JURNAL MEDIA APLIKOM
ISSN : 2086 – 972X
Vol 1 No.1 Maret 2010

PROGRAMMABLE
RING TONE TEXT TRANSFER LANGUAGE PLAYER

Benedictus Herry Suharto

Dosen Sekolah Tinggi Ilmu Komputer Yos Sudarso Purwokerto

Alamat e-mail: bherrys@gmail.com

ABSTRACT

The aim of this research is to design and implement a programmable Ring Tone Text Transfer Language (RTTTL) player. The system consists of microcontroller hardware that produce custom musics from RTTTL data format that programmable using personal computer (PC). USB (unversal Serial Bus) was used to establish communication between PC and the device, in order to program the music data in the device, so the vary music can be customized by user.

The result of the research showed that the music in the device can be played and reprogramming using PC.

Keywords : *RTTTL, programmable, music, USB, PC.*

1. PENDAHULUAN

Kemajuan teknologi komputer di bidang seni musik telah terbukti meningkatkan kualitas suara perangkat musik elektrik dan mendorong terciptanya berbagai format musik dijital baru, seperti *Musical Instrument Digital Interface* (MIDI), *Music Macro Language* (MML), *id Software Music Format* (IMF), *Extensible Music Format* (XMF), *Rich Music Format* (RMF), *Synthetic Music Mobile Application* (SMAF), dan lain sebagainya. Format ini memiliki spesifikasi yang kompleks dan umumnya digunakan untuk memproduksi musik kualitas tinggi.

Selain format tersebut, ada beberapa format musik dijital sederhana yang diimplementasikan pada perangkat elektronik yang juga sederhana dan ekonomis, seperti mainan anak-anak, bel sekolah atau rumah, alarm, klakson sepeda, pohon natal, dan juga telepon genggam (HP). Salah satu contoh formatnya adalah *Ring Tone Text Transfer Language* atau RTTTL. RTTTL (kadangkala disebut sebagai *Nokring format ringtone*) pertama kali dibuat oleh perusahaan HP Nokia pada tahun 1999 dan khusus diimplementasikan untuk HP yang diproduksinya. Tujuan utama dari pengembangan format RTTTL ini adalah untuk mentransfer dan memutar musik ringtone dalam HP Nokia agar HP tersebut memiliki banyak pilihan ringtone bagi pemakainya.

Seiring dengan perkembangan teknologi komputer pada perangkat bergerak (*mobile device*), khususnya *smartphone*, format RTTTL mulai ditinggalkan oleh Nokia dan diganti dengan format *Motion Picture Expert Group-1 Audio Layer 3* (mp3) yang memiliki kualitas suara lebih baik. Tetapi, karena spesifikasinya yang sederhana, RTTTL kemudian banyak diadobsi dan diimplementasikan pada berbagai produk barang komersial untuk membangkitkan efek musik sederhana, seperti efek musik pada pohon Natal, bel sekolah, robot mainan dan lain sebagainya. Efek musik pada produk tersebut umumnya dibangkitkan menggunakan sebuah *chip* tunggal yang didesain sedemikian rupa agar dapat memainkan (*play*) data musik dengan format RTTTL menjadi bunyi musik *tone*.

Kekurangan utama dari *chip* RTTTL ini adalah data musik RTTTL dalam *chip* bersifat permanen dan telah diinjeksikan ke dalam *chip* pada saat *chip* tersebut diproduksi. Sehingga musik yang dimainkan akan diulang terus menerus dan tidak dapat diganti atau diubah. Kekurangan ini menyebabkan produk-produk yang mengimplementasikannya hanya membangkitkan beberapa musik dan mengulangnya terus menerus baik secara sekuensial maupun acak. Hal ini mengakibatkan produk akan menjadi membosankan, terutama jika produk tersebut dipakai sebagai pelengkap suasana atau hiburan, seperti bel sekolah atau pohon Natal.

Tujuan yang ingin dicapai dalam penelitian ini adalah membuat dan mengimplementasikan suatu sistem pemutar (*player*) musik RTTTL yang dapat diprogram ulang menggunakan perangkat PC (*personal computer*) atau *notebook*. Dalam penelitian ini digunakan perangkat keras *single chip computer* atau chip mikrokontroller AT89S52 yang berfungsi sebagai pemutar musik RTTTL dan perangkat keras PC yang digunakan sebagai antarmuka pengelolaan data musik RTTTL yang akan dimainkan oleh pemutar musik RTTTL. Perangkat lunak untuk memprogram *chip* menggunakan bahasa pemrograman C, sedangkan perangkat lunak untuk memprogram antarmuka pengelolaan data musik RTTTL pada komputer PC menggunakan bahasa pemrograman Java.

Hasil penelitian diharapkan dapat digunakan oleh masyarakat luas sebagai alternatif implementasi untuk produk-produk komersil yang membangkitkan musik RTTTL, seperti bel sekolah atau pohon Natal. Sehingga bunyi musik yang dikeluaran oleh produk tersebut dapat dikostumisasi sesuai dengan keinginan pemiliknya. Selain itu, penelitian ini dapat juga digunakan oleh peneliti lain yang membutuhkan perangkat pembangkit efek musik pada penelitiannya, seperti pada bidang robotika, instrumentasi, peralatan kontrol, peralatan survey, peralatan timer, dan peralatan keamanan.

2. DASAR TEORI

2.1. Teori musik

Notasi musik adalah suatu sistem untuk menulis musik. Menulis musik pada selembar kertas memiliki arti yang berbeda dengan merekam musik, walaupun memiliki tujuan yang sama yaitu untuk menyimpan data musik. Sebuah *note* musik digunakan untuk mereferensikan satu titian nada yang dapat digambarkan dengan sistem notasi. Salah satu notasi yang umum adalah menggunakan huruf A, B, C, D, E, F, dan G. Dalam satu baris nada terdapat 12 *tone*, sehingga dibutuhkan modifikasi dalam penulisan notasinya. Dua modifikasi utama adalah penambahan tanda *sharp* (#) dan *flat* (I) yang memiliki nada lebih tinggi atau lebih rendah. Modifikasi ini memberikan tambahan 5 *note* untuk memenuhi skala kromatik.

Durasi *note* tidak disertakan pada penulisan musik, sebagai gantinya digunakan 1, 1/2, 1/4, 1/8, dan 1/16 dari *note* penuh. Durasi dari *note* bergantung pada bagian musik yang dimainkan. Satu bagian dimainkan pada kecepatan (tempo) yang berakhir selama 4 detik, sedangkan bagian lain mungkin hanya 2 detik. Waktu jeda antar *note* juda dimainkan sesuai dengan durasinya.

Oktaf musik merupakan interval antara satu *note* dengan *note* lain yang memiliki frekuensi dua kalinya. Misalnya satu nada memiliki frekuensi 400 Hz, maka nada lain yang berada satu oktaf diatasnya akan memiliki frekuensi 800 Hz. Frekuensi nada-nada pada beberapa oktaf diperlihatan pada tabel 1.

Tabel 1. Frekuensi nada-nada pada oktaf yang berbeda

	Octave								
Note	1	2	3	4	5	6	7	8	9
A	27.5	55.0	110.0	220.0	440.0	880.0	1760.0	3520.0	7040.0
A#/Bb	29.1	58.3	116.5	233.1	466.2	932.4	1864.7	3729.4	7458.9
B	30.9	61.7	123.5	247.0	493.9	987.8	1975.7	3951.3	7902.7
C	32.7	65.4	130.8	261.6	523.3	1046.6	2093.2	4186.5	8372.9
C#/Db	34.6	69.3	138.6	277.2	554.4	1108.8	2217.7	4435.5	8871.1
D	36.7	73.4	146.8	293.7	587.4	1174.8	2349.7	4699.5	9398.9
D#/Eb	38.9	77.8	155.6	311.2	622.4	1244.8	2489.5	4979.1	9958.1
E	41.2	82.4	164.9	329.7	659.4	1318.8	2637.7	5275.3	10550.6
F	43.7	87.3	174.7	349.3	698.7	1397.3	2794.6	5589.2	11178.4
F#/Gb	46.2	92.5	185.1	370.1	740.2	1480.4	2960.8	5921.8	11843.5
G	49.0	98.0	196.1	392.1	784.3	1568.2	3137.1	6274.1	12548.2
G#/Ab	51.9	103.9	207.7	415.5	830.9	1661.9	3323.7	6647.4	13294.8

2.2. Ring Tone Transfer Language

Ring Tone Text Transfer Language atau RTTTL merupakan format file *text-based* sederhana yang digunakan untuk membuat data ringtone. Format ini pertama kali digunakan pada perangkat *handphone* Nokia. Format RTTTL berupa data string yang terbagi dalam tiga bagian, yaitu nama, nilai *default*, dan data.

- Bagian nama berupa data string yang menjelaskan nama dari ringtone. Bagian nama ini tidak boleh lebih dari 10 karakter dan tidak boleh memiliki karakter titik dua ":".
- Bagian nama *default* berupa sekumpulan nilai yang dipisahkan dengan tanda koma, dimana setiap nilai berisi kata kunci dan nilainya. Kata kunci yang dapat diberikan pada bagian ini adalah d (durasi), o (oktaf), dan b (*beat* per menit). Jika bagian nama *default* tidak dispesifikasikan, maka akan otomatis diberi nilai d=4, o=6, dan b=63. Nilai yang dapat diberikan untuk durasi adalah 1, 2, 4, 6, 8, 16, dan 32. Sedangkan untuk oktaf nilai yang valid adalah 5, 6, 7, dan 8. Untuk nilai *beat* yang diijinkan adalah 25, 28, 31, 35, 40, 45, 50, 56, 63, 70, 80, 90, 100, 112, 125, 140, 160, 180, 200, 225, 250, 285, 320, 355, 400, 450, 500, 565, 635, 715, 800 dan 900.
- Bagian data merupakan bagian terakhir yang berisi data musik yang dipisahkan dengan tanda koma. Format kodenya adalah sebagai berikut:

[duration] note [scale] [special-duration],[duration] note [scale] [special-duration], dst...

Contoh data musik RTTTL adalah sebagai berikut:

SilentNight:d=4,o=5,b=160:g.,8a,g,2e.,g.,8a,g,2e.
,2d6,d6,2b.,2c6,c6,2g.

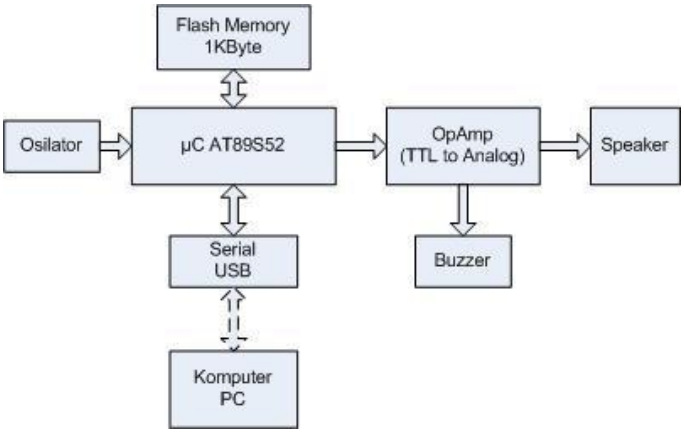
3. METODE PENELITIAN

Metodologi yang dilakukan dalam penelitian ini adalah sebagai berikut:

1) Merancang dan menguji perangkat keras RTTTL Player.
2) Merancang perangkat lunak RTTTL Player.
3) Merancang perangkat lunak pengelolaan data musik RTTTL.
4) Menganalisa hasil.
5) Membuat kesimpulan.

3.1. Perancangan dan Pengujian Perangkat Keras RTTTL Player

Diagram kotak dari sistem perangkat keras RTTTL player adalah sebagai berikut:



Gambar 1. Diagram kotak sistem RTTTL player

Osilator kristal digunakan untuk membangkitkan *clock* dengan frekuensi 11.059 MHz. *Clock* ini merupakan frekuensi running dari mikrokontroller AT89S52 yang digunakan sebagai referensi dari pembangkit nada *tone*.

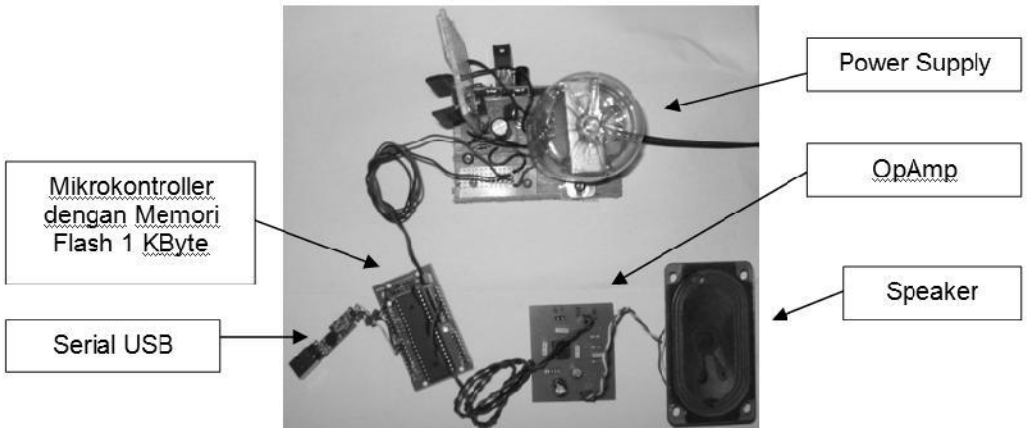
Untuk menyimpan data musik RTTTL yang dikirim dari PC digunakan memori Flash (*nonvolatile*) sebesar 1024 byte. Kapasitas memori ini cukup untuk menyimpan 10 data musik RTTTL dengan panjang data 100 byte.

Mikrokontroller AT89S52 hanya memiliki port komunikasi serial dengan level logika TTL (*Transistor–Transistor Logic*). Agar dapat berkomunikasi langsung dengan PC menggunakan port USB (*Universal Serial Bus*) dan protokol RS232, perlu ditambahkan perangkat keras konversi TTL-ke-USB. Pada penelitian ini perangkat keras konversi tersebut menggunakan kabel data Nokia CA-42 yang telah dimodifikasi seperti pada gambar berikut ini:

Gambar 2. Modifikasi kabel data Nokia CA-42.

Sinyal *tone* musik keluaran dari perangkat RTTTL player ini masih berupa gelombang kotak yang dapat langsung dihubungkan ke *piezoelectric buzzer*. *Piezoelectric buzzer* merupakan *self-oscillating buzzer* daya rendah dangan pembangkit sinyal dan *buzzer* seperti pada *speaker buzzer* komputer PC. Agar dapat digunakan bersama dengan perangkat pengeras suara untuk kebutuhan tertentu, seperti alarm atau bel sekolah, maka perlu ditambahkan perangkat penyesuai (*OpAmp*). Perangkat ini digunakan untuk mengkonversi sinyal TTL ke analog. Selain itu juga digunakan untuk menyamakan impedansi keluaran sinyal RTTTL player dengan impedansi masukan dari perangkat pengeras suara (atau *speaker* perangkat audio) yang biasanya sebesar 8 Ohm.

Berikut ini adalah gambar perangkat keras RTTTL player yang telah dibuat :

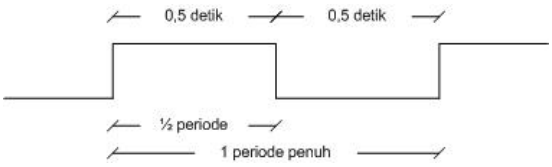


Gambar 3. Perangkat keras RTTTL Player.

Perangkat keras komputer PC yang digunakan dalam penelitian ini adalah perangkat *notebook* yang memiliki spesifikasi sebagai berikut:

- Prosessor Intel Centrino 1,4 GHz.
- Memory 1024 MByte
- Harddisk 40 GByte
- I/O : 2 port USB 2.1, 1 port Audio Out, 1 port Mic.
- Sistem Operasi Windows XP SP3

Pengujian perangkat keras RTTTL player dilakukan dengan menggunakan program sederhana untuk membangkitkan gelombang kotak 1 Hz pada *port* P1.3 dari *chip* mikrokontroller AT89S52. *Port* ini merupakan *port* keluaran sinyal *tone* musik dari RTTTL player. Pada program pengujian ini digunakan internal timer 0 dari mikrokontroller AT89S52 yang bertanggung jawab memberikan waktu penundaan periode gelombang pada saat pembentukan sinyal. Gelombang kotak dibentuk dengan men-set *port* P1.3 ON (1) dan OFF (0) pada selang waktu penundaan sebesar 0,5 detik, sehingga akan didapatkan gelombang kotak seperti pada gambar 4.



Gambar 4. Gelombang kotak 1 Hz.

Untuk memastikan bahwa perangkat RTTTL player telah berfungsi sesuai dengan rancangan, port P1.3 diuji dengan menggunakan peralatan *logic probe*. Jika lampu indikator berkedip dengan jeda waktu 0,5 detik, maka perangkat RTTTL dapat dipastikan telah sesuai dengan rancangan.

Pengujian komunikasi USB serial antara perangkat RTTTL player dengan komputer PC dilakukan dengan menambahkan rutin program pengujian komunikasi serial RS232 (UART) pada perangkat RTTTL player. Rutin program ini berisi konfigurasi parameter komunikasi (*baudrate*, *data bit*, *parity*, dan *stop bit*) dan data-data yang akan dikomunikasikan pada saat pengujian. Prosedur rutin program yang sama juga diterapkan pada program pengujian di sisi komputer PC. Pengujian ini dikatakan telah berhasil jika komputer PC mengirimkan data integer “123”, maka akan dijawab oleh RTTTL player dengan mengirimkan data integer “456”. Sebaliknya, jika komputer PC mengirimkan data integer “456” maka akan dijawab dengan mengirimkan data integer “123”.

3.2. Perancangan dan Pengujian Perangkat Lunak RTTTL Player

Perangkat lunak RTTTL Player berisi 4 bagian algoritma pemrograman, yaitu:

- Algoritma bagian inisialisasi perangkat RTTTL player yang berfungsi untuk mengkonfigurasi protokol akses flash memori dan variabel-variabel pendukung program.
- Algoritma bagian pemutar (*play*) RTTTL yang berfungsi mengkonversi data musik RTTTL menjadi *tone* yang dikeluarkan ke port P1.3.
- Algoritma bagian komunikasi serial USB yang berfungsi untuk menerima dan mengirim data musik RTTTL dari dan ke komputer PC.
- Algoritma bagian pengaksessan memori yang berfungsi mengelola pembacaan dan penulisan data RTTTL dari dan ke flash memori.

Pada kondisi normal (*running*), timer 1 dan timer 0 diinisialisasi untuk membangkitkan bunyi *note* (gelombang kotak) dan mengelola durasi *note*. Prinsip pembangkitan bunyi *note* adalah dengan memberi nilai SFR TH1 dan TL1 dengan nilai tertentu yang mengakibatkan pewaktuan menjadi *overflow*. Nilai tertentu ini berkisar antara 1 sampai 12 yang digunakan untuk mewakili 12 nada yaitu a,a#,b,c,c#,d,d#,e,f,f#,g, dan g#. Ketika terjadi *overflow*, port P1.3 diubah kondisinya menjadi ON atau OFF yang merupakan setengah periode dari gelombang kotak. Berikut potongan program untuk membangkitkan bunyi *note*:

```
while(bit_ada_waktu)
 {
   if(n != 0)
   {
      TH1 = HBYTE(n); TL1 = LBYTE(n);  TR1 = 1;
       while(!TF1);
      TF1 = 0;  TR1 = 0;
      AUDIO_OUTPUT = !AUDIO_OUTPUT;
   }
}
```

Sedangkan prinsip pengelolaan durasi adalah dengan memberikan TH0 dan TL0 suatu nilai tertentu yang dihasilkan dari penghitungan tempo pada setiap nada yang diberikan. Nilai yang valid untuk pengelolaan periode adalah 1, 2, 4, 8, 16 dan 32 sebagai bilangan pembagi dari nilai durasi penuh. Misalkan jika durasi penuh (dp) adalah 4 detik (o = 4), maka nilai diatas akan mewakili:

- durasi = dp / 1 = 4/1 = 4.000 dt (detik)
- durasi = dp / 2 = 4/2 = 2.000 dt
- durasi = dp / 4 = 4/4 = 1.000 dt
- durasi = dp / 8 = 4/8 = 0.500 dt
- durasi = dp / 16 = 4/16 = 0.250 dt
- durasi = dp / 32 = 4/32 = 0.125 dt

Jika tanda “titik” ditemukan dalam deretan data musik RTTTL, maka durasi akan dikalikan dengan 1,5. Ketika pewaktuan timer 0 *overflow*, timer 1 akan di-reset dan informasi nilai nada selanjutkan akan di-set-kan ke timer 1.

Pada kondisi program, komunikasi serial perangkat RTTTL player diinisialisasi menggunakan mode 1 yang merupakan mode 8-bit UART. Pada mode komunikasi ini dibutuhkan pembangkit *baud rate* yang dihasilkan oleh timer 1 dalam mode 2. Untuk memperoleh *baud rate* sebesar 9600 bps (*bit per second*), SFR TH1 diberi nilai dengan menggunakan formula berikut (Schultz, 2004):

$$\text{Baud rate} = \frac{(K * \text{Frekuensi osilator})}{(32 * 12 * (256 - TH1))}$$

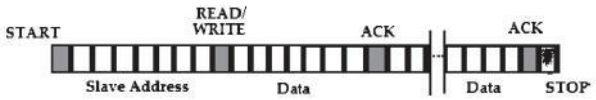
$$9600 = \frac{(1 * 11.059.000)}{(32 * 12 * (256 - TH1))} \quad \text{===> K = 1 untuk SMOD = 0}$$

TH1 = 253 atau 0xFD (dalam heksadesimal)

Ketika data musik RTTTL diterima melalui *port* serial, program akan memanggil rutin algoritma penyimpanan data ke dalam memori flash melalui *port* P3.6 (SCL) dan P3.7 (SDA) menggunakan protokol I^2C (*Inter-*

Integrated Circuit). Gambar 5 memperlihatkan format protokol I^2C, sedangkan langkah penulisan dan pembacaan data pada memori flash menggunakan protokol I^2C (Parab, 2008) (Durham, 2004) adalah sebagai berikut:

1) Memanggil rutin Start() untuk mengirim bit START.
2) Memanggil rutin WriteI2C(alamat) untuk menunjuk pada alamat yang dituju.
3) Memanggil rutin WriteI2C(baca/tulis) untuk mengirim bit baca atau bit tulis.
4) Memanggil rutin ReadI2C atau WriteI2C untuk mendapatkan atau mengirim bit acknowledge.
5) Memanggil rutin ReadI2C atau WriteI2C untuk membaca atau menulis byte data.
6) Memanggil rutin ReadI2C atau WriteI2C untuk mendapatkan atau mengirim bit acknowledge.
7) Memanggil rutin Stop() untuk mengirim bit STOP.

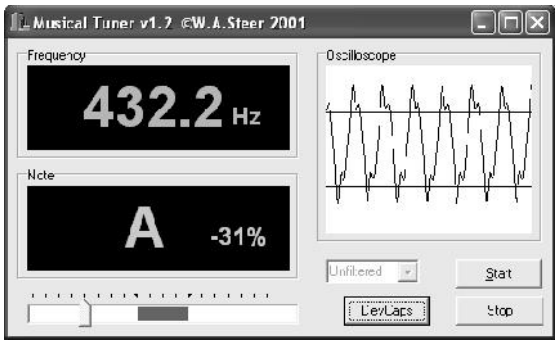


Gambar 5. Format Protokol I^2C.

Perangkat lunak RTTTL player diuji dengan merekayasa nilai dari SFR TH0, TL0, TH1, dan TL1 agar dapat membangkitkan sinyal *tone* tangga nada (A, A#, B, C, C#, D, D#, E, F, F#, G, G#) pada *port* keluaran P1.3 masing-masing selama 1 detik. Data musik RTTTL yang digunakan untuk pengujian ini adalah sebagai berikut:

{UjiTone:d=4,o=5,b=150:1p,1a,1a#,1b,1c
,1c#,1d,1d#,1e,1f,1f#,1g,1g#,1p}

Tone keluaran dari *port* P1.3 kemudian disampel menggunakan mikrofon PC komputer dan dianalisis menggunakan aplikasi "*Musical Tuner*" rancangan William Andrew Steer, PhD (University College London). Pengujian dikatakan berhasil jika seluruh tangga nada yang dikeluarkan dari port P1.3 dapat dideteksi oleh aplikasi "*Musical Tuner*". Prosentase kesalahan dari ketepatan frekuensi tiap nada note dapat diabaikan karena perangkat yang dirancang tidak ditujukan untuk menghasilkan suara musik kualitas tinggi.



Gambar 6. Aplikasi "Musical Tuner".

3.3. Perancangan dan Pengujian Perangkat Lunak Pengelolaan Data Musik RTTTL

3.3.1. Kebutuhan Sistem

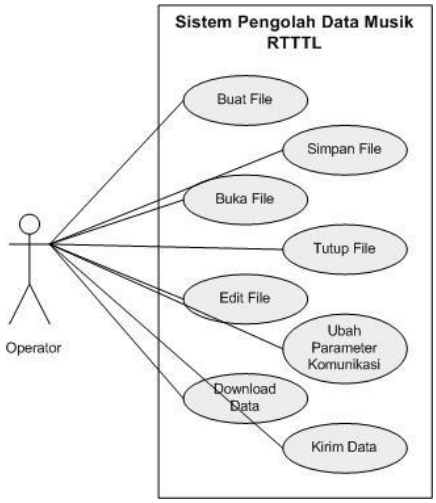
Ada tiga kebutuhan yang harus disediakan oleh sistem aplikasi Pengelolaan Data Musik RTTTL, yaitu:

1) Kemampuan mengolah data musik RTTTL. Pada penelitian ini, perancangan pengolahan data musik RTTTL dibatasi hanya pada penyediaan editor untuk menulis data musik RTTTL. Kemampuan untuk melakukan konversi dari format data musik lain tidak dirancang.
2) Kemampuan menyimpan dan menggunakan kembali data musik yang telah diolah.
3) Kemampuan melakukan komunikasi ke perangkat RTTTL player, sekaligus dapat mengirim dan mengambil data

musik RTTTL ke dan dari perangkat RTTTL player.

3.3.2. Perancangan Sistem

Sistem aplikasi Pengelolaan Data Musik RTTTL merupakan sebuah aplikasi *stand alone*, oleh karenanya pada sistem aplikasi ini hanya memiliki satu aktor, yaitu operator. Operasi yang dapat dilakukan oleh aktor operator digambarkan pada diagram *use case* berikut ini:



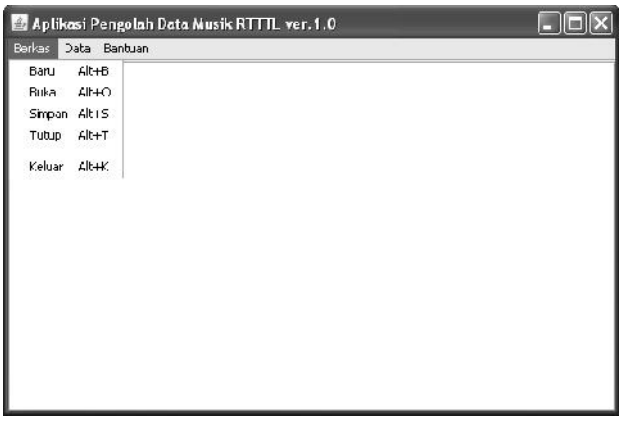
Gambar 7. *Use case* Sistem Pengolahan Data Musik RTTTL.

3.3.3. Perancangan Antarmuka Pengguna Sistem

Desain antarmuka aplikasi sistem pengelolaan data musik RTTTL dirancang berbasis pada GUI Swing Java. Antarmuka utama berupa area teks yang digunakan untuk mengolah data musik dengan format RTTTL. Pengguna dapat melakukan interaksi dengan sistem melalui menu-menu navigasi dan tombol-tombol fungsi. Menu utama terdiri atas tiga menu, yaitu:

1) Berkas, berisi sub menu Baru, Buka, Simpan, Tutup, dan Keluar.
2) Data, berisi sub menu Konfig COM, Uji COM, Kirim Data, dan Download data.
3) Bantuan, berisi sub menu Tentang Program, Lisensi dan Manual.

Berikut ini disain antarmuka yang dirancang:



Gambar 8. Disain antarmuka aplikasi.

3.3.4. Perancangan Program

Bahasa pemrograman Java menyediakan class GUI swing yang lengkap, sehingga mempermudah pembuatan program Aplikasi Pengolahan Data Musik RTTTL. Beberapa komponen Swing “siap-pakai” yang dipakai

dalam pemrograman aplikasi ini adalah *JFrame, JTextAres, JMenu, JMenuItem, JButton, JTextField, JDialog, JFileChooser,* dan *JPanel*. Program aplikasi bantu pemrograman yang digunakan adalah IDE Netbeans 5.5. Dengan menggunakan aplikasi bantu ini, programmer hanya perlu melakukan pengubahan dan penyesuaian properti komponen-komponen swing untuk membangun aplikasi sesuai dengan rancangan antarmuka pengguna yang telah dibuat (Hartati, 2007).

Komponen API Java yang memiliki peranan penting dalam pemrograman ini adalah class *InputStream, OutputStream, FileInputStream,* dan *FileOutputStream*. Class-class ini digunakan untuk mengalirkan data keluar dan masuk program. Sedangkan pustaka eksternal yang dipakai adalah pustaka *rxtx* yang dirancang oleh Keane Jarvi (*www.rxtx.org*). Pustaka ini digunakan untuk melakukan komunikasi dengan perangkat RTTTL player menggunakan protokol RS232. Berikut ini method *setSerialPortParameters()* dan *connect()* yang digunakan dalam program untuk melakukan komunikasi dengan perangkat RTTTL player:

```
private void setSerialPortParameters() throws IOException {
    int baudRate = 9600;
    try {
serialPort.setSerialPortParams(baudRate,
SerialPort.DATABITS_8,
SerialPort.STOPBITS_1,
SerialPort.PARITY_NONE);
serialPort.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);
    } catch (UnsupportedCommOperationException ex) { }
  }
  public boolean connect(String portName) throws IOException {
    boolean sukses = true;
    try {
      portId = CommPortIdentifier.getPortIdentifier(portName);
      serialPort = (SerialPort) portId.open("KTP test", 5000); setSerialPortParameters();
      outputStream = serialPort.getOutputStream();
      inputStream = serialPort.getInputStream();
    } catch (NoSuchPortException ex) { sukses = false;
    } catch (PortInUseException e) { sukses = false;
    } catch (IOException e) { serialPort.close(); sukses = false; }
    return sukses;
  }
```

4. HASIL PENELITIAN DAN PEMBAHASAN

Data hasil pengujian tangga nada tone dengan menggunakan aplikasi "*Musical Tuner*" ditampilkan dalam tabel 2. Selisih antara frekuensi referensi dan frekuensi aktual berbeda-beda untuk tiap tangga nada. Selisih perbedaan ini tidak membentuk sebuah pola. Walaupun terlihat pada tabel bahwa mulai dari nada A sampai E terjadi peningkatan selisih positif, tetapi untuk nada F sampai G# terjadi peningkatan selisih negatif. Hal ini menunjukkan bahwa kesalahan tidak terjadi pada penempatan dan perhitungan nilai pada SFR timer 0 dan 1 perangkat RTTTL player. Tetapi lebih disebabkan karena faktor kualitas perangkat keras yang dipakai dalam pengujian, seperti osilator kristal, catu daya, mikrokontroller, dan sistem audio pada PC (mikrofon dan *soundcard*). Selain itu derau (*noise*) dari luar sistem dapat pula masuk ke dalam mikrofon PC dan mengganggu proses konversi analog ke digital pada sistem audio PC.

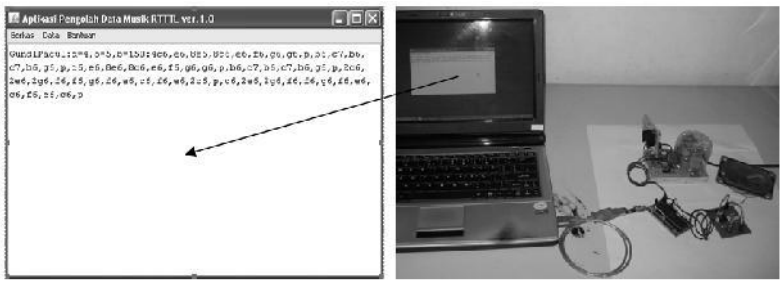
Tabel 2. Data pengujian tangga nada tone.

Durasi = 4/4 Oktaf = 5 Tempo = 150				
RTTTL	**Nada**	**Frekuensi Referensi (Hz)**	**Frekuensi Aktual (Hz)**	**Selisih Frekuensi (Hz)**
1a	A	440Hz	432,2 Hz	7,8
1a#	A#	466 Hz	457,7 Hz	8,3
1b	B	494 Hz	484,4 Hz	9,6
1c	C	523 Hz	511,7 Hz	11,3
1c#	C#	554 Hz	542,1 Hz	11,9
1d	D	587 Hz	573,1	13,9
1d#	D#	622 Hz	606,2	15,8
1e	E	659 Hz	641,9	17,1
1f	F	698 Hz	717,0	-19
1f#	F#	740 Hz	757,8	-17,8
1g	G	784 Hz	802,7	-18,7
1g#	G#	830 Hz	840,1	-10,11

Pada penelitian ini, pergeseran ketepatan frekuensi tiap nada *note* yang relatif kecil tersebut dapat diabaikan. Karena perangkat yang dirancang tidak ditujukan untuk menghasilkan suara musik kualitas tinggi. Faktor lain yang menjadi penguat pernyataan ini adalah nada *tone* keluaran dari perangkat RTTTL player yang dideteksi oleh aplikasi "*Musical Tuner*" tidak menunjukkan adanya perbedaan dari mulai nada A sampai G#. Misalnya Data RTTTL yang diberikan adalah 1a, dibunyikan oleh perangkat RTTTL player sebagai nada A dan dideteksi oleh aplikasi "*Musical Tuner*" sebagai nada A pula.

Pengujian keseluruhan sistem (RTTTL player dengan PC) dilakukan dengan membuat atau menulis data musik RTTTL menggunakan aplikasi sistem pengelolaan data musik RTTTL. Data ini kemudian dikirimkan ke perangkat RTTTL player menggunakan koneksi USB. Hasil pengamatan memperlihatkan bahwa setelah seluruh proses pengiriman selesai dan perangkat RTTTL player berada pada kondisi *running*, perangkat tersebut kemudian mengeluarkan bunyi musik *tone* sesuai dengan data-data yang dikirimkan. Pada proses pengambilan (*download*) data dari perangkat RTTTL player ke PC, juga memperlihatkan konsistensi data yang sama seperti pada data saat dikirim ke perangkat.

Konfigurasi perangkat keras dan cuplikan aplikasi pada saat pengujian ini dapat dilihat pada gambar 9.



Gambar 9. Konfigurasi RTTTL player dan aplikasi pengelolaan data musik RTTTL.

5. KESIMPULAN DAN SARAN

Berdasarkan hasil pengujian dan implementasi pada penelitian ini, maka dapat ditarik kesimpulan sebagai berikut:

1. Perangkat RTTTL player yang dirancang dapat diprogram ulang (*programmable*) dengan menggunakan PC atau notebook untuk menkostumisasi bunyi musik *tone* sesuai dengan keinginan pengguna.
2. Aplikasi sistem pengelolaan data musik RTTTL yang dirancang memberikan kemudahan dalam penyuntingan data musik RTTTL dan proses pemrograman ulang data musik pada perangkat RTTTL player.
3. Nilai tambah dari sistem aplikasi yang dikembangkan adalah kemampuan untuk dijalankan pada berbagai sistem operasi (*multiplatform*), seperti Linux dan MacOS. Hal ini dimungkinkan karena aplikasi dibuat menggunakan bahasa pemrograman Java yang *multiplatform*.

Berikut ini beberapa saran yang dapat dilakukan untuk menyempurnakan sistem yang telah dibuat:

1. Perangkat keras mikrokontroller dari RTTTL player dapat diganti dengan mikrokontroller keluarga MCS51 yang memiliki dimensi lebih kecil (misalnya AT89C2051) agar dapat diimplementasikan pada sistem yang memiliki keterbatasan ruang.

2. Aplikasi sistem pengelolaan data musik RTTTL dapat ditingkatkan kemampuannya dengan menambahkan fungsi konversi dari format data musik tertentu ke format data musik RTTTL. Bahasa pemrograman Java memiliki pustaka Java™ Sound API yang dapat dipakai untuk meningkatkan kemampuan tersebut.

6. DAFTAR PUSTAKA

----------. 2009. *Ring Tone Transfer Language*, Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Ring_Tone_T ransfer_Language. diakses Agustus 2009.

Schultz, T. 2004, *C and the 8051*, PageFree Publishing.

Parab, Jivan S. Shinde, Santosh A. Shelake, Vinod G. Kamat, Rajanish K. Naik, Gourish M. 2008. *Practical Aspects of Embedded System Design using Microcontrollers*, Springer Science.

Durham, Marcus O. 2004, *Systems Design and the 8051: The hardware, firmware, and software design of microprocessor systems*, 2nd. ed. TechnoPress, Tulsa.