4.3 Pengujian	39
4.4 Analisis	43
5 KESIMPULAN	48
6 DAFTAR REFERENSI.....	49
7 DAFTAR PUSTAKA	50

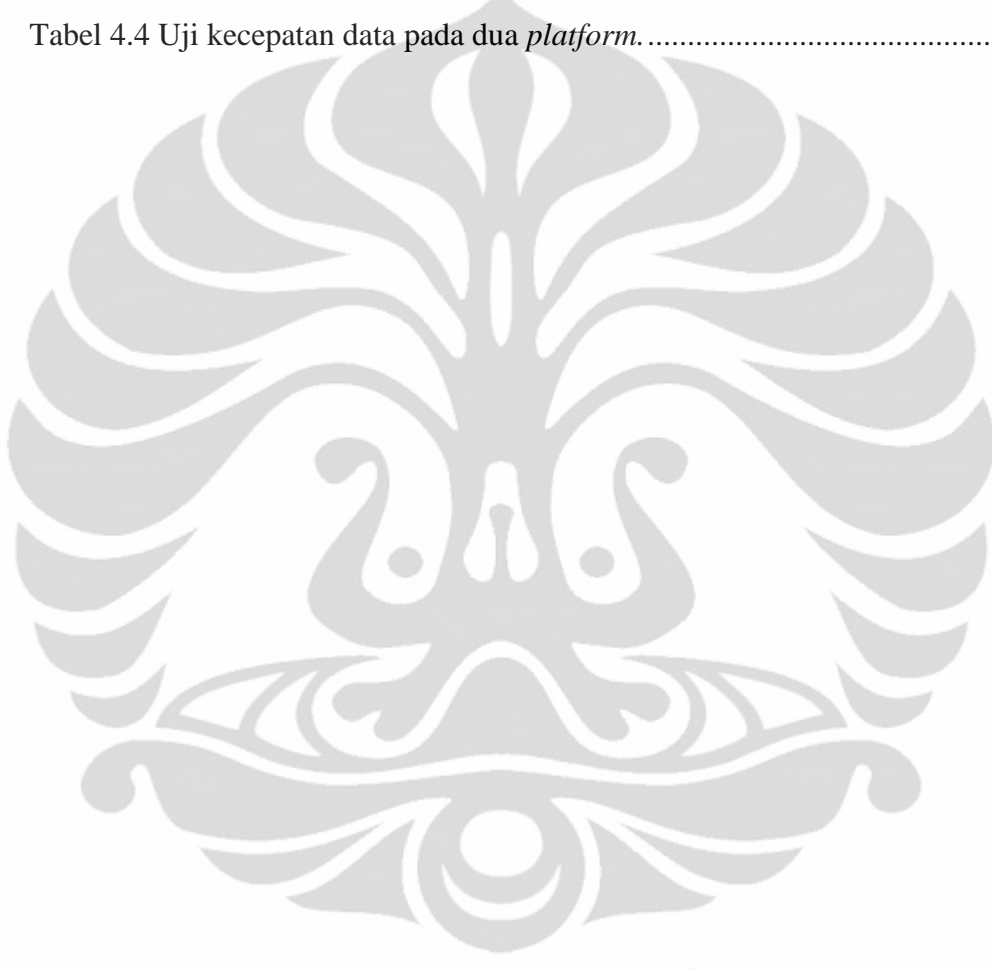
DAFTAR GAMBAR

Gambar 1.1 Sistem arsitektur	2
Gambar 2.1 Skematika diagram Arduino UNO Rev3	6
Gambar 2.2 Contoh tipe USB dan spesifikasinya (USB Implementers Forum, 2007)10	
Gambar 3.1 Skema umum.....	16
Gambar 3.2 Diagram proses EEC	16
Gambar 3.3 State diagram penelitian	17
Gambar 3.4 Sequence diagram EEC.....	20
Gambar 3.5 Class diagram fase 3.....	21
Gambar 3.6 Aktifitas diagram.....	24
Gambar 3.7 Kabel USB dengan tipe A dan B.....	25
Gambar 3.8 Algoritma cara kerja Mikrokontroler	26
Gambar 4.1 Skema dan bentuk asli rangkaian LED	27
Gambar 4.2 Gambar nyata rangkaian SisMin, LED dan kabel USB	28
Gambar 4.3 Inisialisasi variabel awal	29
Gambar 4.4 Pengaturan komunikasi serial.....	29
Gambar 4.5 Identifikasi adanya data yang masuk	30
Gambar 4.6 Pemisahan karakter dalam data yang masuk.....	31
Gambar 4.7 Identifikasi LED dan kontrol fungsinya.....	31
Gambar 4.8 Bentuk tatap muka aplikasi	32
Gambar 4.9 Kode untuk pengaktifan program.....	32
Gambar 4.10 Kode untuk membangun hubungan <i>server</i> ratron dan komputer	33
Gambar 4.11 Kode untuk perintah IDLE.....	33
Gambar 4.12 Kode untuk mengaktifkan pengambilan ratron baru.....	34
Gambar 4.13 Kode <i>buffer</i> dan pencarian ratron baru.....	35
Gambar 4.14 Pengurutan ratron yang masuk	35
Gambar 4.15 Identifikasi pengirim dan pengaktifan komunikasi serial	36
Gambar 4.16 Identifikasi alamat mikrokontroler.....	36
Gambar 4.17 Inisialisasi kanal serial dan aktifasi bila ada data.....	37

Gambar 4.18 Penetapan parameter dan pengaktifasian <i>thread</i>	38
Gambar 4.19 Kode <code>initwritetoport()</code>	38
Gambar 4.20 Kode <code>writetoport()</code>	39
Gambar 4.21 Letak alamat mikrokontroler dalam Windows Operating System	39
Gambar 4.22 Hasil <i>debug</i> untuk koneksi <i>server</i> dan komputer	40
Gambar 4.23 Hasil <i>debug</i> perintah IDLE	40
Gambar 4.24 Contoh pengiriman ratron	41
Gambar 4.25 Hasil <i>debug</i> pengambilan ratron dan komunikasi serial	42
Gambar 4.26 Hasil perintah R255 dan G255	42
Gambar 4.27 Grafik besar memori yang digunakan	43
Gambar 4.28 Banyak <i>thread</i> dan kelas yang digunakan.....	46

DAFTAR TABEL

Tabel 4.1 List perintah dalam sistem ini dan fungsinya.....	41
Tabel 4.2 Alokasi memori untuk masing objek	44
Tabel 4.3 Lama waktu untuk masing-masing proses <i>method</i>	45
Tabel 4.4 Uji kecepatan data pada dua <i>platform</i>	47



1 PENDAHULUAN

1.1 Latar belakang

Penggunaan surat elektronik (ratron) saat ini sudah sangat luas sekali pemakaiannya. Namun pemakaiannya saat ini masih sekedar untuk saling berkomunikasi antara sesama manusia. Salah satu contoh penggunaan ratron, misalnya untuk pemberitahuan tentang adanya sesuatu yang baru, baik itu peralatan elektronik ataupun jasa, atau bisa juga dipakai untuk memasarkan suatu produk. Namun masih ada lagi kemampuan ratron yang lain yaitu sebagai jembatan penghubung komunikasi antara bahasa tingkat tinggi milik manusia dengan bahasa tingkat rendah milik mesin/peralatan elektronika.

Babak baru dalam sistem pertelekomunikasian selanjutnya adalah komunikasi antara manusia dan mesin/elektronika dengan menggunakan ratron. Latar belakang pembuatan proyek ini adalah untuk menguji salah satu perintah didalam pemrograman Java yang mana saat ini perintah itu masih dalam tahap *Highly Experimental*. Perintah dalam pemrograman tersebut adalah IMAP IDLE, yaitu suatu perintah dalam Java untuk melakukan notifikasi ratron baru.

Ada alasan utama ratron dan sistem yang dikembangkan saat ini digunakan dalam komunikasi antara manusia dan mesin. Hal itu adalah dilihat dari segi biaya ratron dalam desain menggunakan 1KB untuk transfer data dan berharga Rp. 5,00 ke semua tujuan dan harga ini diambil dari *provider* TELKOMSEL (Telkomsel, 2010). Alasan inilah yang menjadi dasar serta latar belakang pembuatan proyek kali ini.

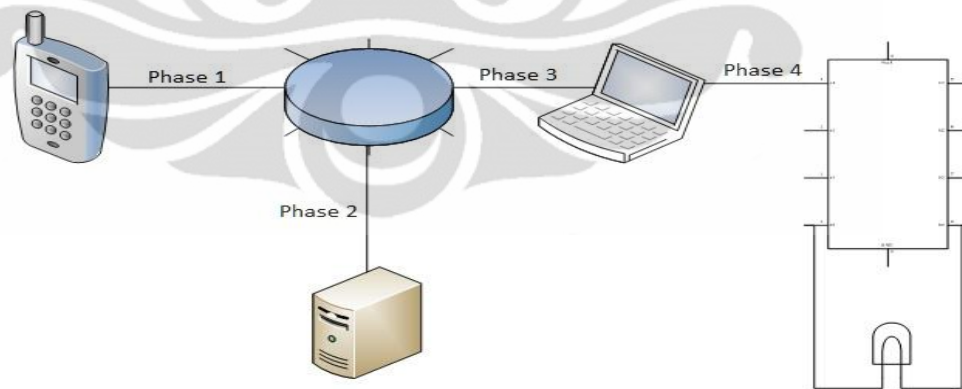
1.2 Tujuan dan sasaran

Tujuan dari penulisan dan riset kali ini adalah untuk melakukan pengujian salah satu perintah di Java yang saat ini masih dalam *Highly Experimental*, sehingga diharapkan dapat meningkatkan produktifitas hasil keilmuan terutama dalam bidang teknologi komputer dan elektronika. Selain itu diharapkan riset ini dapat membuka komunikasi antara manusia dan mikrokontroler melalui ratron.

Sasaran dari proyek kali ini adalah perusahaan umum atau elektronika, kebutuhan rumah tangga, serta instansi pemerintah yang bekerja di bidang teknologi dan dikhususkan pada komputer dan elektronika. Misalnya perusahaan produk komersial yang mempunyai perangkat elektroniknya sendiri untuk dilakukan pengaturan dalam jarak jauh.

1.3 Kelayakan Teknis dan Metodologi

Proyek kali ini bertujuan mengembangkan program dan sistem tertanam untuk melakukan komunikasi antara manusia dan mikrokontroler melalui ratron. Secara garis besar proyek ini akan mempunyai sistem arsitektur seperti pada Gambar 1.1.



Gambar 1.1 Sistem arsitektur

Seperti yang terlihat dalam Gambar 1.1. terdapat empat bagian penting dari proyek ini yaitu:

1. Fase 1 – Melakukan komunikasi ratron dari peralatan elektronik yang dapat menggunakan fasilitas ratron dan mengirimkannya ke internet.
2. Fase 2 – Melakukan penerimaan ratron dari peralatan elektronik yang berada di internet ke dalam *server* ratron.
3. Fase 3 – Melakukan pengambilan ratron dari *server* ratron ke dalam komputer untuk diproses.
4. Fase 4 – Melakukan komunikasi data antara mikrokontroler dengan komputer serta menjalankan perintah yang dikirimkan ke dalam mikrokontroler.

Metodologi yang dipakai dalam pengerjaan proyek ini adalah *trial and error*. Sistem metodologi ini adalah dengan melakukan percobaan rekayasa diperangkat keras atau lunak dengan menggunakan teori-teori yang sudah ada serta melakukan analisis kesalahan dari percobaan dan memperbaikinya. Sedangkan metodologi yang dipakai untuk penulisan adalah *practical used* yaitu dengan menggunakan teori-teori yang sudah terbukti berhasil dan telah mencapai proses pengujian kemudian menggunakan teori-teori tersebut dalam melakukan analisis desain serta pembuatan proyek.

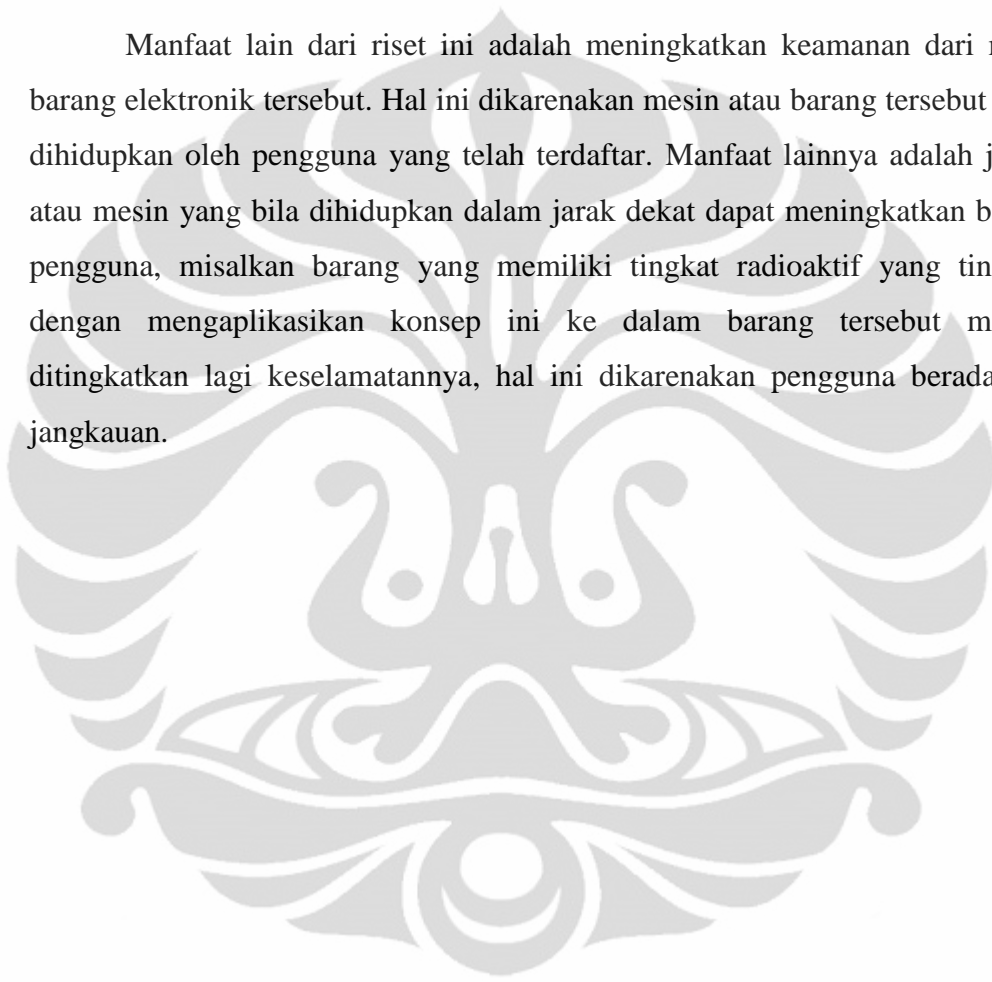
1.4 Prospek dan Pemanfaatan Hasil Riset

Untuk prospek kedepannya, hasil riset ini dapat dimanfaatkan hingga ketinggian yang lebih jauh dimana diberikan peranti tambahan yang mampu untuk mendukung kinerjanya. Desain dari proyek kali ini diusahakan dapat *easy to upgrade* sehingga memudahkan dalam implementasinya dikemudian hari. Pemanfaatan dari hasil riset ini dapat digunakan untuk berbagai macam hal dalam hal ini dapat dikhususkan ke bidang komunikasi elektronika dan komputer.

Misalkan, bila proyek ini diaplikasikan untuk perumahan maka akan menghasilkan sebuah *smarhome*. Sebuah *smarhome* ini dapat dikendalikan hanya melalui ratron. Fungsinya pun bisa berbagai macam misalkan menghidup atau mematikan lampu, membuka pintu elektronik bila mana ada saudara yang berkunjung disana, dan lain sebagainya.

Konsep ini juga bila diaplikasikan ke dalam sebuah pabrik maka dapat meningkatkan produktifitasnya. Misalkan, bila didalam pabrik terdapat sebuah mesin otomatis yang hanya bisa dihidupkan secara manual maka pengguna dapat menghidupkannya dari jarak jauh tanpa harus pengguna untuk pergi ke areal mesin.

Manfaat lain dari riset ini adalah meningkatkan keamanan dari mesin atau barang elektronik tersebut. Hal ini dikarenakan mesin atau barang tersebut hanya bisa dihidupkan oleh pengguna yang telah terdaftar. Manfaat lainnya adalah jika barang atau mesin yang bila dihidupkan dalam jarak dekat dapat meningkatkan bahaya bagi pengguna, misalkan barang yang memiliki tingkat radioaktif yang tinggi. Maka dengan mengaplikasikan konsep ini ke dalam barang tersebut maka dapat ditingkatkan lagi keselamatannya, hal ini dikarenakan pengguna berada jauh dari jangkauan.



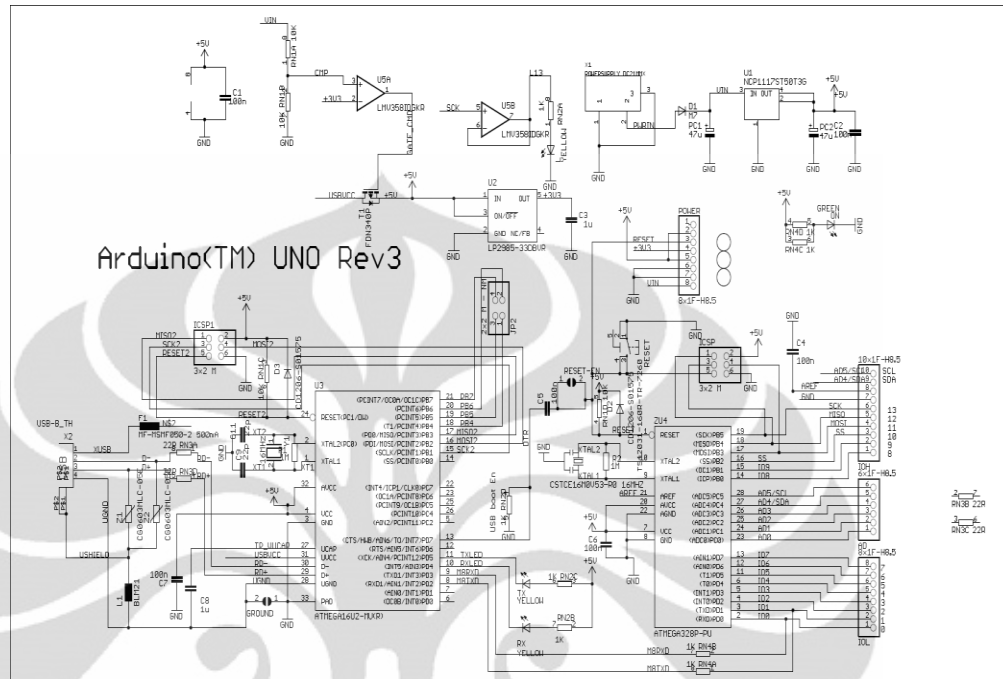
2 KOMPONEN PENDUKUNG SISTEM KENDALI RATRON

2.1 Mikrokontroler

Sebuah mikrokontroler (atau biasa disingkat dengan μC , uC atau MCU) adalah suatu sistem yang memiliki sebuah prosesor sebagai otak dari jalannya perangkat keras, memori untuk menyimpan data serta program yang menggunakan *assembly code* dan juga suatu unit untuk melakukan perhitungan logika. Mikrokontroler ini mempunyai banyak input serta output bila dibandingkan dengan mikroprosesor yang hanya dapat melakukan pemrosesan. Memori program biasanya dalam bentuk *Not OR Flash* atau kebalikan dari logika *OR*, dalam hal ini disingkat *NOR flash* atau *One Time Programmable Read-only Memory* (OTP ROM) juga dimasukkan dalam chip mikrokontroler tersebut, begitu juga dengan beberapa *Random Access Memory* (RAM) berkapasitas kecil. Mikrokontroler didesain untuk aplikasi terapan, berbeda dengan mikroprosesor yang biasanya digunakan untuk komputer pribadi atau beberapa aplikasi umum yang bisa digunakan untuk berbagai macam.

Mikrokontroler biasanya digunakan untuk produk atau divais yang dikontrol secara otomatis, seperti sistem kontrol pada mesin mobil, peralatan kesehatan yang diimplantasikan, *remote control*, mesin kantor, peralatan, peralatan listrik dan mainan. Dengan memangkas besar dan biaya yang dibutuhkan bila dibandingkan dengan mikroprosesor, memori dan divais masukan/keluaran yang terpisah, membuat mikrokontroler ini lebih ekonomis untuk mengontrol divais dan proses yang terjadi. Mikrokontroler sinyal campuran sudah umum digunakan, yaitu untuk mengintegrasikan komponen analog yang diperlukan pada sistem kontrol elektronik non-digital.

2.1.1 Mikrokontroler Arduino UNO



Gambar 2.1 Skematika diagram Arduino UNO Rev3

Arduino UNO adalah papan berbasis mikrokontroler pada ATmega328 (datasheet). Mikrokontroler ini memiliki 14 digital input / output pin (dimana 6 dapat digunakan sebagai keluaran *Pulse-width Modulation* (PWM)), 6 input analog, 16 MHz osilator kristal, koneksi Universal Serial Bus (USB), *jack* listrik, *In-Circuit Serial Programming* (ICSP) header, dan tombol *reset*. Arduino UNO telah berisi semua yang diperlukan untuk mendukung mikrokontroler, hanya dengan menghubungkannya ke komputer dengan kabel USB atau menggunakan listrik dengan adaptor AC-DC atau baterai untuk menghidupkannya, untuk lebih jelasnya lihat Gambar 2.1.

Arduino Uno berbeda dari semua papan sebelumnya yaitu tidak digunakannya *Future Technology Devices International Ltd. (FTDI) USB-to-serial Chip driver*. Sebaliknya, fitur yang terdapat dalam ATmega8U2 diprogram untuk digunakan sebagai konverter *USB-to-serial*. Revisi 2 dari papan Arduino Uno memiliki resistor

yang menarik garis HWB 8U2 ke tanah, sehingga lebih mudah untuk dimasukkan ke dalam mode *Device Firmware Update* (DFU).

"Uno" berarti satu di Italia dan diberi nama untuk menandai peluncuran Arduino 1,0. Uno dan versi 1.0 akan ada pada versi referensi dari Arduino dan terus berkembang. Uno adalah yang versi terbaru dalam serangkaian papan USB Arduino, dan model referensi untuk platform Arduino.

2.2 Embedded System

Embedded System atau Sistem Tertanam adalah suatu sistem yang menjadi pusat proses eksternal, sensor dan *actuator* serta *human interface*-nya. Sebuah sistem mikroprosesor tertanam biasanya mempunyai beberapa komponen seperti mikroprosesor, RAM, penyimpanan *nonvolatile: erasable programmable read-only memory*(EPROM), ROM, *Flash Memory*, *battrey backed RAM* dan seterusnya serta I/O (beberapa dimaksudkan untuk memonitor atau mengontrol dunia nyata) (Ball, 2000). Suatu *embedded system* haruslah mempunyai karakteristik yang mampu berdiri sendiri sebagai berikut:

1. *Reliability* – Tingkat besaran dari kehandalan dari sistem tersebut, menggunakan $R(t)$ sebagai simbolnya. $R(t)$ itu sendiri adalah probabilitas dari suatu sistem untuk bekerja secara benar saat $t = 0$.
2. *Maintainability* – Tingkat besaran yang digunakan untuk mengukur pemeliharaan dari suatu sistem, menggunakan $M(d)$ sebagai simbolnya. $M(d)$ itu sendiri adalah probabilitas dari suatu sistem untuk bekerja secara benar pada saat waktu d ditunjukkan setelah *error* muncul.
3. *Availability* – Tingkat ketersediaan dari sistem untuk bekerja pada waktu yang telah ditentukan. Kemudian ditentukan probabilitas yang dibutuhkan untuk itu dalam waktu yang dinyatakan dalam t .
4. *Safety* – Tingkat keselamatan yang diakibatkan dari sistem ini. Sistem ini berjalan tanpa ada bahaya yang ditimbulkan bagi yang lain.

5. *Security* – Tingkat keamanan dan keaslian komunikasi dari sistem ini. Tingkat keamanan terletak pada rahasia dari data dan komunikasi yang terjadi tetap sama dan sesuai dengan aslinya.

Selanjutnya adalah *embedded system* haruslah efisien, berikut adalah contoh yang dimaksudkan dalam hal efisiensi:

1. *Energy efficient*
2. *Code-size efficient*
3. *Run-time efficient*
4. *Weight efficient*
5. *Cost efficient*

Suatu *embedded system* (ES) juga harus memilih apakah dia diperuntukan untuk beberapa aplikasi atau untuk tatap muka pengguna. Banyak ES harus mencapai *real-time constraints*. Suatu sistem dikatakan *real-time* bila beraksi untuk menstimulus objek yang dikontrol atau dari operator diantara interval waktu yang diberikan oleh lingkungan. Untuk suatu sistem *real-time* bila jawaban yang benar datang terlalu telat atau terlalu cepat maka dinyatakan telah terjadi kesalahan.

2.3 LED

Sebuah Diode pancaran cahaya (bahasa Inggris: *light-emitting diode*; LED) adalah suatu semikonduktor yang memancarkan cahaya monokromatik yang tidak koheren ketika diberi tegangan maju. Gejala ini termasuk bentuk elektroluminesensi. Warna yang dihasilkan bergantung pada bahan semikonduktor yang dipakai, dan bisa juga ultraviolet dekat atau inframerah dekat.

Sebuah LED adalah jenis dioda semikonduktor istimewa. Seperti sebuah dioda normal, LED terdiri dari sebuah chip bahan semikonduktor yang diisi penuh, atau di-dop, dengan ketidakmurniannya untuk menciptakan sebuah struktur yang disebut *p-n junction*. Pembawa-muatan - elektron dan lubang mengalir ke *junction*

dari elektroda dengan voltase berbeda. Ketika elektron bertemu dengan lubang, dia jatuh ke tingkat energi yang lebih rendah, dan melepaskan energi dalam bentuk foton.

Panjang gelombang dari cahaya yang dipancarkan berbeda oleh karena itu warnanya juga berbeda, tergantung dari selisih pita energi dari bahan yang membentuk p-n junction. Sebuah dioda normal, biasanya terbuat dari silikon atau germanium, memancarkan cahaya tampak inframerah dekat, tetapi bahan yang digunakan untuk sebuah LED memiliki selisih pita energi antara cahaya inframerah dekat, tampak, dan ultraungu dekat.

Tak seperti lampu pijar dan neon, LED mempunyai kecenderungan polarisasi. Chip LED mempunyai kutub positif dan negatif (p-n) dan hanya akan menyala bila diberikan arus maju. Ini dikarenakan LED terbuat dari bahan semikonduktor yang hanya akan mengizinkan arus listrik mengalir ke satu arah dan tidak ke arah sebaliknya. Bila LED diberikan arus terbalik, hanya akan ada sedikit arus yang melewati chip LED. Ini menyebabkan chip LED tidak akan mengeluarkan emisi cahaya. Chip LED pada umumnya mempunyai tegangan rusak yang relatif rendah. Bila diberikan tegangan beberapa volt ke arah terbalik, biasanya sifat isolator searah LED akan jebol dan menyebabkan arus dapat mengalir ke arah sebaliknya.

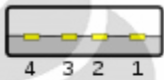

Karakteristik chip LED pada umumnya adalah sama dengan karakteristik dioda yang hanya memerlukan tegangan tertentu untuk dapat beroperasi. Namun bila diberikan tegangan yang terlalu besar, LED akan rusak walaupun tegangan yang diberikan adalah tegangan maju. Tegangan yang diperlukan sebuah dioda untuk dapat beroperasi adalah tegangan maju (V_f).

2.4 USB

USB atau *Universal Serial Bus* adalah suatu spesifikasi untuk membangun komunikasi antara perangkat dan sebuah *host controller* (biasanya komputer pribadi).

USB ini dikembangkan dan diciptakan oleh Ajay Bhatt saat bekerja untuk Intel. USB secara efektif telah menggantikan berbagai antarmuka seperti port serial dan paralel.

USB dapat menghubungkan peralatan komputer seperti *mouse*, *keyboard*, kamera digital, *printer*, *personal media player*, *flash drive*, *Network Adapters*, dan *hard drive eksternal*. USB ini menggunakan 4 pin sebagai alat komunikasi datanya, lihat Gambar 2.2. untuk lebih jelasnya.

USB 1.x/2.0 standard pinning					
Pin	Name	Cable color	Description		
1	VCC	Red	+5 V	4 3 2 1 Type A	1 2 4 3 Type B
2	D ⁻	White	Data -	5 4 3 2 1 Mini-A	5 4 3 2 1 Mini-B
3	D ⁺	Green	Data +		
4	GND	Black	Ground	5 4 3 2 1 Micro-A	5 4 3 2 1 Micro-B

Gambar 2.2 Contoh tipe USB dan spesifikasinya (USB Implementers Forum, 2007)

2.5 JAVA

Java adalah bahasa pemrograman yang awalnya dikembangkan oleh James Gosling di Sun Microsystems (yang sekarang menjadi anak perusahaan dari Oracle Corporation) dan dirilis pada tahun 1995 sebagai komponen inti dari platform Java Sun Microsystems. Bahasa Java berasal dari banyak sintaks seperti C dan C++ namun memiliki model objek sederhana dan lebih sedikit rendah tingkat fasilitasnya. Aplikasi Java biasanya dikompilasi untuk *bytecode* (file kelas) yang dapat berjalan pada *Java Virtual Machine* (JVM) terlepas dari bentuk arsitektur komputer yang dipakai. Java mempunyai tujuan umum untuk bahasa berorientasi objek, berbasis kelas dan yang khusus dirancang untuk memiliki dependensi pelaksanaan sesedikit mungkin. Hal ini dimaksudkan untuk memungkinkan pengembang aplikasi untuk melakukan "tulis sekali, jalankan di mana saja.". Java saat ini merupakan salah satu

bahasa pemrograman paling populer yang digunakan, terutama untuk aplikasi *client-server* berbasis pada jaringan.

Pelaksanaan asli dan referensi *Java compiler*, mesin virtual, dan perpustakaan kelas yang dikembangkan oleh Sun dilakukan mulai dari tahun 1995. Pada Mei 2007, sesuai dengan spesifikasi dari *Java Community Process*, Sun memiliki lisensi sebagian besar teknologi Java yang berada di bawah *GNU's Not UNIX (GNU) General Public License*. Beberapa teknologi lainnya juga telah dikembangkan untuk implementasi alternatif dari Sun, seperti Kompilator GNU untuk Java, GNU Classpath, dan Dalvik.

2.5.1 JavaMail

Application Programming Interface (API) JavaMail menyediakan kerangka kerja platform-independen dan protokol-independen untuk membangun aplikasi pesan dan ratron itu sendiri. API JavaMail tersedia sebagai paket tambahan untuk digunakan dengan platform *Java Standard Edition (SE)* dan juga termasuk dalam platform *Java Enterprise Edition (EE)*. *Command* ini berguna untuk membangun fasilitas ratron atau *mail client* didalam pemrograman java.

2.5.2 Java IMAP IDLE

IMAP IDLE pertama kali diluncurkan pada bulan juni tahun 1997 yang kemudian dimasukan didalam JavaMail versi 1.4.1. IMAP IDLE yang digunakan untuk proyek ini menggunakan JavaMail versi 1.4.4. Perintah IMAP IDLE dapat digunakan jika didukung oleh *server*, untuk masuk ke modus siaga sehingga *server* dapat mengirim pemberitahuan ratron yang masuk tanpa perlu bagi klien untuk terus melakukan *polling* di server. *ConnectionListener* digunakan untuk memberitahu peristiwa yang terjadi. Ketika *thread* lain (misalnya, *Thread Listener*) perlu untuk mengeluarkan perintah IMAP dalam kelas Store ini, modus siaga akan dihentikan dan metode ini akan kembali seperti semula. Biasanya pemanggil akan memanggil metode ini dalam satu lingkaran penuh.

Jika properti *mail.imap.enableimapevents* diatur, pemberitahuan diterima saat perintah IDLE aktif yang kemudian akan dikirimkan ke *ConnectionListeners* sebagai peristiwa dengan jenis *IMAPStore.RESPONSE*. *Event Notifier* akan mempunyai bentuk *string* IMAP sebagai respon yang baku. Perlu diperhatikan bahwa kebanyakan server IMAP tidak memberikan notifikasi bila menggunakan perintah IDLE pada koneksi tanpa adanya kotak surat yang dipilih. Dalam kebanyakan kasus, Anda akan ingin menggunakan metode menganggur atau *idle* di *IMAPFolder*. Kemampuan ini sangat *Highly Experimental* sehingga ada kemungkinan akan berubah dalam rilis-rilis yang mendatang (Oracle, Class *IMAPStore*, 2011).

Properti *mail.imap.minidletime* memaksa penundaan minimum sebelum kembali dari metode ini, untuk memastikan bahwa thread lain memiliki kesempatan untuk mengeluarkan perintah sebelum pemanggil memanggil metode ini lagi. *Delay default* atau penundaan normalnya adalah sekitar 10 milidetik (Leiba, 1997).

2.6 E-mail / Ratron

Surat elektronik, atau yang biasa disebut ratron atau *e-mail*, adalah metode bertukar pesan digital dari penulis untuk satu atau lebih penerima. Ratron modern beroperasi di Internet atau jaringan komputer lainnya. Beberapa sistem ratron awal harus memerlukan adanya kedua penulis dan penerima yang *online* pada saat yang sama, hal ini mirip dengan *instant messaging*. Sistem ratron pada hari ini didasarkan pada model *store-and-forward*. *Server* ratron menerima, mengirim kembali, memberikan dan menyimpan pesan. Baik pengguna maupun komputer mereka diharuskan untuk *online* secara bersamaan, mereka hanya perlu untuk dihubungkan selama sebentar saja, biasanya untuk *server* ratron, selama yang dibutuhkan untuk mengirim atau menerima pesan.

Pesan ratron terdiri dari tiga komponen, *message envelope*, *message header*, dan *body message*. *Message header* berisi informasi kontrol, termasuk didalamnya alamat ratron pencetus dan satu atau lebih alamat penerima. Biasanya informasi

deskriptif juga ditambahkan, seperti kolom *header* untuk subyek dan penyerahan pesan tanggal / waktu cap pengiriman.

Awalnya ratron hanya sebuah teks saja (7-bit *American Standard Code for Information Interchange* (ASCII) dan lain-lain). Namun semakin berkembangnya komunikasi media, ratron diperpanjang untuk membawa lampiran konten multimedia, sebuah standar proses dalam RFC 2045 melalui 2049 kemudian dilampirkan. Secara kolektif, RFC ini pada saat yang telah datang akan disebut *Multipurpose Internet Mail Extensions* (MIME).

Elektronik mail mendahului lahirnya Internet dan merupakan alat penting dalam menciptakan Internet tersebut. Tetapi semakin berkembangnya sejarah, layanan ratron internet secara global kembali kepada *Advanced Research Projects Agency Network* (ARPANET). ARPANET merupakan standar pengkodean pesan ratron yang diusulkan pada awal 1973 (RFC 561). Konversi dari ARPANET ke Internet pada awal 1980-an menghasilkan inti dari layanan internet pada saat ini. Sebuah ratron yang dikirimkan pada awal tahun 1970 terlihat cukup mirip bila dibandingkan dengan pesan teks dasar yang dikirim melalui Internet seperti pada saat ini.

Jaringan berbasis ratron yang pada awalnya dipertukarkan melalui ARPANET di ekstensi *File Transfer Protocol* (FTP), tetapi sekarang dapat dibawa melalui *Simple Mail Transfer Protocol* (SMTP). Hal ini pertama kali diterbitkan sebagai standar Internet 10 (RFC 821) pada tahun 1982. Dalam proses pengangkutan pesan ratron antara sistem, SMTP berkomunikasi melalui parameter pengiriman menggunakan amplop pesan terpisah dari pesan *header* dan tubuh itu sendiri.

2.6.1 E-Mail Server

E-Mail Server adalah tempat dimana ratron itu akan dikirim atau alamat dari penyedia tempat maya di internet untuk menyimpan data pesan ratron tersebut. Contoh *server* ratron seperti Gmail, YahooMail, PlasaMail, dan seterusnya. *Server*

ratron ini akan menkoordinir semua ratron yang masuk ke dalam akun pengguna. Beberapa ratron server ini mampu untuk membedakan antara SPAM dan Ratron yang otentik.

2.6.2 IMAP

Internet Message Access Protocol (IMAP) adalah salah satu dari dua protokol Internet yang paling umum/standar untuk pengambilan ratron, yang lainnya adalah seperti menggunakan *Post Office Protocol* (POP). Hampir semua ratron yang modern baik itu klien atau *mail server* mendukung kedua protokol sebagai sarana untuk mentransfer pesan ratron dari server.

IMAP itu sendiri adalah *Application Layer* didalam protokol Internet yang memungkinkan klien ratron untuk mengakses ratron pada *mail server* secara *remote*. Versi saat ini, yaitu IMAP versi 4 revisi 1 (IMAP4rev1) yang didefinisikan melalui RFC3501. Server IMAP untuk mendengarkan terletak pada *port*/kanal 143.

IMAP mendukung kedua mode operasi *on-line* dan *off-line*. Klien ratron menggunakan IMAP umumnya untuk meninggalkan pesan pada server sampai pengguna secara eksplisit ingin menghapus mereka. Ini adalah satu dari karakteristik lainnya dari operasi IMAP yang memungkinkan beberapa klien untuk mengelola kotak surat yang sama. Kebanyakan klien ratron mendukung IMAP di samping untuk POP untuk mengambil pesan. Namun, untuk saat ini masih sedikit layanan ratron yang mendukung IMAP. IMAP menawarkan akses ke penyimpanan pesan secara langsung. Klien dapat menyimpan beberapa salinan dari pesan yang kemudian akan dianggap sebagai *cache* sementara.

Pengguna mengambil pesan dengan klien ratron yang menggunakan salah satu dari sejumlah *e-mail* protokol pengambilan. Beberapa klien dan server secara *preferentially* menggunakan spesifik dari *vendor*, protokol *proprietary*. Tetapi dukungan untuk protokol Internet standar adalah menggunakan SMTP untuk mengirim e-mail dan POP serta IMAP untuk mengambil ratron, yang memungkinkan

interoperabilitas antara *server* lain dengan klien. Sebagai contoh, klien *Microsoft Outlook* menggunakan protokol khusus untuk berkomunikasi dengan *server Microsoft Exchange Server* seperti halnya klien. Sebagai contoh IBM ketika berkomunikasi dengan *server Domino*, tetapi semua produk ini juga mendukung POP, IMAP, dan SMTP sebagai keluarannya. Dukungan untuk protokol standar Internet memungkinkan banyak klien ratron seperti PegasusMail atau Mozilla Thunderbird dapat digunakan untuk mengakses *server*, dan memungkinkan klien untuk digunakan dengan server lain.

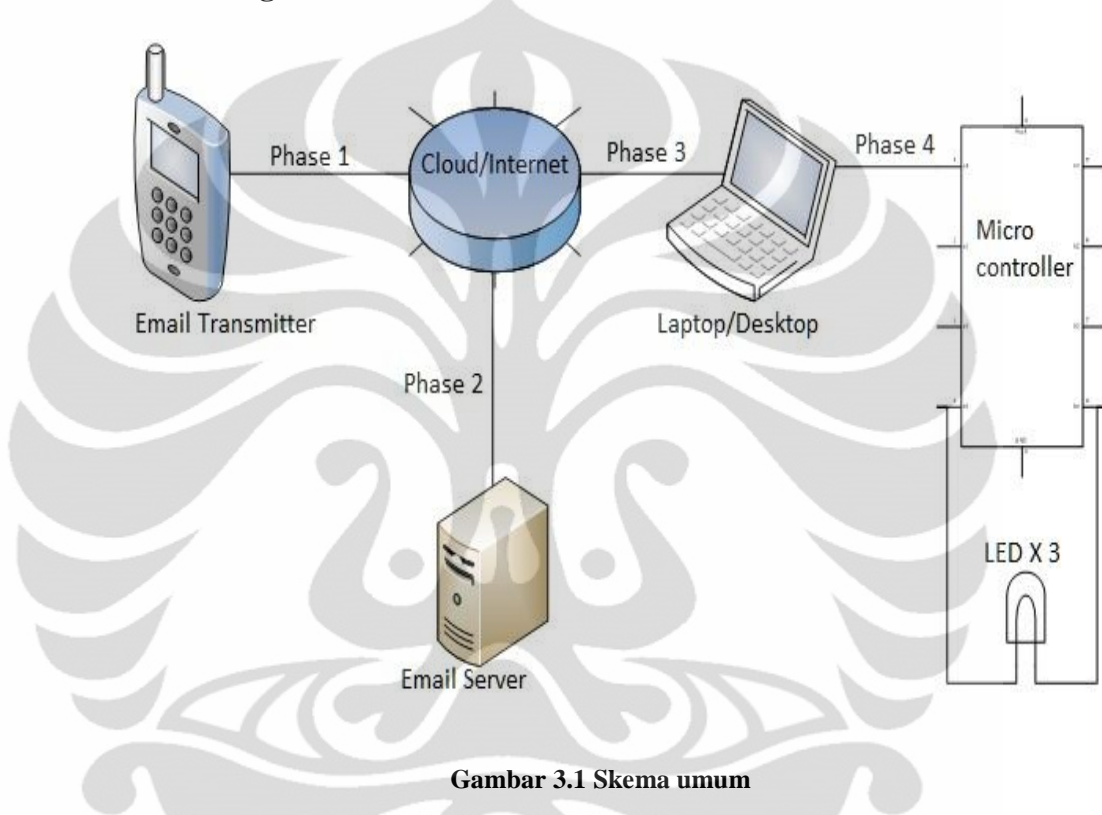
2.6.3 SMTP

Simple Mail Transfer Protocol (SMTP) adalah standar Internet untuk surat elektronik (ratron) yang ditransmisikan melalui Internet Protokol (IP) pada jaringan. SMTP pertama kali didefinisikan oleh RFC 821 (1982, yang pada akhirnya dinyatakan sebagai STD 10) dan terakhir diperbarui oleh RFC 5321 (2008) yang meliputi *Extended SMTP* (ESMTP) dan juga merupakan penambahan protokol yang digunakan secara luas pada saat ini. SMTP sudah ditentukan untuk transportasi surat keluar dengan menggunakan *port Transmission Control Protocol* (TCP) 25. Protokol untuk pengajuan baru secara efektif sama seperti SMTP, tetapi menggunakan *port* 587 sebagai gantinya. Koneksi SMTP diamankan oleh *Secure Socket Layer* (SSL) yang kemudian dikenal dengan singkatan SMTPS, meskipun tidak semua protokol SMTPS berada dalam dirinya sendiri.

Sementara *server* surat elektronik dan agen yang mentransfer surat lainnya menggunakan SMTP untuk mengirim dan menerima pesan ratron, aplikasi klien pada *user-level* yang biasanya hanya menggunakan SMTP untuk mengirimkan pesan ke *server* mail untuk menyampaikan. Untuk menerima pesan, aplikasi-aplikasi client biasanya menggunakan baik POP atau IMAP atau sistem yang berlisensi (seperti Microsoft Exchange atau Lotus Notes / Domino) untuk mengakses akun ratron mereka melalui kotak surat pada sebuah *mail server*.

3 PERANCANGAN SISTEM KENDALI ELEKTRONIK MELALUI RATRON

3.1 Ratron Pengendali Elektronik



Gambar 3.1 Skema umum

Gambar 3.1 ini adalah denah atau diagram dari skema umum untuk *Email Electronic Control* (EEC) atau Ratron Pengendali Elektronik (RPE). Dari Gambar 3.1 tersebut terdapat 4 fase dalam pembuatan proyek kali ini. Setiap fase mempunyai fungsinya masing-masing, seperti fase satu untuk melakukan pengiriman ratron ke Internet. Penjelasan dari masing-masing fase akan dijelaskan dalam subbab tersendiri.

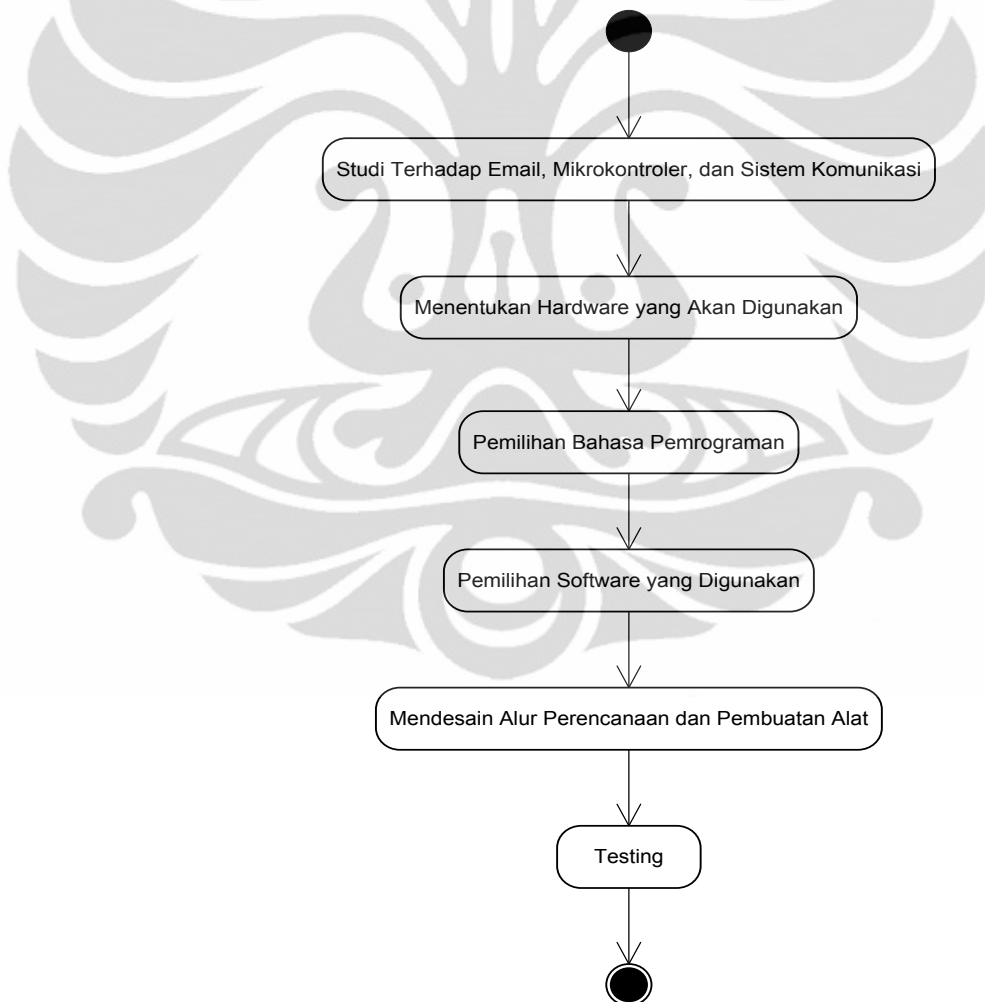


Gambar 3.2 Diagram proses EEC

Gambar 3.2 menjelaskan tentang proses pembuatan pada proyek kali ini. Dimulai dari fase 4 pada tingkatan rendah hingga berakhir pada fase 1. Fase 1 hanya merupakan fasilitas terakhir yang menjadi tatap muka pengguna dengan sistem.

3.2 Fase 1 : Pengiriman Ratron ke Ratron Server

Fase satu memiliki fungsi mengirimkan ratron ke *server* ratron melalui internet. Ini adalah lapisan pertama untuk pengguna pada saat mereka ingin mengirimkan ratron. Ratron itu sendiri berisi perintah kepada perangkat elektronik yang akan dijalankan.



Gambar 3.3 State diagram penelitian

Gambar 3.3 menjelaskan mengenai pembuatan proyek kali ini. Proyek ini dimulai dari studi literatur hingga pembuatan dan kemudian pengujian. Pengujian dilakukan setelah semua fase dibuat dan dimulai dari Fase 4 hingga berakhir pada Fase 1. Pengujian tersebut berada pada Fase 1 dari pembuatan proyek kali ini.

3.2.1 Spesifikasi Fase 1

Fase satu memiliki spesifikasi berupa perangkat elektronik yang nantinya akan digunakan untuk mengirimkan ratron. Perangkat elektronik tersebut harus sudah diberi fasilitas untuk mengirimkan ratron. Contoh untuk perangkat itu misalnya SmartPhone, Laptop, Desktop, dan perangkat elektronik lainnya. Spesifikasi yang lain adalah adanya koneksi ke internet untuk peralatan tersebut. Untuk koneksi internet dapat menggunakan *modem*, *LAN*, *WIFI* atau lainnya yang pastinya mempunyai konektisifitas ke internet.

3.2.2 Cara Kerja Fase 1

Melalui smartphone atau perangkat elektronik tersebut dilakukan pengiriman ratron ke server ratron. Contoh server ratron adalah seperti Gmail, Yahooemail, dll. Namun *server* ratron tersebut harus yang sudah dilengkapi dengan protokol IMAP dan mampu menjalankan Perintah IDLE dari IMAP. Hal ini dikarenakan perintah tersebut digunakan untuk transfer ratron dan notifikasi ratron. Ratron bisa berisi perintah sederhana, sebagai contoh untuk melakukan pematian lampu cukup digunakan “LAMP1 OFF” pada *body text* atau *subject* di dalam badan ratron.

3.3 Fase 2 : Modifikasi Ratron Server

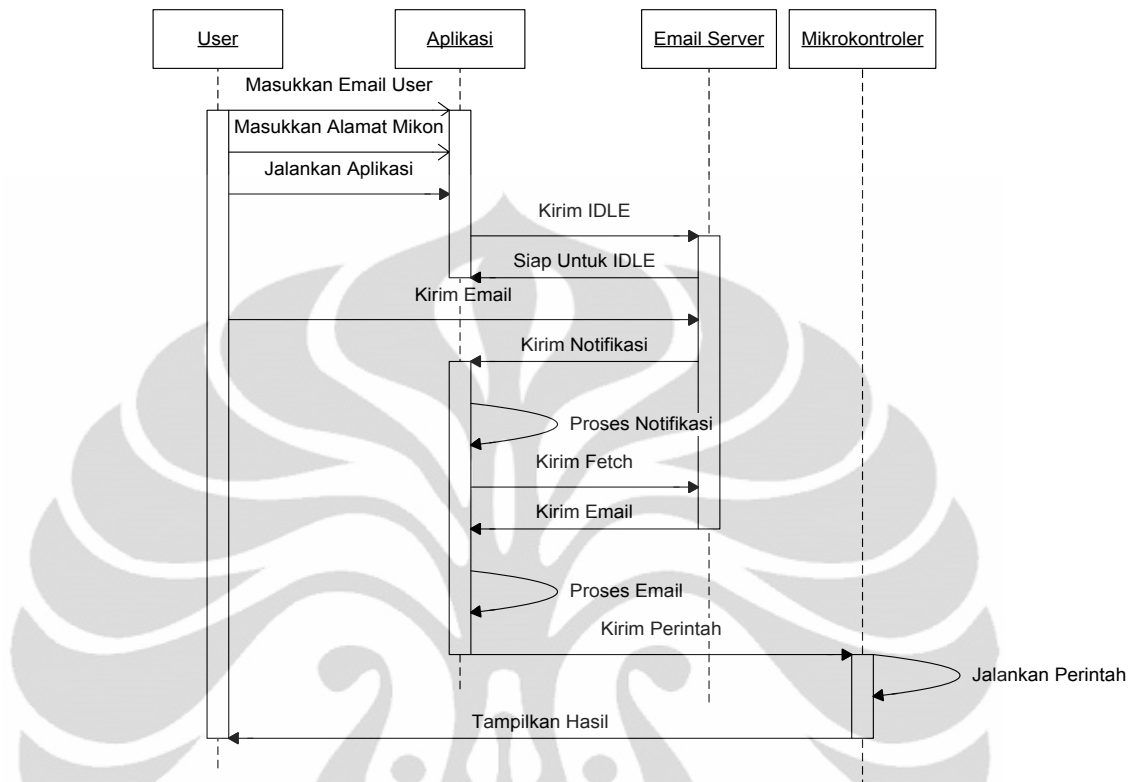
Fase ini memiliki fungsi sebagai tempat penyimpanan sementara dari perintah yang dikirimkan ke internet dan berguna untuk mentransferkannya kembali ke *laptop* atau *desktop* yang berisi program sebagai jembatan antara ratron dan mikrokontroler.

3.3.1 Spesifikasi Fase 2

Pada fase ini memiliki spesifikasi berupa *server* ratron. *Server* tersebut haruslah sudah memiliki fasilitas untuk melakukan IMAP dan IMAP IDLE. IMAP digunakan untuk melakukan pemindaian ratron dan melakukan perubahan langsung pada *server*. Berbeda dengan POP, IMAP dapat menjalankan perintah dua arah, jadi apabila di klien ratron dilakukan penghapusan ratron maka ratron yang terdapat di ratron *server* juga akan dihapus. Untuk proyek ini IMAP sangat berguna untuk melakukan pergantian *FLAG* dari *UNSEEN* menjadi *SEEN*. Sedangkan IMAP IDLE digunakan agar ratron server mengirim notifikasi bila ada ratron yang masuk sehingga klien ratron tidak diharuskan untuk melakukan pooling ratron setiap saat.

3.3.2 Cara Kerja Fase 2

Cara kerja untuk fase ini cukup mudah yaitu pertama server menerima ratron yang berisi perintah dari pengguna dan menyimpannya ke dalam inbox akun ratron server. Kedua adalah server kemudian mengirimkan notifikasi kepada email klien setelah email klien mengajukan perintah IDLE kepada ratron server. Ketiga adalah setelah ratron server mendapatkan perintah untuk mengirimkan salinan ratron ke komputer (*fetch*), server kemudian mengirimkan ratron tersebut ke klien ratron di komputer pengguna.



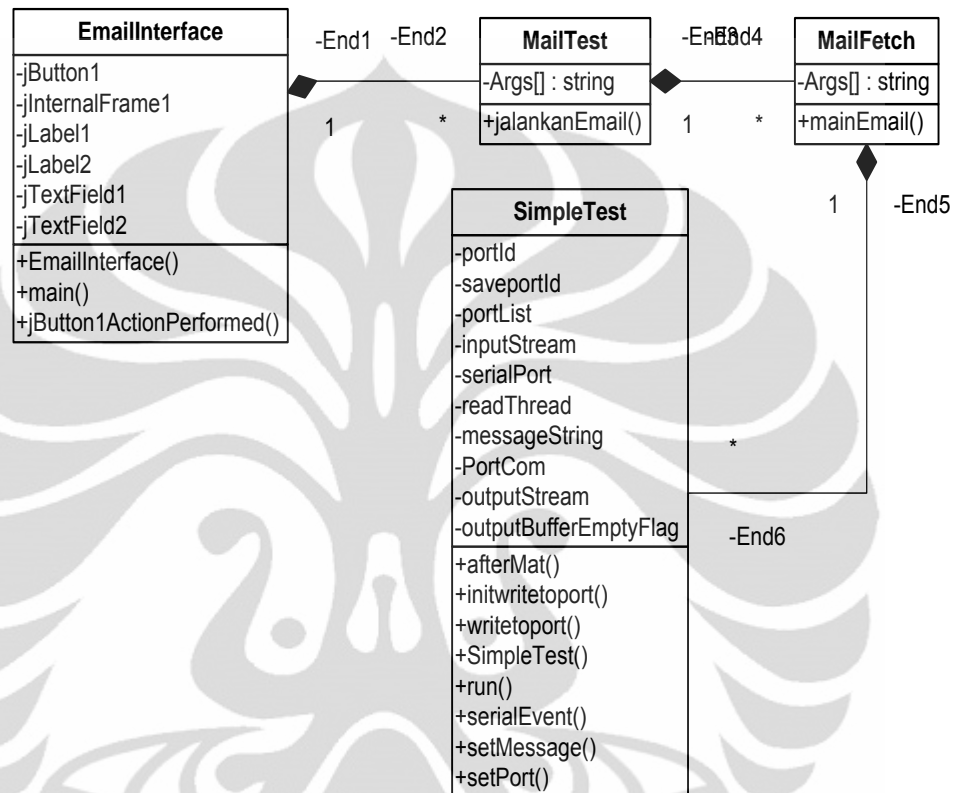
Gambar 3.4 Sequence diagram EEC

Dari Gambar 3.4 dapat dilihat bahwa urutan yang ada lebih banyak terjadi antara aplikasi dan *server* ratron. Pengiriman IDLE dilakukan oleh aplikasi yang kemudian diterima oleh *server* ratron. Kemudian pengguna mulai mengirimkan ratron yang kemudian akan ditandai sebagai ratron baru. *Server* kemudian mengirimkan notifikasi ke aplikasi untuk dilakukan pengambilan ratron. Setelah ratron diterima oleh aplikasi kemudian akan diproses lebih lanjut. Proses selama di dalam *server* inilah yang merupakan bagian dari Fase 2.

3.4 Fase 3 : Penghubung Antara Ratron dan Mikrokontroler

Pada fase ini memiliki fungsi yaitu sebagai jembatan antara ratron server dan mikrokontroler serta sebagai otak operasi dari keseluruhan proyek ini. Fase ini adalah

fase paling krusial diantara fase yang lainnya. Sehingga dibutuhkan perhatian yang cukup besar pada fase ini untuk proses pengerjaannya.



Gambar 3.5 Class diagram fase 3

Gambar 3.5 menggambarkan kelas diagram yang akan digunakan dalam pembuatan program. **EmailInterface** berisi tentang tatap muka antara pengguna dengan aplikasi dan bertugas mengendalikan proses yang terjadi. **MailTest** berfungsi untuk melakukan pengontrolan notifikasi pengguna. **MailFetch** berguna untuk mengambil pengguna baru yang masuk dan mengirimkan data dari pengguna ke **SimpleTest**. Sedangkan **SimpleTest** merupakan tempat pengontrolan mikrokontroler, perintah yang diterima dari **MailFetch** kemudian ditransferkan ke mikrokontroler.

3.4.1 Spesifikasi Fase 3

Pada fase ini dibutuhkan dua spesifikasi yaitu untuk perangkat keras serta perangkat lunaknya. Untuk hardware dibutuhkan *Personal Computer(PC) Desktop* atau *Laptop* yang bisa untuk internet dan memiliki port USB untuk nanti melakukan komunikasi data serial. Berikut adalah spesifikasi PC yang digunakan untuk membangun sistem ini:

1. Processor : AMD Phenom II X4 965 3,4 GHz
2. RAM : 2,00 GB
3. Mother Board : ASUS Crosshair III
4. Hardisk : 600 GB
5. Sound Card : Creative XiFi
6. Operating System : Windows 7 64-bit

Untuk melakukan pemrograman di komputer ini digunakan program NetBeans IDE 6.7. Hal ini terjadi karena untuk bahasa pemrogramannya digunakan bahasa pemrograman JAVA. Java dipilih karena didalam java itu sendiri memiliki fasilitas JavaMail untuk aplikasi ratronnya, kemudian fasilitas IMAP dan IMAP idle. Ditambah dengan adanya RXTXComm yang berguna untuk melakukan interaksi antara program dan mikrokontroler.

3.4.2 Cara Kerja Fase 3

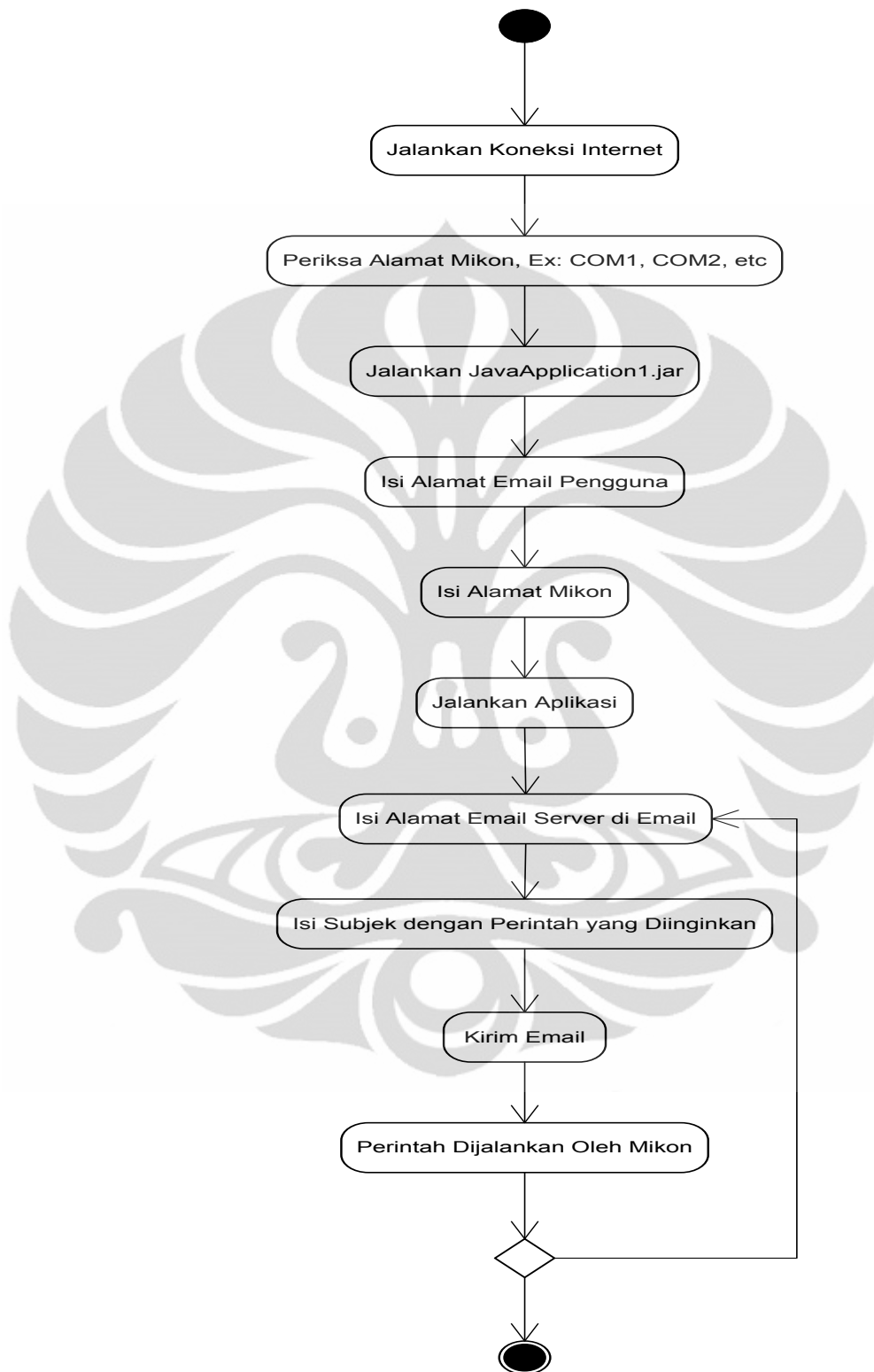
Fase tiga memiliki cara kerja sebagai berikut, pertama adanya fasilitas dari java yaitu IMAP IDLE akan membaca adanya notifikasi ratron baru dari ratron server. Kemudian setelah mengetahui adanya notifikasi baru maka program akan secara otomatis melakukan *fetch* atau mengambil isi dari ratron yang baru masuk tersebut. Kedua adalah setelah ratron diterima kemudian dilakukan pemindaian alamat ratron pengirim dan melakukan pencocokan dengan ratron pengguna yang diinput pada saat pertama kali program ini dijalankan. Bila alamat ratron pengirim sama dengan alamat ratron pengguna maka dilakukan pemindaian kedua yaitu pada

subjek ratron, namun bila tidak sama maka ratron akan diabaikan. Subjek yang berisi perintah dan berupa String akan langsung dimasukkan ke program untuk melakukan komunikasi serial antara komputer dan mikrokontroler. Program pertama kali akan melakukan handshake dengan mikrokontroler untuk mengetahui mikrokontroler itu ada di port yang sudah ditentukan atau tidak. Setelah diketahui mikrokontroler itu ada, maka program akan membuka hubungan data komunikasi serial melalui USB dengan mikrokontroler. Kemudian perintah yang masih berupa String dikonversi menjadi bentuk *buffer byte*, hal ini dilakukan karena mikrokontroler membaca data yang dikirim dalam bentuk *byte*. Setelah dikonversi dilakukan pengiriman data tersebut.

3.5 Fase 4 : Pemrograman Mikrokontroler dan Peralatan Elektronik

Pada fase ini mempunyai fungsi untuk melakukan penerimaan perintah. Perintah tersebut diambil dari komputer dan kemudian dijalankan di mikrokontroler. Fungsi lainnya adalah melakukan pengontrolan alat-alat elektronik yang ada di mikrokontroler. Ini merupakan fase akhir dari proyek kali ini.

Pada Gambar 3.6 dibawah ini dijelaskan mengenai aktifitas yang terjadi oleh pengguna selama pengguna menggunakan sistem ini. Pertama pengguna menjalankan koneksi pada komputer. Setelah itu dilakukan pemeriksaan letak dari mikrokontroler di komputer. Kemudian pengguna menjalankan aplikasi yang telah disediakan. Pengguna lalu memasukkan alamat ratron pengguna dan alamat dari mikrokontroler di komputer. Setelah semua data di isi maka aplikasi ini dapat dijalankan. Melalui peralatan elektronik pengguna, pengguna kemudian mengirimkan ratron. Dalam alamat ratron yang dituju dimasukkan alamat *server* ratron dari sistem ini, kemudian diisikan juga perintah yang diinginkan ke dalam subjek. Lalu ratron dapat dikirimkan setelah data yang harus dimasukkan sudah terisi. Perintah kemudian dapat di eksekusi oleh program baik itu di aplikasi atau di mikrokontroler itu sendiri. Pengguna dapat melihat langsung akibat dari pengiriman langsung pada mikrokontrolernya.



Gambar 3.6 Aktifitas diagram

3.5.1 Spesifikasi Fase 4

Fase empat memiliki spesifikasi sebagai berikut, pertama adalah satu buah Sistem Minimum yang menggunakan Arduino Board UNO yang merupakan sebuah sistem minimum dengan lisensi *Open Source Electronic*. Kedua adalah sebuah kabel komunikasi serial USB dengan tipe A dan tipe B. Tipe A digunakan untuk port pada komputer, sedangkan tipe B digunakan untuk port pada sistem minimum Arduino UNO, untuk lebih jelasnya dapat melihat pada Gambar 3.7. Ketiga adalah lampu LED dan resistor 220 ohm. LED digunakan sebagai perwakilan alat-alat elektronik yang dalam fase mendatang dapat ditingkatkan atau lebih kompleks lagi. Kemudian resistor digunakan untuk menahan arus agar LED tidak cepat untuk putus atau rusak.

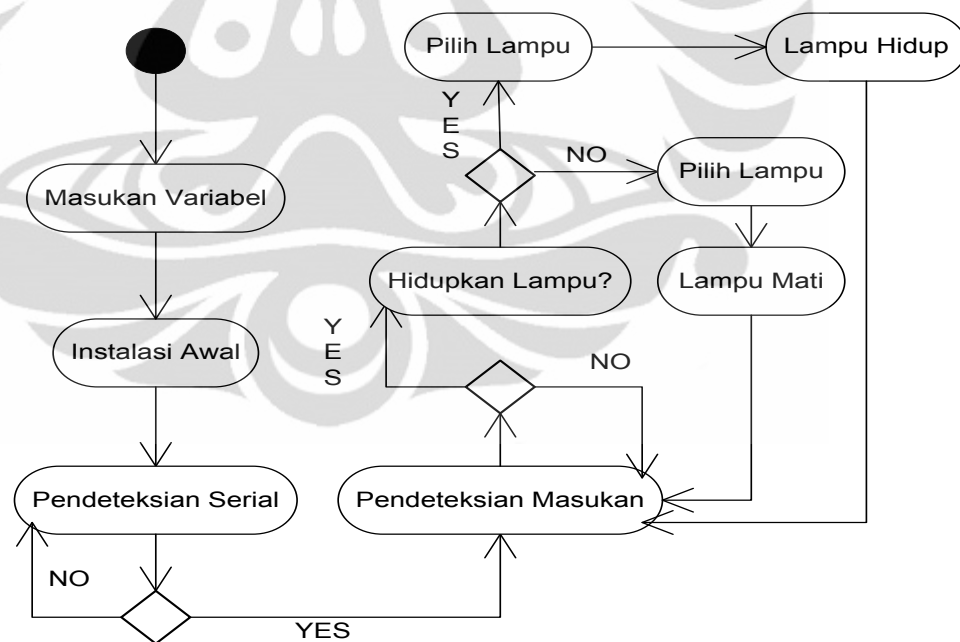


Gambar 3.7 Kabel USB dengan tipe A dan B

Software untuk pembangunan program pada fase ini menggunakan bahasa C, karena sistem minimum untuk Arduino sudah bisa untuk menggunakan bahasa tersebut. Serta untuk kompilasi dan melakukan penulisan di mikrokontroler menggunakan aplikasi bawaan langsung dari Arduino yang menggunakan bahasa JAVA juga.

3.5.2 Cara Kerja Fase 4

Cara kerja pada fase ini tidak terlalu rumit. Pertama, mikrokontroler akan melakukan penyesuaian atau *setup* pertama pada saat mikrokontroler dihidupkan. Kedua, mikrokontroler akan menunggu datangnya byte yang masuk melalui komunikasi serial dengan melakukan *Looping* secara berkala. Bila terdapat *byte* yang masuk maka dilakukan pemindaian byte yang masuk dan dilakukan pencocokan dengan perintah yang ada didalam mikrokontroler. Bila byte yang masuk tidak sesuai dengan yang diinginkan maka mikrokontroler akan kembali melakukan looping untuk menunggu ada *byte* yang masuk atau tidak. Bila *byte* yang masuk sesuai dengan perintah yang ada didalam mikrokontroler maka mikrokontroler akan menjalankan perintah tersebut. Perintah didalam mikrokontroler berisi LED yang mana yang akan dihidupkan atau dimatikan. Untuk lebih jelasnya dapat mengenai cara kerja mikrokontroler dapat dilihat pada Gambar 3.8.



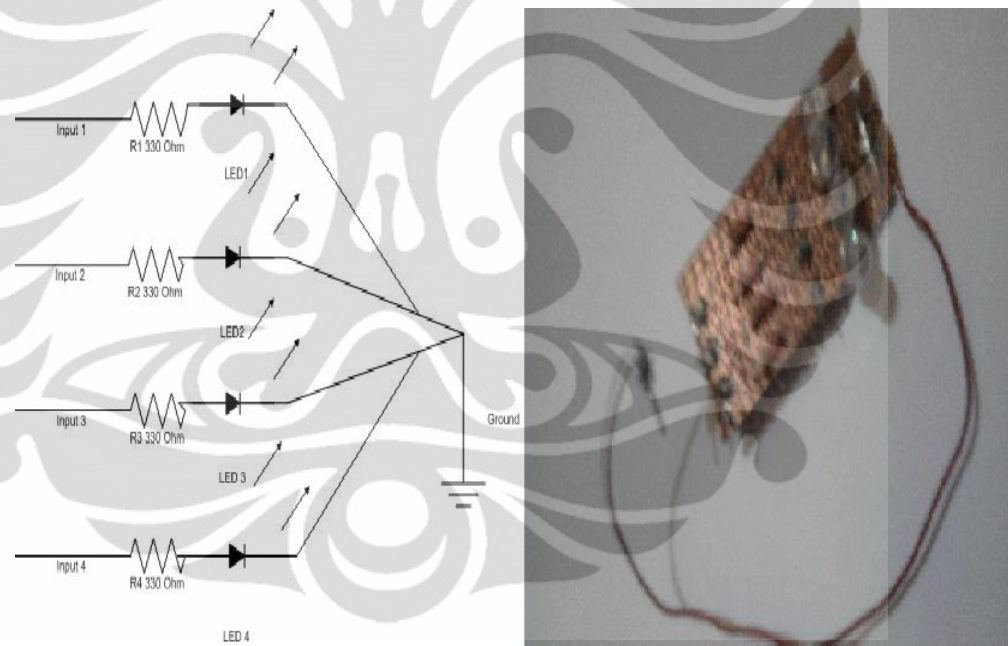
Gambar 3.8 Algoritma cara kerja Mikrokontroler

4 IMPLEMENTASI DAN ANALISIS SISTEM KENDALI ELEKTRONIKA MELALUI RATRON

4.1 Implementasi Fase 4

4.1.1 Perangkat Keras

Untuk pembuatan perangkat keras hal yang pertama dibuat adalah pengendali untuk LED. LED yang digunakan dalam proyek kali ini terdapat empat buah yang kemudian akan disusun dalam rangkaian. Gambar 4.1 menggambarkan skema rangkaian dan bentuk dari rangkaian secara nyata.



Gambar 4.1 Skema dan bentuk asli rangkaian LED

Resistor pada Gambar 4.1 digunakan sebagai penghambat arus agar LED tidak cepat untuk putus. Terdapat empat masukan untuk masing-masing LED, yang setiap masukan akan dikontrol melalui mikrokontroler. LED1 dinyatakan sebagai “R”, LED2 sebagai “G”, LED3 sebagai “B”, dan LED4 dinyatakan sebagai “Y”.

Pernyataan itu digunakan nantinya dalam pemrograman dan pengontrolan LED oleh pengguna.

Setelah rangkaian pengendali LED sudah selesai dikerjakan kemudian rangkaian tersebut digabungkan dengan Sistem Minimum(SisMin) Arduino UNO. Input1 dimasukan kedalam port2, Input2 ke dalam port4, Input3 kedalam port6, dan terakhir Input4 dimasukan dalam port7. Setelah semua masukan ditaruh diport yang telah ditetapkan, masukan kabel *Ground* ke port GND pada SisMin, untuk skematika diagram dapat melihat pada Gambar 2.1. Terakhir adalah menyambungkan antara SisMin dengan komputer melalui kabel USB. Bentuk nyata dari konfigurasi ini dapat dilihat pada Gambar 4.2.



Gambar 4.2 Gambar nyata rangkaian SisMin, LED dan kabel USB

4.1.2 Perangkat Lunak

Untuk pembuatan dari perangkat lunak yang disematkan didalam mikrokontroler digunakan bahasa pemrograman C. Pemrograman menggunakan aplikasi yang telah disediakan oleh Arduino dalam setiap kemasannya. Pemrograman didasarkan pada Gambar 3.8 yang memuat algoritma dari pemrograman mikrokontroler ini. Hal pertama dilakukan adalah melakukan inialisasi variabel untuk masing LED dan menentukann *buffer*. LED1 berada pada port 2 dengan variabel RedPin. Begitu juga dengan LED yang lainnya untuk lebih jelasnya dapat melihat pada Gambar 4.3.

```
char buffer[18];
int RedPin = 2;
int GreenPin = 4;
int YellowPin = 6;
int BluePin = 7;
```

Gambar 4.3 Inialisasi variabel awal

Selanjutnya adalah melakukan pengaturan awal untuk komunikasi serial. Pengaturan awal adalah menetapkan *bitrate*, dalam hal ini ditetapkan sebesar 9600. Kemudian melakukan *flush* untuk membersihkan kanal dalam komunikasi serial. Selanjutnya menetapkan masing variabel LED menjadi *OUTPUT* atau keluaran. Untuk lebih jelasnya dapat melihat pada Gambar 4.4.

```
void setup()
{
  Serial.begin(9600);
  Serial.flush();
  pinMode(RedPin, OUTPUT);
  pinMode(YellowPin, OUTPUT);
  pinMode(BluePin, OUTPUT);
  pinMode(GreenPin, OUTPUT);
}
```

Gambar 4.4 Pengaturan komunikasi serial

Pada Gambar 4.5 diperlihatkan kode untuk melakukan deteksi adanya komunikasi serial. Kemudian banyaknya masukan dari komunikasi serial dimasukkan kedalam numChar. Setelah itu melalui Serial.read masukan yang berisi perintah dimasukkan kedalam *array buffer* untuk nantinya dilakukan pemisahan antara karakter dari perintah yang dimasukkan. Bila perintah yang dimasukkan lebih dari 15 karakter maka perintah tersebut akan secara otomatis dirubah hanya untuk 15 karakter.

```
void loop()
{
  if (Serial.available() > 0){
    int index =0;
    delay(100);
    int numChar = Serial.available();
    if (numChar>15) {
      numChar = 15;
    }
    while (numChar-->0) {
      buffer[index++] = Serial.read();
    }
    splitString(buffer);
  }
}
```

Gambar 4.5 Identifikasi adanya data yang masuk

Gambar 4.6 menjelaskan tentang kode untuk memasukkan data untuk dipisahkan menjadi per karakter yang kemudian dimasukkan dalam pengaturan LED. Bila parameter yakni karakter huruf ada maka pengaturan LED dapat dilakukan. Setelah proses selesai, data kemudian dikosongkan dan kanal serial dibersihkan.

```

void splitString(char* data) {
  Serial.print("Data entered: ");
  Serial.println(data);
  char* parameter;
  parameter = strtok (data, " ,");
  while (parameter != NULL){
    setLED(parameter);
    parameter = strtok (NULL, " ,");
  }

  for (int x=0; x<16; x++){
    buffer[x]='\0';
  }
  Serial.flush();
}

```

Gambar 4.6 Pemisahan karakter dalam data yang masuk

Untuk melakukan pengaturan LED dilakukan dengan melakukan pemindaian pada karakter huruf pertama dari perintah yang dikirimkan. Huruf yang dikirim menandakan LED mana yang harus dihidupkan atau untuk dimatikan. Huruf yang digunakan adalah R (LED1), G (LED2), Y (LED3) dan B (LED4). Lalu *array* dari data yang masuk kemudian ditambah satu agar data yang tersisa hanya isi dari perintah. Untuk mematikan lampu data yang dimasukkan adalah "0". Kemudian untuk menghidupkan adalah menggunakan "255". Untuk lebih jelasnya dapat melihat Gambar 4.7 untuk identifikasi LED dan kontrol fungsinya.

```

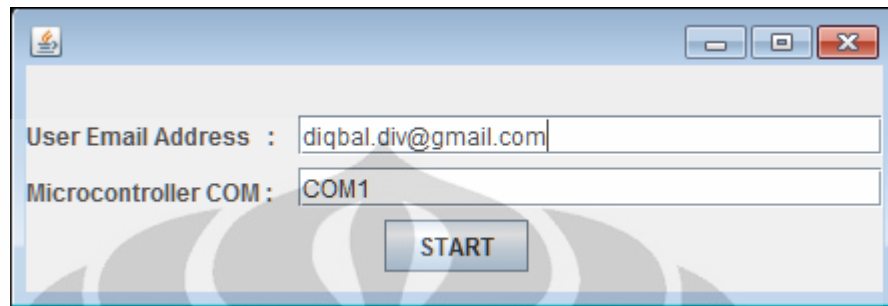
void setLED(char* data) {
  if ((data[0] == 'r') || (data[0] == 'R')){
    int Ans = strtol(data+1, NULL, 10);
    Ans = constrain(Ans, 0, 255);
    analogWrite(RedPin, Ans);
    Serial.print("Red is set to: ");
    Serial.println(Ans);
  }
  .
  .
}

```

Gambar 4.7 Identifikasi LED dan kontrol fungsinya

4.2 Implementasi Fase 3

4.2.1 Tatap Muka Aplikasi



Gambar 4.8 Bentuk tatap muka aplikasi

Gambar 4.8 menggambarkan bentuk dari sistem tatap muka antara pengguna dan aplikasi. Secara *default* alamat pengguna adalah “diqbal.div@gmail.com” sebagai contoh dari bentuk pengguna yang ingin digunakan. Kemudian untuk COM dari mikrokontroler adalah COM1 sebagai *default*-nya. Tombol “START” digunakan untuk menjalankan aplikasi ini.

Pengkodean dalam EmailInterface.java mempunyai bentuk yang cukup umum digunakan dalam pembangunan tatap muka menggunakan bahasa pemrograman java. Perbedaan terletak pada penggunaan `actionEvent` yang pengkodeannya disesuaikan dengan penggunaannya. Gambar 4.9 menggambarkan mengenai pengaktifan program bila tombol “START” ditekan. Isi didalam `jTextField1` dan `jTextField2` diambil untuk kemudian dimasukkan dalam `args[]`. Setelah itu `args` ditambahkan ke dalam `MailTest.java` sebagai masukan. `MailTest` digunakan sebagai pengindai adanya ratron baru atau tidak.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String args[] = {jTextField1.getText(), jTextField2.getText()};
    //Mengambil text yang terdapat di jTextField1 dan 2 dan menyimpannya ke args[]
    MailTest.jalankanEmail(args); //Menjalankan MailTest
}
```

Gambar 4.9 Kode untuk pengaktifan program

4.2.2 Notifikasi Ratron

Untuk melakukan notifikasi ratron digunakan MailTest.java sebagai bagian pengendalinya. Hal pertama yang dilakukan untuk mengaktifkan notifikasi ratron adalah dengan membangun hubungan terlebih dahulu antara *server* ratron dengan komputer yang menggunakan aplikasi ini. Kode yang digunakan untuk membangun hubungan ini dapat dilihat pada Gambar 4.10. Setelah hubungan dapat disambungkan maka hal selanjutnya adalah memilih *folder* “Inbox” didalam *server* ratron.

```
public static void jalankanEmail(final String[] args) {
    Properties props = System.getProperties();
    props.setProperty("mail.store.protocol", "imaps");

    try {
        Session session = Session.getDefaultInstance(props, null);
        session.setDebug(true);
        Store store = session.getStore("imaps");
        store.connect("imap.gmail.com", "emailcontrolserver@gmail.com", "emailcontrol");
        IMAPFolder inbox = (IMAPFolder) store.getFolder("Inbox");
        inbox.addMessageCountListener(new MessageCountListener() {
```

Gambar 4.10 Kode untuk membangun hubungan *server* ratron dan komputer

Hal selanjutnya adalah menambahkan MessageCountListener untuk mengetahui jika ada ratron yang masuk. Di dalam MessageCountListener ditambahkan pengendali jika ada pesan yang bertambah atau jika ada pesan yang dihapus. Untuk mengetahui adanya pesan masuk atau tidak ditambahkan juga perintah IDLE, untuk kodenya dapat dilihat pada Gambar 4.11. Tempat untuk melakukan pengawasan yang dilakukan dalam ketentuan READ_ONLY agar mencegah terjadinya perubahan yang akan mengganggu jalannya aplikasi.

```
inbox.open(Folder.READ_ONLY);
inbox.idle();
```

Gambar 4.11 Kode untuk perintah IDLE

Setelah perintah idle jalan maka *server* ratron akan mulai melakukan pengawasan ratron yang masuk. Indikasi adanya ratron baru yang masuk atau

terhapus dikendalikan oleh `messageCountEvent`. Jika terdapat ratron baru maka secara otomatis kode dalam Gambar 4.12 akan melakukan pemrosesan. Setelah proses yang terjadi selesai, hal selanjutnya yang dilakukan adalah melakukan pengambilan ratron melalui `MailFetch.java`.

```

public void messagesAdded(MessageCountEvent messageCountEvent) {
    Message[] messages = messageCountEvent.getMessageCountEvent().getMessages();
    for (Message message : messages) {
        System.out.println(message);
        try {
            MailFetch.mainEmail(args);
        } catch (IOException ex) {
            Logger.getLogger(MailTest.class.getName()).log(
                Level.SEVERE, null, ex);
        }
    }
}

```

Gambar 4.12 Kode untuk mengaktifkan pengambilan ratron baru

4.2.3 Pengambilan Ratron

Untuk pengambilan ratron baru yang masuk tugas sepenuhnya dipegang oleh `MailFetch.java`. Sistem pengkodeannya mirip seperti `MailTest.java` dengan beberapa perbedaan. Salah satu perbedaannya adalah *folder* yang digunakan, yaitu menggunakan `READ_WRITE` karena akan ada beberapa hal yang akan dirubah didalam *server* ratron. Kemudian dilakukan penyetelan untuk melakukan pembacaan pengguna yang masuk dengan menggunakan *buffer*. Untuk lebih jelasnya dapat melihat Gambar 4.13 mengenai *buffer* dan pencarian pengguna baru.

```

Folder inbox = store.getFolder("Inbox");
inbox.open(Folder.READ_WRITE);

BufferedReader reader = new BufferedReader (
new InputStreamReader(System.in));

Message[] message = inbox.search(
new FlagTerm(new Flags(Flags.Flag.SEEN), false));

InternetAddress mess = new InternetAddress(args[0]);

```

Gambar 4.13 Kode *buffer* dan pencarian ratron baru

Pencarian dilakukan dengan parameter FLAG. Ratron yang memiliki FLAG.SEEN dalam nilai salah (UNSEEN) akan dimasukkan ke dalam variabel message. Selanjutnya yang dilakukan adalah memasukkan alamat ratron pengguna yang sudah diberikan pada saat pertama kali. Alamat ratron pengguna tersebut kemudian akan dimasukkan ke dalam variabel mess untuk nantinya dicocokkan dengan pengguna baru yang masuk.

```

for (int i=0, n=message.length; i<n; i++) {
System.out.println(i + ": " + message[i].getFrom()[0]
+ "\t" + message[i].getSubject() + "D=" + message[i].getContentType());
}

```

Gambar 4.14 Pengurutan ratron yang masuk

Gambar 4.14 melakukan pengurutan ratron yang masuk. Ratron yang masuk kemudian ditampilkan ke sistem agar dapat dibaca. Terdapat tiga informasi yang diberikan yaitu alamat pengirim menggunakan `getFrom()`, subjek ratron menggunakan `getSubject()` dan terakhir tipe isi dari ratron yang menggunakan `getContentType()`. Setelah itu dilakukan identifikasi alamat pengirim mencocokkan antara `mess` dengan `message[i].getFrom()[0]`. Jika berhasil maka sistem akan menampilkan kata "SUKSES", jika tidak maka sistem akan kembali ke awal. Hal itulah yang digambarkan oleh Gambar 4.15.


```

if (message[i].getFrom()[0].equals(mess))
{
    System.out.print("SUKSES");
    message[i].setFlag(Flags.Flag.SEEN, true);

    String[] Args = {args[1], message[i].getSubject() };

    SimpleTest.afterMat(Args);
}

```

Gambar 4.15 Identifikasi pengirim dan pengaktifan komunikasi serial

Hal selanjutnya yang dilakukan adalah merubah FLAG ratron dalam *server* ratron dari UNSEEN menjadi SEEN dengan mengganti nilai dari FLAG menjadi *true*. Alamat dari mikrokontroler dinyatakan oleh variabel `args[1]` yang kemudian dimasukkan kedalam variabel `Args` bersama dengan isi dari subjek ratron. Setelah itu komunikasi serial diaktifkan dengan mengirim kedua variabel itu kedalam `SimpleTest.java`.

4.2.4 Komunikasi Data Serial

`SimpleTest.java` melakukan pengendalian komunikasi serial dalam aplikasi kali ini. Hal pertama yang dilakukan adalah melakukan penetapan awal alamat dari mikrokontroler. Kemudian melakukan identifikasi alamat mikrokontroler yang tercatat dengan yang sudah ditentukan. Jika identifikasi berhasil maka dijalankan proses komunikasi serialnya. Untuk kode mengenai hal itu dapat melihat pada Gambar 4.16.

```

// parse ports and if the default port is found, initialized the reader
portList = CommPortIdentifier.getPortIdentifiers();
while (portList.hasMoreElements()) {
    portId = (CommPortIdentifier) portList.nextElement();
    if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
        if (portId.getName().equals(defaultPort)) {
            System.out.println("Found port: "+defaultPort);
            portFound = true;
            // init reader thread
            SimpleTest.reader = new SimpleTest();
        }
    }
}

```

Gambar 4.16 Identifikasi alamat mikrokontroler

Didalam reader dilakukan inialisasi komunikasi serial. Inialisasi yang dilakukan meliputi penamaan untuk komunikasinya, penambahan eventListener dan yang terakhir memasukkan arus input komunikasi data kedalam inputStream. Selanjutnya adalah melakukan aktifasi bila ada data yang masuk. Kode dari proses ini dapat dilihat pada Gambar 4.17.

```
// initialize serial port
try {
    serialPort = (SerialPort) portId.open("SimpleReadApp", 2000);
} catch (PortInUseException e) {}

try {
    inputStream = serialPort.getInputStream();
} catch (IOException e) {}

try {
    serialPort.addEventListener(this);
} catch (TooManyListenersException e) {}

// activate the DATA_AVAILABLE notifier
serialPort.notifyOnDataAvailable(true);
```

Gambar 4.17 Inialisasi kanal serial dan aktifasi bila ada data

Setelah inialisasi berhasil dilakukan hal selanjutnya dilakukan adalah penetapan parameter untuk komunikasi serial. Parameter yang ditetapkan adalah sebagai berikut *bitrate* 9600, *DATABITS_8*, *STOPBITS_1*, dan *PARITY_NONE*. Parameter yang ditetapkan harus sesuai dengan parameter dari mikrokontroler itu sendiri. Setelah paramameter selesai ditetapkan hal selanjutnya adalah melakukan pembacaan *thread*.

```

try {
    // set port parameters
    serialPort.setSerialPortParams(9600, SerialPort.DATABITS_8,
        SerialPort.STOPBITS_1,
        SerialPort.PARITY_NONE);
} catch (UnsupportedCommOperationException e) {}

// start the read thread
readThread = new Thread(this);
readThread.start();

```

Gambar 4.18 Penetapan parameter dan pengaktifasian *thread*

Pada saat *thread* berjalan terdapat dua perintah yang dilakukan. Pertama adalah `initwritetoport()` yang digunakan untuk mendapatkan arus keluaran, Gambar 4.19. Dari sana kemudian dilakukan penulisan subjek dari pengguna yang telah dimasukan kedalam `messageString`. Pesan yang ada masih dalam bentuk *string* sehingga perlu dikonversikan kedalam *byte* agar perintah dapat dibaca oleh mikrokontroler. Setelah dirubah dalam bentuk *byte* barulah kemudian dilakukan penulisan kedalam mikrokontroler. Kode untuk itu dapat dilihat pada Gambar 4.20 yang menggambarkan tentang kode yang telah selesai.

```

public void initwritetoport() {
    // initwritetoport() assumes that the port has already been opened and
    // initialized by "public nulltest()"

    try {
        // get the outputstream
        outputStream = serialPort.getOutputStream();
    } catch (IOException e) {}
}

```

Gambar 4.19 Kode `initwritetoport()`

```

public void writetoport() {
    System.out.println("Writing \""+messageString+"\" to "+serialPort.getName());
    try {
        // write string to serial port
        outputStream.write(messageString.getBytes());
    } catch (IOException e) {}
}

```

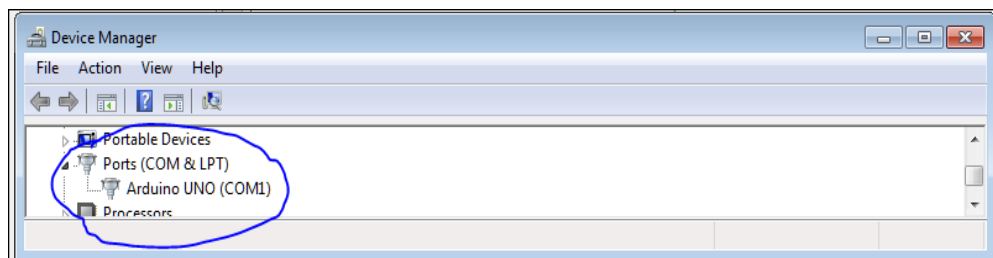
Gambar 4.20 Kode writetoport()

4.3 Pengujian

Pengujian pada alat ini dilakukan sebanyak enam kali. Tiga kali dilakukan menggunakan *desktop* dengan dua kali melakukan *debug*, dan sisanya dilakukan secara normal. Tiga kali yang terakhir dilakukan menggunakan *laptop* tanpa *debug*. Semua pengujian berjalan lancar tanpa ada masalah yang cukup berarti.

Proses pengujian yang dilakukan pertama adalah melakukan penghubungan antara komputer dan internet. Koneksi dapat menggunakan modem dengan pilihan penyedia internet yang diinginkan pengguna. Dalam pengujian kali ini digunakan Telkomsel sebagai penyedia jasa internet.

Selanjutnya adalah menjalankan aplikasi `JavaApplication1.jar` sebagai tatap muka antara pengguna, lihat Gambar 4.8. Selanjutnya memasukkan data yang diperlukan dalam hal ini adalah alamat ratron pengguna dan alamat mikrokontroler dalam komputer, lihat Gambar 4.21 untuk mengetahui letak alamat dari mikrokontroler.



Gambar 4.21 Letak alamat mikrokontroler dalam Windows Operating System

Setelah data selesai dimasukkan kemudian klik tombol “START” untuk menjalankan aplikasi ini. Program akan berjalan dan mulai melakukan pembangunan hubungan antara komputer dengan *server* ratron. Jika dalam *debug* proses yang terjadi selama pembangunan koneksi dapat dilihat langsung, seperti contohnya dalam Gambar 4.22.

```
Profiler Agent: waiting for connection on port 5140, timeout
10 seconds (Protocol version: 9)
Profiler Agent: Established local connection with the tool
DEBUG: setDebug: JavaMail version 1.4.4
DEBUG: getProvider() returning javax.mail.Provider
[STORE,imaps,com.sun.mail.imap.IMAPSSLStore,Sun
Microsystems, Inc]
DEBUG: mail.imap.fetchsize: 16384
DEBUG: mail.imap.statuscachetimeout: 1000
DEBUG: mail.imap.appendbuffersize: -1
DEBUG: mail.imap.minidletime: 10
DEBUG: trying to connect to host "imap.gmail.com", port 993,
isSSL true
* OK Gimap ready for requests from 182.5.139.131
i10if2673752pbf.78
```

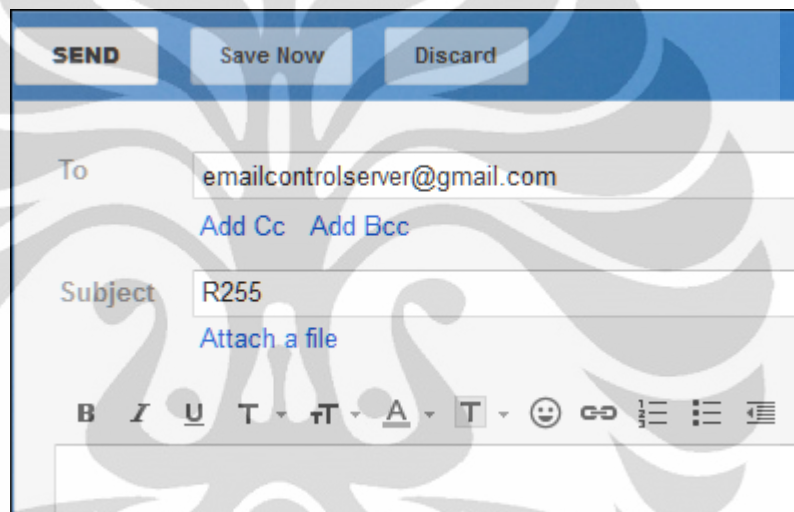
Gambar 4.22 Hasil *debug* untuk koneksi *server* dan komputer

Jika koneksi dapat dibangun hal selanjutnya dilakukan oleh program ini adalah melakukan pembacaan data didalam *server* ratron. Setelah pembacaan selesai maka dijalankan perintah IDLE. Pada saat perintah IDLE dijalankan *debug* dari program akan berbentuk seperti pada Gambar 4.23. Jika sudah mendapatkan +idling maka program telah siap untuk menerima pengguna baru yang akan masuk.

```
* CAPABILITY IMAP4rev1 UNSELECT IDLE NAMESPACE QUOTA ID
XLIST CHILDREN X-GM-EXT-1 UIDPLUS COMPRESS=DEFLATE
A2 OK Success
DEBUG: connection available -- size: 1
A3 EXAMINE Inbox
* FLAGS (\Answered \Flagged \Draft \Deleted \Seen)
* OK [PERMANENTFLAGS ()] Flags permitted.
* OK [UIDVALIDITY 659900942] UIDs valid.
* 23 EXISTS
* 0 RECENT
* OK [UIDNEXT 66] Predicted next UID.
A3 OK [READ-ONLY] Inbox selected. (Success)
A4 IDLE
+ idling
```

Gambar 4.23 Hasil *debug* perintah IDLE

Pengguna kemudian dapat mengirimkan ratron yang berisi perintah ke mikrokontroler. Struktur pengiriman ratron berisi perintah adalah alamat ratron tujuan adalah “emailcontrolserver@gmail.com”. Kemudian perintah dimasukkan kedalam isi dari subjek, lihat Gambar 4.24 untuk contoh dari pengiriman ratron. Perintah-perintah yang didukung dalam sistem ini dapat terlihat pada Tabel 4.1. Pengiriman ratron dapat menggunakan PC atau telepon selular yang dilengkapi dengan fasilitas pengiriman ratron.



Gambar 4.24 Contoh pengiriman ratron

Tabel 4.1 List perintah dalam sistem ini dan fungsinya

Lampu Tujuan	Mati	Hidup
LED 1	R0 / r0	R255 / r255
LED 2	G0 / g0	G255 / g255
LED 3	Y0 / b0	Y255 / b255
LED 4	B0 / y0	B255 / y255

Setelah pengguna mengirim ratron tersebut, maka *server* akan mulai melakukan penerimaan ratron-ratron yang masuk. Setelah ratron masuk, kemudian *server* mengirimkan notifikasi ke komputer. Selanjutnya adalah komputer mengambil

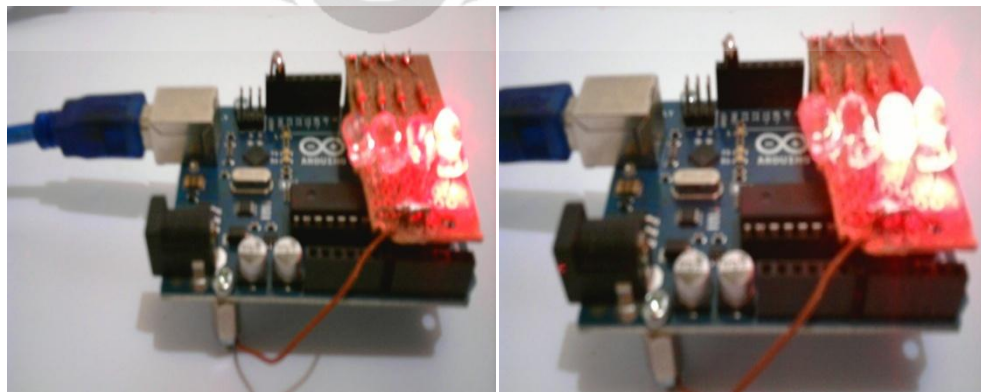
ratron dan mengirimkannya ke mikrokontroler secara komunikasi serial. Dalam *debug* proses yang terjadi akan berbentuk seperti Gambar 4.25.

```
A5 OK Success
A6 FETCH 24 (BODYSTRUCTURE)
* 24 FETCH (BODYSTRUCTURE (("TEXT" "PLAIN" ("CHARSET" "ISO-
8859-1") NIL NIL "7BIT" 2 1 NIL NIL NIL)("TEXT" "HTML"
("CHARSET" "ISO-8859-1") NIL NIL "7BIT" 6 1 NIL NIL NIL)
"ALTERNATIVE" ("BOUNDARY" "0016367b62dccd4e6604b11e0942")
NIL NIL))
A6 OK Success
0: Dean Iqbal Divanda <diqbal.div@gmail.com>
r255D=multipart/ALTERNATIVE;
boundary=0016367b62dccd4e6604b11e0942
SUKSES A7 STORE 24 +FLAGS (\Seen)
* 24 FETCH (FLAGS (\Seen))
A7 OK Success
Set default port to COM1
Stable Library

=====
Native lib Version = RXTX-2.1-7
Java lib Version   = RXTX-2.1-7
Found port: COM1
Writing "r255" to //./COM1
Writing "r255" to //./COM1
Writing "r255" to //./COM1
Data entered: r255
Red is set to: 255
```

Gambar 4.25 Hasil *debug* pengambilan ratron dan komunikasi serial

Dari pengujian tersebut maka didapatkanlah hasil seperti Gambar 4.26 untuk perintah menghidupkan LED 1 dan LED 2. Untuk mematikan LED cukup mengganti “255” menjadi “0”. Inilah hasil akhir pengujian alat ini yang berhasil dengan baik.



Gambar 4.26 Hasil perintah R255 dan G255

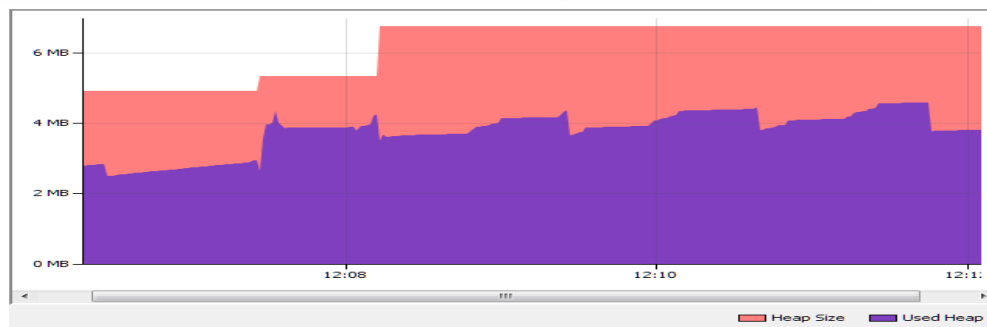
4.4 Analisis

Setelah pengujian dilakukan maka hal terakhir yang dilakukan adalah melakukan analisa kinerja dari alat ini. Analisa dilakukan dengan menggunakan fasilitas bantuan dari NetBeans untuk melakukan *benchmark* untuk alat ini. Terdapat empat kategori pengujian yang ditawarkan oleh NetBeans. Pertama adalah analisa memori yang digunakan secara umum. Kedua adalah analisa memori untuk masing objek yang digunakan. Ketiga adalah waktu yang dibutuhkan untuk masing-masing *method* untuk diproses. Lalu terakhir yang keempat adalah banyaknya *thread* yang digunakan selama proses aplikasi ini berjalan.

Data yang diberikan untuk sebuah aplikasi dengan basis pemrograman JAVA (Nick Mitchell, 2008):

1. Dibutuhkan 1G memori untuk mendukung beberapa ratus pengguna.
2. Menyimpan 500K *session state* untuk setiap pengguna.
3. Dibutuhkan 2M untuk sebuah *text index* untuk dokumen yang kecil.
4. Menciptakan 100K *temporary objects* untuk setiap *web hit*.

Untuk pembahasan pertama adalah penggunaan memori yang digunakan. Berdasarkan analisa yang diberikan oleh NetBeans didapatkan *heap size* yang dialokasikan untuk program ini adalah kurang lebih sebesar 7 MB. Sedangkan untuk *heap* yang paling sering digunakan berdasarkan grafik adalah 4 MB. Hasil ini didapatkan setelah dilakukan pengukuran selama 3 kali. Grafik dari analisa ini dapat dilihat pada Gambar 4.27.



Gambar 4.27 Grafik besar memori yang digunakan

Penggunaan *heap size* dengan besar memori 7 MB bisa dikatakan rendah. Hal ini disebabkan karena berdasarkan referensi (Nick Mitchell, 2008) aplikasi berbasis java rata-rata mempunyai *heap size* 256MB. Menurut (IBM, 2008) aplikasi dapat dikatakan efisien bila melakukan alokasi *heap* yang digunakan berada pada 40-70% dari *heap size*. Jika kita melihat pada Gambar 4.27 maka diketahui penggunaan heap berada dikisaran 50%.

Analisa kedua mengetahui besar memori yang digunakan untuk masing-masing objek yang menjadi sasaran utama dari program ini. Bila dilihat pada Tabel 4.2 maka terlihat bahwa objek `byte[]` menggunakan memori terbesar dalam program ini. Hal ini disebabkan karena penggunaannya untuk melakukan komunikasi data serial dan untuk melakukan *buffering*. Sedangkan menempati posisi kedua adalah `int[]` dan ketiga adalah `char[]`. Tabel 4.2 hanya menunjukkan cuplikan dari tabel yang sebenarnya karena untuk tabel sebenarnya objek yang ada mencapai 3001 objek.

Tabel 4.2 Alokasi memori untuk masing objek

Class Name – Allocated Objects	Bytes Allocated	Bytes Allocated	Objects Allocated
<code>byte[]</code>	33,64%	695392	12361
<code>int[]</code>	27,01%	558344	3839
<code>char[]</code>	18,34%	379224	29295
<code>java.lang.String</code>	2,54%	52464	20819
<code>sun.java2d.SunGraphics2D</code>	0,99%	20400	981
<code>java.util.HashMap\$Entry[]</code>	0,97%	20056	2158
...
<code>com.sun.mail.imap.protocol.IMAPAddress[]</code>	0,00%	0	0

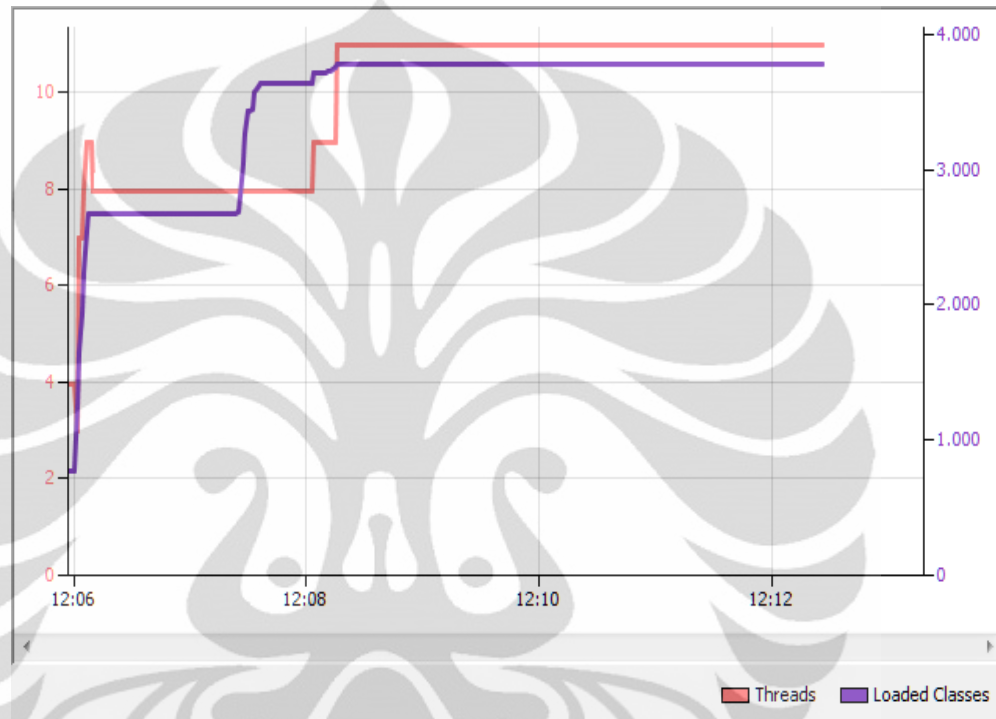
Tabel 4.3 Lama waktu untuk masing-masing proses *method*

Hot Spots - Method	Self time [%]	Self time	Invocations
javaapplication1.MailFetch.mainEmail(String[])	897.938	65234.991 ms	6
javaapplication1.MailTest.jalankanEmail(String[])	8.462.173	6147.75 ms	1
javaapplication1.EmailInterface.initComponents()	0.55222607	401.191 ms	1
javaapplication1.SimpleTest.serialEvent(gnu.io.SerialPortEvent)	0.28847027	209.573 ms	873
javaapplication1.SimpleTest.writetoport()	0.2873526	208.761 ms	222
javaapplication1.EmailInterface.<init>()	0.2007508	145.845 ms	1
javaapplication1.EmailInterface.main(String[])	0.115456656	83.879 ms	1
javaapplication1.SimpleTest.afterMat(String[])	0.112955615	82.062 ms	7
javaapplication1.SimpleTest.<init>()	0.068812326	49.992 ms	7
javaapplication1.EmailInterface jButton1ActionPerformed(java.awt.event.ActionEvent)	0.06623283	48.118 ms	1
javaapplication1.EmailInterface\$2.run()	0.03553074	25.813 ms	1
javaapplication1.SimpleTest.run()	0.011128734	8.085 ms	1
javaapplication1.MailTest\$1.messagesAdded(javax.mail.event.MessageCountEvent)	0.0048878337	3.551 ms	6
javaapplication1.SimpleTest.initwritetoport()	4,96E+02	0.036 ms	1
javaapplication1.EmailInterface\$1.actionPerformed(java.awt.event.ActionEvent)	3,99E+02	0.029 ms	1
javaapplication1.SimpleTest.<clinit>	3,44E+01	0.025 ms	1
javaapplication1.EmailInterface\$1.<init>(javaapplication1.EmailInterface)	2,89E+02	0.021 ms	1
javaapplication1.MailTest\$1.<init>(String[])	2,75E+02	0.02 ms	1
javaapplication1.EmailInterface.access\$000(javaapplication1.EmailInterface, java.awt.event.ActionEvent)	2,34E+02	0.017 ms	1
javaapplication1.EmailInterface\$2.<init>()	2,06E+02	0.015 ms	1

Analisa ketiga adalah mengenai total dan lama waktu yang digunakan untuk masing-masing *method*. Dari hasil yang didapat pada Gambar 4.28 diketahui bahwa *method* terlama yang bekerja adalah javaapplication1.MailFetch.mainEmail(String[]) yang mempunyai lama waktu 65234.991 mili detik. Hal ini terjadi dikarenakan penggunaannya dalam program ini memang paling banyak dilakukan untuk pengambilan ratron. Jadi dapat disimpulkan penggunaan terbesar *method* dalam program ini digunakan untuk pengendalian ratron.

Analisa keempat adalah banyaknya *thread* dan kelas yang digunakan selama proses ini berlangsung. Dari Gambar 4.28 terlihat bahwa grafik terus meningkat seiring dengan

jalannya waktu. Hal ini terjadi karena aktifitas yang dilakukan pengguna juga semakin tinggi sehingga jumlah *thread* serta kelas yang digunakan semakin meningkat. Namun kedua bagian itu menjadi normal dikarenakan sudah digunakannya semua *thread* dan kelas yang ada. Analisa ini kemudian mengakhiri pembahasan analisa yang disediakan oleh NetBeans.



Gambar 4.28 Banyak *thread* dan kelas yang digunakan

Analisa selanjutnya adalah mengenai penggunaan internet langsung (*direct internet*) yang diharuskan oleh alat ini, dalam hal ini menggunakan *modem*. Hal ini terjadi karena dalam aplikasinya penggunaan *proxy* kampus Universitas Indonesia (UI) sebagai media untuk melakukan hubungan antara komputer dan *server* ratron tidak dapat dilakukan. Hal itu terjadi karena setelah dilakukan pengetesan dengan debug, pada saat melakukan koneksi dengan ratron server, server dari *proxy* memberikan kesalahan berupa *malformed message*. Penjelasan dari pihak Java menyatakan bahwa untuk penggunaan paket JavaMail tidak disediakan fasilitas untuk melakukan *bypass proxy* tersebut (Oracle, Java Mail API - FAQ, 2011). Pihak Java hanya memberikan fasilitas menembus *proxy* dengan menggunakan SOCKET4, sedangkan

di dalam *proxy* UI fasilitas ini telah ditutup demi keamanan jaringan. Dari penjelasan tersebut dapat disimpulkan bahwa alat ini dapat berfungsi dengan baik bila jaringan yang digunakan tidak menggunakan pembatasan *proxy* dan hanya bisa untuk jaringan dengan internet langsung untuk saat ini.

Analisa selanjutnya adalah kecepatan akses dari ratron dikirim hingga lampu menyala. Ada dua jenis tes yang dilakukan pertama adalah menggunakan *handphone* (HP) dan satu lagi menggunakan komputer personal. HP yang digunakan adalah Samsung GT-S5620 untuk melakukan pengiriman ratron dengan provider Telkomsel dan menggunakan HSDPA. Sedangkan komputer menggunakan provider Telkom Speedy dengan akses data 384 Kbps. Berikut hasil yang didapat dari kedua tes dengan pengujian sebanyak 20 kali dalam satuan detik, lihat Tabel 4.4.

Tabel 4.4 Uji kecepatan data pada dua platform.

Percobaan Ke-	HP-Speedy	Speedy-Speedy
1	23,68	13,07
2	30,06	9
3	26,02	12,03
4	27,07	13,08
5	29	13,2
6	25,04	9,28
7	32,09	10,02
8	29	14,49
9	26,01	14,08
10	31,01	12,78
11	24	11,63
12	24,08	13,13
13	25,02	9,68
14	22,04	10,13
15	25,01	10,09
16	22,08	8,31
17	22,05	18,27
18	29	12,61
19	24,02	13,08
20	25	10,81
Rata-rata =	26,064	11,9385

5 KESIMPULAN

1. Aplikasi *Email Electronic Control*(EEC) atau Ratron Pengendali Elektronika(RPE) telah berjalan dengan sempurna sesuai dengan konsep desain yang sudah dijabarkan dengan menggunakan bahasa pemrograman JAVA dan C.
2. Salah satu tujuan utama dalam proyek kali ini yaitu pengujian penggunaan IMAP IDLE sebagai media bantu notifikasi ratron baru, berjalan dengan baik dengan hasil menyalanya lampu setelah dilakukan pengujian dengan spesifikasi *single user and single command*.
3. Analisa mengenai memori yang digunakan menyatakan bahwa aplikasi ini menggunakan memori *heap size* 7 MB dan dikatakan berjalan dengan efisien.
4. Penggunaan terbesar *method* dalam program ini digunakan untuk proses pengendalian ratron.
5. Kecepatan rata-rata akses dari HP ke Mikrokontroler adalah 26,064 detik dan dari komputer adalah 11,9385 detik.
6. JavaMail tidak dapat digunakan didalam sistem keamanan jaringan yang tinggi seperti didalam *proxy* kampus jadi sebagai alternatif penggunaan dapat digunakan internet langsung menggunakan *modem*. Untuk pengembangan selanjutnya adalah melakukan *bypass proxy*.
7. Untuk pengembangan lebih lanjut bisa ditambahkan *multiple user and multiple command* dan analisa lain seperti optimasi proses yang dilakukan.
8. Konsep pengendalian elektronika melalui ratron yang sudah terbukti berhasil, dapat dikembangkan untuk ditanamkan pada berbagai macam aplikasi seperti *Smart Home*, automasi pabrik dan kontrol jarak jauh.

6 DAFTAR REFERENSI

Ball, S. R. (2000). *Embedded Microprocessor Systems Real World Design* (2nd Edition ed.). Woburn, USA: Butterworth-Heinemann.

IBM. (2008). *Verifying that the Java heap size is correct*. Dipetik Januari 6, 2012, dari publib.boulder.ibm.com:publib.boulder.ibm.com/infocenter/javasdk/tools/topic/com.ibm

Leiba, B. (1997, Juni). Diambil kembali dari RFC 2177 - IMAP4 IDLE command: <http://tools.ietf.org/html/rfc2177>

Nick Mitchell, G. S. (2008, Oktober 19–23). *Building Memory-efficient Java Applications*. Nashville, Tennessee, USA.

Oracle. (2011, Januari). Diambil kembali dari Class IMAPStore: <http://javamail.kenai.com/nonav/javadocs/com/sun/mail/imap/IMAPStore.html>

Oracle. (2011, Oktober 11). *Java Mail API - FAQ*. Dipetik Januari 10, 2012, dari Java Mail API FAQ: <http://www.oracle.com/technetwork/java/javamail/faq-135477.html#proxy>

Telkomsel. (2010). *Telkomsel*. Dipetik Januari 10, 2012, dari Tarif KartuHalo: <http://www.telkomsel.com/product/kartuhalo/8723-Tarif-kartuHALO.html>

USB Implementers Forum, I. (2007, April). *Universal Serial Bus Micro-USB Cables and Connectors Specification*.

7 DAFTAR PUSTAKA

(2008, Februari). Diambil kembali dari Two way communcation with the serial port:
http://rxtx.qbang.org/wiki/index.php/Two_way_communcation_with_the_serial_port

(2008, Maret). Diambil kembali dari Arduino Forum - Serial Communication in Java: <http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1206094086>

(2009, April). Diambil kembali dari Parallel Communications:
http://rxtx.qbang.org/wiki/index.php/Parallel_Communications

(2009, Juli). Diambil kembali dari Discovering available comm ports:
http://rxtx.qbang.org/wiki/index.php/Discovering_available_comm_ports

(2010, April). Diambil kembali dari Event Based Two Way Communication:
http://rxtx.qbang.org/wiki/index.php/Event_Based_Two_Way_Communication

(2011, Juni). Diambil kembali dari Discovering comm ports:
http://rxtx.qbang.org/wiki/index.php/Discovering_comm_ports

(2011, Juni). Diambil kembali dari Deploying JAVA with RXTX:
http://rxtx.qbang.org/wiki/index.php/Deploying_JAVA_with_RXTX

(2011, Desember). Diambil kembali dari Arduino Playground - Java:
<http://www.arduino.cc/playground/Interfacing/Java>

Ayala, K. J. (1991). *The 8051 Microcontroller Architecture Programming and Applications* (1st Edition ed.). (T. T. Inc., Penyunt.) St. Paul, USA: West Publishing Company.

Ball, S. R. (2000). *Embedded Microprocessor Systems Real World Design* (2nd Edition ed.). Woburn, USA: Butterworth-Heinemann.

Deitel, H. M. (2004). *Java™ How to Program* (6th Edition ed.). New Jersey, USA: Prentice Hall.

Deyn, M. (2010, Juni 15). *Deyn Blog*. Dipetik September 25, 2011, dari <http://mikedeyn.blogspot.com>: <http://mikedeyn.blogspot.com/2010/06/ic-mikrokontroler-at89s51.html>

Fitzgerald, S., Cerrito, C., & Igoe, T. (2011, November). Diambil kembali dari Arduino Switch Case 2: <http://www.arduino.cc/en/Tutorial/SwitchCase2>

Fitzgerald, S., Mellis, D. A., Igoe, T., & Cerrito, C. (2011, Desember). Diambil kembali dari Arduino Pysical Pixel: <http://www.arduino.cc/en/Tutorial/PhysicalPixel>

Harold, E. R. (2006). *Java™ I/O* (2nd Edition ed.). (D. Cameron, Penyunt.) Sebastopol, USA: O'Reilly Media, Inc.

IBM. (2008). *Verifying that the Java heap size is correct*. Dipetik Januari 6, 2012, dari publib.boulder.ibm.com:publib.boulder.ibm.com/infocenter/javasdk/tools/topic/com.ibm

Leiba, B. (1997, Juni). Diambil kembali dari RFC 2177 - IMAP4 IDLE command: <http://tools.ietf.org/html/rfc2177>

Naik, P. (2009, April). Diambil kembali dari Communicating with ports using javax.comm package for Windows: <http://pradnyanaik.wordpress.com/2009/04/07/communicating-with-ports-using-javaxcomm-package-for-windows/>

Nick Mitchell, G. S. (2008, Oktober 19–23). *Building Memory-efficient Java Applications*. Nashville, Tennessee, USA.

Oracle. (2011, Januari). Diambil kembali dari Class IMAPStore: <http://javamail.kenai.com/nonav/javadocs/com/sun/mail/imap/IMAPStore.html>

Oracle. (2011, Oktober 11). *Java Mail API - FAQ*. Dipetik Januari 10, 2012, dari Java Mail API FAQ: <http://www.oracle.com/technetwork/java/javamail/faq-135477.html#proxy>

Ramli, K. (2009, September). *Embedded System and Wireless Sensor Networks*.

Sickle, T. V. (2001). *Programming Microcontrollers in C* (2nd Edition ed.). (C. S. Lewis, & L. T. Publishing, Penyunt.) USA: LLH Technology Publishing.

Singla, V. (2010, Januari). Diambil kembali dari E-Mailing Through Java: <http://www.vipan.com/htdocs/javamail.html>

Telkomsel. (2010). *Telkomsel*. Dipetik Januari 10, 2012, dari Tarif KartuHalo: <http://www.telkomsel.com/product/kartuhalo/8723-Tarif-kartuHALO.html>

USB Implementers Forum, I. (2007, April). *Universal Serial Bus Micro-USB Cables and Connectors Specification*.

ZHENG, H., & CAI, L. (2009). The Utilization of Push Mail in China. *Scientific Research* , 2 (3), 186-189.