UNIVERSITAS INDONESIA

IMPLEMENTASI SISTEM BANTUAN PENDERITA BUTA WARNA:
INTERAKSI SUARA UNTUK PERANGKAT TERTANAM DENGAN SISTEM OPERASI TERTANAM MICROSOFT

SKRIPSI

RUKI HARWAHYU

0706276103

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KOMPUTER
DEPOK
JULI 2011

UNIVERSITAS INDONESIA

IMPLEMENTASI SISTEM BANTUAN PENDERITA BUTA WARNA: INTERAKSI SUARA UNTUK PERANGKAT TERTANAM DENGAN SISTEM OPERASI TERTANAM MICROSOFT

SKRIPSI

Diajukan sebagai salah satu syarat memperoleh gelar sarjana

RUKI HARWAHYU

0706276103

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KOMPUTER
DEPOK
JULI 2011

HALAMAN PERNYATAAN ORISINILITAS

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

Nama	: Ruki Harwahyu
NPM	: 0706276103
Tanda Tangan	: 
Tanggal	: 4 Juli 2011

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh:

Nama	:	Ruki Harwahyu
NPM	:	0706276103
Program Studi	:	Teknik Komputer
Judul Skripsi	:	Implementasi Sistem Bantuan Penderita Buta Warna: Interaksi Suara untuk Perangkat Tertanam dengan Sistem Operasi Tertanam Microsoft

Telah berhasil dipertahankan di hadapan dewan penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk mata kuliah Skripsi pada program studi Teknik Komputer Fakultas Teknik Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Prof. Dr. Ir. Riri Fitri Sari, M.M., M.Sc. ( )

Penguji : Prof. Dr. –Ing. Ir. Kalamullah Ramli, M.Eng. ( )

Penguji : Prima Dewi Purnamasari, ST., MT., MSc ( )

Ditetapkan di : Depok

Tanggal : 11 Juli 2011

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada:

1. Keluarga Darobi yang tiada henti mencurahkan dukungan dalam berbagai bentuk yang tiada terbalaskan. Segala hal yang telah penulis lewati hingga kini tiada lepas dari doa mereka.
2. Ibu Prof. Dr. Ir. Riri Fitri Sari, M.M., M.Sc. selaku dosen pembimbing, atas segenap teladan, bimbingan, dan pelajaran hidup yang telah ditorehkan kepada penulis, baik selama penulisan skripsi maupun selama masa studi di Teknik Komputer.
3. Teman-teman di program studi teknik komputer dan teknik elektro, khususnya angkatan 2007. Terima kasih atas 4 tahun terakhir yang luar biasa! SUNGGUH!
4. Para peneliti dan komunitas MSDN yang telah membagi dan menyediakan referensi bagi saya.

sehingga penulisan skripsi ini dapat diselesaikan dengan baik.

Depok, 4 Juli 2011

Penulis,

Ruki Harwahyu

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNUTK KEPENTINGAN AKADEMIS

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

Nama	: Ruki Harwahyu
NPM	: 0706276103
Program Studi	: Teknik Komputer
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis Karya	: Skripsi

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

IMPLEMENTASI SISTEM BANTUAN PENDERITA BUTA WARNA: INTERAKSI SUARA UNTUK PERANGKAT TERTANAM DENGAN SISTEM OPERASI TERTANAM MICROSOFT

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

Demikian pernyataan ini saya buat dengan sebenarnya,

Dibuat di: Depok

Pada tanggal: 4 Juli 2011

Yang menyatakan

Ruki Harwahyu

ABSTRAK

Nama : Ruki Harwahyu

Program Studi : Teknik Komputer

Judul : Implementasi Sistem Bantuan Penderita Buta Warna: Interaksi Suara untuk Perangkat Tertanam dengan Sistem Operasi Tertanam Microsoft

Pembimbing : Prof. Dr. Ir. Riri Fitri Sari, M.M., M.Sc.

Fitur suara dapat menjadi alternatif model interaksi pada perangkat tertanam yang dirancang tanpa memiliki banyak tombol kendali, seperti sistem bantu penderita buta warna yang dirancang, yang disebut Chromophore. Skripsi ini membandingkan kinerja fitur suara yang dibuat dengan SAPI5.1 dan fitur suara yang dibuat manual dengan metode penggabungan fonem dan DTW, untuk diimplementasikan pada Chromophore. Skripsi ini juga membandingkan kompatibilitas OS tertanam WinCE6 dan WES09 untuk mendukung fitur suara tersebut. Pengujian fitur suara dilakukan dengan 10 responden untuk mengenali kata-kata yang disintesis sistem dan mengucapkan kata agar dikenali sistem. Pengujian OS dilakukan dengan melihat ukuran, durasi *boot*, dan dukungannya terhadap aplikasi berfitur suara. Dari uji coba tersebut, diketahui bahwa fitur suara yang dibuat dengan SAPI5.1 memiliki kinerja yang lebih baik dibandingkan dengan fitur suara yang dibuat manual, dengan keberhasilan sintesis suara sebesar 88,33% dan pengenalan suara sebesar 75,87% pada kondisi tenang dan 74,76% pada kondisi bising. Pengujian kedua membuktikan WES09 lebih cocok digunakan dikarenakan dukungannya pada .NET 3.5 dan SAPI5.1.

Kata Kunci: pengenalan suara, sintesis suara, Microsoft *Speech* API, OS tertanam, Windows *Embedded*.

ABSTRACT

Name : Ruki Harwahyu

Study Program : Computer Engineering

Title : Implementation of Color Blind Aid System: Speech Feature for Embedded System with Microsoft Embedded Operating System

Supervisor : Prof. Dr. Ir. Riri Fitri Sari, M.M., M.Sc.

Speech feature can be an alternative interaction model for embedded device, which is designed without many buttons for its control, such as color-blind aid system that is designed, namely Chromophore. This paper compares performance of a speech feature created using SAPI5.1 and a speech feature created manually using phone-concatenate and DTW, to be implemented in Chromophore. This paper also compares the compatibility of embedded operating systems, WinCE6 and WES09, to support the speech feature. The testing for speech feature is done using 10 respondents to identify words synthesized by the systems and to say words to be recognized by the systems. The testing for operating systems is done by observing their size, boot time, and their support for the speech feature. As the result, speech feature created using SAPI5.1 is better than the manually-created one, with success rate 88,33% for speech synthesis, 75,87% and 74,76% for speech recognition in silent and noisy condition. The second testing shows that WES09 is more suitable because of its support for .NET 3.5 and SAPI5.1.

Keywords: speech recognition, speech synthesis, Microsoft Speech API, embedded OS, Windows Embedded.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR ISTILAH

Audacity: aplikasi untuk pengolahan suara, yang dalam skripsi ini khususnya digunakan untuk perekaman, pemotongan, dan pengamatan *waveform.*

Byte: ukuran data mentah sepanjang 8-bit.

CAPTCHA: kependekan dari *Completely Automated Public Turing test to tell Computers and Humans Apart*, yakni metode pengecekan apakah pengguna benar-benar manusia atau sistem/robot.

Character: jenis data yang merepresentasikan karakter ASCII dengan masing-masing berukuran maksimal 8-bit.

Chromophore: nama yang diberikan untuk sistem bantu penderita buta warna yang dibuat dengan fitur suara dan sistem operaasi tertanam yang dirancang.

Difon: satuan bunyi yang terdiri dari dua fonem.

Euclidean: istilah matematika untuk jarak terpendek antara dua titik. Pada sistem Cartesian, jarak ini adalah akar dari jumlah selisih-selisih representasi titik pada tiap ruang yang dikuadratkan.

Fonem: satuan bunyi terkecil.

Integer: jenis data yang merepresentasikan angka bertanda berukuran maksimal 32-bit.

Namespace: wadah abstrak yang menyediakan fungsionalitas yang tersedia di dalamnya. Dalam pemrograman berorientasi objek, *namespace* mirip seperti *package*. Di dalamnya bisa terdapat *namespace*/*package* atau kelas.

Short: jenis data yang merepresentasika angka *integer* bertanda dengan ukuran maksimal 16-bit.

Task Manager: istilah di sistem operasi Windows untuk aplikasi bawaannya yang muncul saat pengguna menekan CTRL+ALT+DEL. Aplikasi ini berguna untuk diagnosis sederhana.

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Warna memegang peran yang sangat penting dalam kehidupan manusia. Selain bentuk dan gelap terang, warna merupakan fitur visual yang juga dapat secara signifikan menyampaikan informasi tertentu. Meskipun demikian, ada sekitar 8% pria dan 1% wanita yang terlahir dengan keterbatasan dalam menterjemahkan keberagaman warna [1]. Mereka adalah penderita buta warna, dan buta warna ini merupakan kelainan bawaan yang tidak dapat dipulihkan.

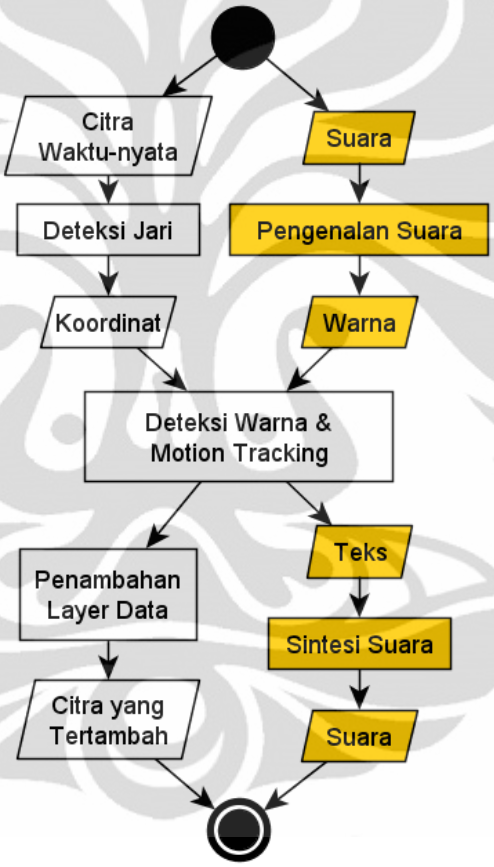
Meskipun jumlah penyandang buta warna tidak terlalu signifikan dibandingkan kelainan lain, dinilai perlu adanya usaha untuk membantu para penyandang buta warna agar bisa mengatasi keterbatasan mereka, utamanya agar mereka memiliki kesempatan yang sama dalam kehidupan sehari-hari dengan orang berpenglihatan normal. Atas dorongan ini, dirancanglah sebuah prototipe sistem bantuan bagi penyandang buta warna agar mereka mampu mengenali warna sebagaimana mestinya.

Sistem ini nantinya akan dikemas dalam perangkat *head mounted display* (HMD) agar mudah digunakan untuk keperluan sehari-hari. Fitur suara dirancang sebagai media interaksi dengan sistem, agar perangkat ini semakin ringkas (tidak membutuhkan banyak tombol) dan intuitif.

Agar pengembangan aplikasi ini dapat diimplementasikan secara luas, khususnya di dunia industri perangkat tertanam (*embedded*), sistem ini akan diletakkan di atas sistem operasi (OS) Windows *Compact Embedded* (WinCE). Berdasarkan hasil survey yang dirilis EE Times tentang OS tertanam [2], sebagian besar pengembang memilih OS komersial, dan dari 25 OS tertanam yang paling banyak digunakan, WinCE berada di urutan kedua dengan pangsa pasar sekitar 13% (setelah VxWorks dengan pangsa pasar sekitar 18%). Selain WinCE, Windows *Embedded Standard* (WES) juga digunakan untuk hal yang sama demi menguji kompatibilitas aplikasi.

1.2. Perumusan Masalah

Sistem bantuan penderita buta warna yang akan dirancang bernama Chromophore. Aspek yang menjadi fokus dalam skripsi ini adalah pengembangan fitur sintesis suara dan perintah suara untuk menyediakan antarmuka interaktif bagi pengguna (bagian berwarna kuning pada Gambar 1.1), dan perancangan OS untuk perangkat tertanam sebagai target *deployment* dari sistem Chromophore.



Gambar 1.1. Diagram Alur Kerja Aplikasi Chromophore

1.3. Tujuan Penelitian

Tujuan dari penelitian yang tertuang dalam skripsi ini adalah untuk mengetahui pengembangan fitur suara dan OS tertanam yang tepat untuk diimplementasikan pada sistem Chromphore, dengan menguji beberapa alternatif

yang sudah tersedia dan cukup sering digunakan orang dalam pengembangan hal yang serupa.

1.4. Batasan Masalah

Masalah yang akan dibahas pada penulisan skripsi ini hanya meliputi tahapan dan analisis implementasi fitur suara dengan Microsoft *Speech Application Programming Interface* versi 5.1 (SAPI5.1) sebagai *engine* fitur suara terbaik [3], implementasi fitur suara sederhana untuk platform *Compact Framework*, serta pengujian kompatibilitas *Windows Compact Embedded* versi 0.6 (WinCE6) dan *Windows Embedded Standard* versi 2009 (WES09) untuk OS eBox-3310A-JSK (selanjutnya disebut eBox) sebagai perangkat keras untuk emulasi HMD.

1.5. Metode Penelitian

Metode penelitian yang digunakan dalam penulisan skripsi ini meliputi:

1. tinjauan pustaka, dengan melakukan studi literatur dari buku-buku, karya tulis (*paper*), situs Internet, dan buku panduan dari perangkat yang digunakan.
2. diskusi, dengan pembimbing dan teman-teman yang berkaitan topik bahasan skripsinya, ataupun yang lebih berpengalaman.
3. korespondensi, lewat forum-forum di Internet untuk menggali informasi dari komunitas pengembang teknologi terkait.
4. perancangan OS, modul fitur suara, dan integrasi perangkat lunak.
   observasi dengan sampel *convenience*, uji coba dengan sampel *purposive*, dan wawancara terstruktur.

Pada dasarnya penelitian yang dilakukan mengacu pada perancangan dan pembuatan prototipe OS dan sistem perangkat lunak melalui prosedur sebagai berikut:

1. Pengembangan aplikasi Chromophore dan melengkapinya dengan fitur suara.

2. Perancangan OS spesifik (dari segi fitur dan *driver* perangkat keras) berbasis WinCE6 dan WES09.
3. Pemasangan OS beserta aplikasi Chromophore pada eBox.
4. Pengujian performa modul fitur suara, performa sistem, dan efektifitas aplikasi.

1.6. Sistematika Penulisan

Sistematika dari penulisan skripsi ini adalah sebagai berikut:

BAB 1 PENDAHULUAN

Bab Pendahuluan ini berisi gambaran umum mengenai pembahasan penelitian yang dilakukan. Bagian ini mencakup latar belakang pemilihan tema, perumusan masalah, tujuan penelitian, batasan masalah, metode penelitian, dan sistematika penulisan.

BAB 2 DASAR TEORI

Bab 2 membahas mengenai teori-teori yang mendasari penelitian ini untuk memberikan pengantar mengenai buta warna, karakteristiknya, dan permasalahan yang umumnya dihadapi oleh penyandang buta warna. Bab ini kemudian dilanjutkan dengan menjelaskan lebih lanjut mengenai pengembangan perangkat tertanam dan pengolahan suara dengan teknologi *Microsoft*, dan penjelasan bagaimana program sintesis dan pengenalan suara sederhana dapat dibuat.

BAB 3 PERANCANGAN SISTEM

Bab 3 membahas rancang bangun salah satu bagian dari sistem bantuan penderita buta warna yang berisi tahapan persiapan dan konfigurasi, serta rancangan cara kerja program yang dilengkapi dengan diagram-diagram *Unified Modelling Language* (UML).

BAB 4 IMPLEMENTASI SISTEM

Bagian ini menguraikan teknis implementasi secara urut, mulai dari implementasi fitur suara dengan .NET, dengan .NETCF, OS WinCE6, dan WES09.

BAB 5 UJI COBA DAN ANALISIS

Bagian ini berisi tahapan pengujian yang dilakukan terhadap sistem dan analisis dari data hasil uji coba tersebut.

BAB 6 PENUTUP

Bab 6 berisi penutup dari penulisan skripsi, yaitu berupa kesimpulan dan saran yang berguna untuk dilakukan pada penelitian selanjutnya.

BAB 2
TINJAUAN PUSTAKA

2.1. Kelainan Buta Warna

Buta warna adalah kelainan bawaan atau keturunan yang dibawa oleh kromosom X orang tua yang menyandang buta warna, dan ini bukanlah suatu penyakit. Buta warna merupakan kelainan bawaan, tidak ada obat untuk menyembuhkannya.

Buta warna dikelompokkan dalam dua jenis, buta warna total dan buta warna sebagian. Penderita buta warna yang hanya dapat melihat dengan warna hitam, abu-abu, dan putih saja adalah buta warna total, sedangkan buta warna parsial tidak dapat membedakan warna-warna tertentu tergantung dari jenis buta warna parsialnya.

Retina mata memiliki dua sel yang diperlukan untuk melihat benda, yaitu sel batang (*rod*) yang sensitif terhadap cahaya dan sel kerucut (*conus*) yang sensitif terhadap warna [4]. Penderita buta warna memiliki perbedaan dengan mata normal dalam hal dua sel ini. Pada penderita buta warna parsial, sel kerucut tidak berfungsi sempurna. Pada penyandang buta warna total, retinanya tidak memiliki sel kerucut sehingga matanya tidak sensitif terhadap warna.

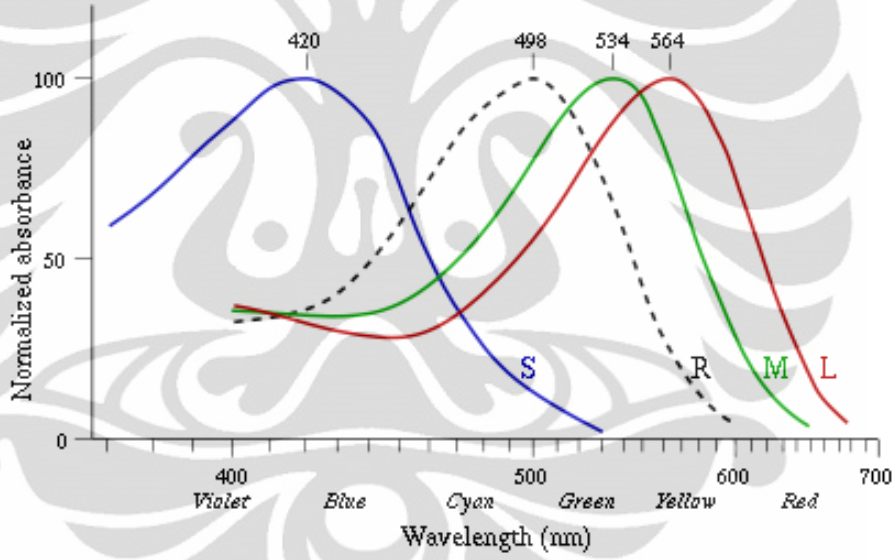
Berdasarkan beberapa sumber, diperkirakan 9%-12% pria dari total populasi di dunia mengalami defisiensi penglihatan terhadap warna atau buta warna [5]. Sebuah penelitian lain menyebutkan bahwa terdapat sekitar 8% pria dan kurang dari 1% wanita yang memiliki kesalahan persepsi warna sejak lahir [1]. Bahkan, menurut data statistik Pusat Kesehatan Nasional di Amerika, yang dihimpun oleh *Center for Disease Control* (CDC), terdapat sekitar 3,4 juta penderita buta warna di Amerika Serikat [6]. Adapun data-data tersebut adalah data untuk semua kategori atau jenis buta warna, baik buta warna total maupun parsial.

2.1.1. Buta Warna Parsial

Buta warna parsial artinya tidak dapat melihat warna tertentu. Kelainan ini disebabkan oleh kelainan genetik pada sel-sel kerucut mata sebagai sel utama

yang mampu menterjemahkan panjang gelombang yang berbeda-beda sebagai representasi warna.

Ada tiga jenis sel kerucut yang ada pada mata manusia, yakni sel kerucut yang peka terhadap panjang gelombang panjang, menengah, dan panjang gelombang pendek. Ketiga panjang gelombang tersebut berkaitan dengan tiga warna utama, yakni merah (gelombang panjang), hijau (gelombang menengah) dan biru (gelombang pendek). Dengan kemampuan reseptor menerima tiga warna dengan panjang gelombang yang berbeda-beda tersebut, sistem penglihatan ini disebut sebagai trikromasi (*trichromatic vision*) [7]. Berikut ini visualisasi perbedaan persentase daya serap tiap sel reseptor tersebut pada gelombang cahaya tampak [8]:



Gambar 2.1. Perbedaan Daya Serap Sel Kerucut (S,M,L) dan Sel Batang (R)

Buta warna parsial selanjutnya dibagi lagi menjadi dikromasi dan anomali trikromasi. Dikromasi disebabkan karena ketiadaan salah satu dari tiga sel kerucut yang seharusnya dimiliki, sedangkan anomali trikromasi disebabkan karena ketidaksempurnaan pada ketiga sel kerucut yang ada. Masing-masing, baik dikromasi maupun anomali trikrommasi dapat dibagi lagi menjadi tiga warna pokok kebuta-warnaannya, sehingga jenis buta warna parsial ada 6 [9], yaitu:

1. Dikromasi merah (buta warna merah).
2. Dikromasi biru (buta warna biru).

3. Dikromasi hijau (buta warna hijau).
4. Protanomali (defisiensi merah, menyebabkan tidak bisa membedakan warna merah dan hijau).
5. Deutranomali (defisiensi hijau, menyebabkan tidak bisa membedakan warna hijau dan merah tertentu).
6. Tritanomali (defisiensi biru, menyebabkan tidak bisa membedakan warna biru dan kuning tertentu).

Kasus anomali trikromasi lebih sering ditemui dari pada dikromasi. Secara umum, penyandang anomali trikromasi akan sulit membedakan warna merah, kuning, atau hijau dan paduan-paduannya, khususnya untuk warna dengan intensitas rendah.

2.1.2. Buta Warna Total

Buta warna total berarti tidak bisa melihat warna sama sekali. Penderitanya hanya bisa melihat gelap terang (hitam, abu-abu, dan putih), seperti gambar *greyscale*. Buta warna jenis ini diakibatkan karena tidak berfungsinya sel-sel kerucut yang ada pada retina mata [10]. Jenis buta warna seperti ini cukup jarang ditemui.

2.1.3. Kesulitan yang Dihadapi Penyandang Buta Warna

Dari banyaknya jumlah penderita buta warna yang telah disebutkan, hanya sedikit persentase jumlah penyandang buta warna total. Namun, terlepas dari hal tersebut, semua penyandang buta warna jenis apapun tentu memiliki masalah dalam kehidupan sehari-harinya. Banyak orang yang tidak mengetahui bahwa orang yang menderita buta warna parsial juga memiliki permasalahan yang terkait dengan kelainan penglihatannya terhadap warna. Berikut merupakan beberapa masalah yang dihadapi oleh beberapa penyandang buta warna [11]:

1. Masalah sehari-hari.

   Dampak kelainan buta warna tentu dihadapi oleh penyandang buta warna dalam kehidupannya sehari-hari. Misalnya dalam membedakaan warna pakaian, warna lampu lalu lintas saat berkendara, warna simbol-simbol tertentu, dan lain sebagainya.

2. Masalah bidang pendidikan.

   Buta warna mempengaruhi penderitanya dalam memilih program studi untuk melanjutkan pendidikan dan karir. Beberapa pilihan program studi dan pekerjaan mensyaratkan mahasiswa atau karyawannya tidak buta warna.

3. Masalah psikologis.

   Diskriminasi terhadap penyandang buta warna masih sering terjad di wilayah-wilayah tertentu. Sebagian orang menganggap bahwa buta warna adalah penyakit. Selain itu, ketidak-mampuan dalam membedakan warna sering kali menjadi bahan ejekan, bahkan hinaan yang dapat membuat penyandang buta warna merasa minder dan tidak percaya diri.

2.2. Teknologi Realitas Tertambah

Teknologi realitas tertambah, atau yang lebih populer dengan istilah *Augmented Reality* (AR) merupakan sebuah teknologi yang menambahkan objek baru pada kondisi nyata dan memadukannya dalam suatu kemasan secara komputer [12]. Belakangan, teknologi ini sering kali digunakan untuk fitur-fitur multimedia dengan menambahkan objek 3 dimensi, meskipun masih banyak model pemanfaatan lain dari teknologi ini.

AR merupakan pengembangan dari *virtual reality* (VR), yang dimengerti kebanyakan orang sebagai simulasi elektronik dari sebuah lingkungan yang dapat dirasakan pengguna, sehingga pengguna merasakan keadaaan yang realistis [13]. Alih-alih mensimulasikan, teknologi AR lebih banyak bekerja untuk ‘menambahkan’ informasi-informasi tertentu ‘di atas’ objek nyata yang ditangkap oleh perangkat pengindera komputer. AR tidak sepenuhnya menggantikan lingkungan nyata, sebagaimana konsep yang dipakai pada VR, karena objek nyata dari lingkungan asli yang ditangkap masih memiliki peran yang penting dalam sistem AR [14]. Dengan teknologi AR, pengguna dapat menikmati model interakasi baru terhadap objek nyata di sekitarnya, karena AR menambahkan informasi (bisa berupa objek) tertentu secara *real-time* dan membuat objek tambahan tersebut seolah-olah bagian dari lingkungan yang nyata.

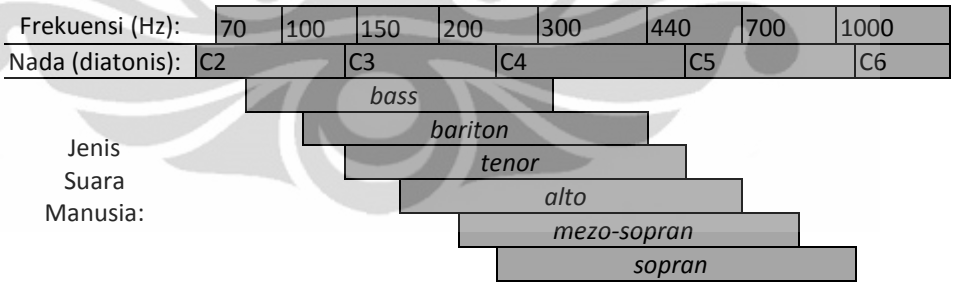
2.3. Pola Interaksi Suara Lisan (*Speech*) pada Sistem Komputer

Bunyi, derau (*noise*), suara, dan percakapan/lisan/ucapan (*speech*) merupakan hal yang berbeda bagi manusia. Bunyi adalah sembarang getaran yang merambat melalui udara dan diindera oleh telinga. Derau adalah bunyi yang tidak jelas, cenderung mengganggu dan tidak diinginkan. Suara adalah bunyi yang dihasilkan oleh pita suara (*larynx*) manusia dan diucapkan lewat mulut [15]. Bunyi lisan adalah ucapan manusia yang terdiri dari kata-kata mengandung informasi tertentu sebagai salah satu media dalam berkomunikasi [16].

Pada kenyataannya, pemahaman tersebut tidaklah mudah bagi sistem komputer pengolah bunyi. Komputer hanya mengenal bunyi, entah itu derau, ataukah ucapan. Meskipun itu adalah bunyi lisan, komputer melihatnya sebagai proses kontinu tanpa pembeda antar unit/kata yang jelas.

2.3.1. Suara Manusia dan Suara Digital

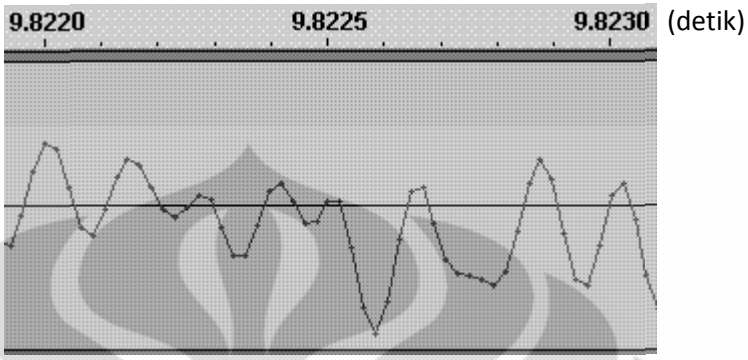
Suara manusia diproduksi oleh pita suara, dengan frekuensi umumnya berada pada rentang 80 Hz sampai 1100 Hz dan bersifat analog. Dalam musik, suara manusia diklasifikasikan berdasarkan suara pria dan wanita. Suara pria pun jenisnya diklasifikasikan lagi menjadi *bass*, *bariton*, *tenor*, dan *countertenor*. Suara wanita dibedakan menjadi *contralto*, *mezzosopran*, dan *sopran* [17].



Gambar 2.2. Klasifikasi Suara Manusia dalam Musik

Suara digital dapat secara umum disebut bunyi, karena komputer menyimpan semua bunyi dalam representasi gelombangnya (*waveform*), namun dalam format penyimpanan digital. Terdapat beberapa atribut yang melekat pada suatu data suara digital, di antaranya adalah amplitudo, teknik dan frekuensi pencuplikan (*sampling*), teknik dan skala kompresi, kanal (mono, stereo, atau lebih), dan *bit depth* (banyaknya bit yang digunakan untuk menyimpan hasil kuantisasi suara analog). Suara digital tak lain adalah nilai yang disimpan untuk

tiap satuan waktu. Visualisasi ini menunjukkan data suara selama sekitar 1 milidetik yang diambil dengan aplikasi Audacity.



Gambar 2.3. Satu Milidetik Suara Digital Dilihat dengan Audacity

Suara digital disimpan dalam format data tertentu, dan untuk diproses kembali, format yang paling umum digunakan adalah WAV. WAV menyimpan audio dengan teknik *Pulse Code Modulation* (PCM). WAV menyimpan hasil PCM ini tanpa kompresi, dan menambahkannya dengan beberapa informasi (termasuk *metadata*) sebagai *header*. Struktur berkas WAV secara urutan byte adalah sebagai berikut [18]:

Tabel 2.1. Standar Struktur Berkas WAV

Byte ke-	**Nama *Field***	**Isi**	**Format data**
0-3	*ChunkID*	teks "RIFF"	*Character*
4-7	*Chunk Size*	Banyak byte untuk berkas WAV ini dikurangi 8.	32-bit *Integer* bertanda
8-15	Format dan *Subchunk1 ID*	Teks "WAVEfmt " (dengan spasi di akhir)	*Character*
16-19	*Subchunk1 Size*	"16" untuk format PCM	32-bit *Integer* bertanda
20-21	Format Audio	"1" untuk format PCM	*Short*
22-23	*Channel*	Jumlah kanal audio	*Short*
24-27	*Sample Rate*	Frekuensi sampling	32-bit *Integer* bertanda
28-31	*Byte Rate*	Jumlah byte per detik	32-bit *Integer* bertanda
32-33	*Block Align*	Jumlah byte tiap sampel	*Short*
34-35	*Bits Per Sample*	Jumlah bit tiap sampel	*Short*
36-39	*Subchunk2 ID*	teks "data"	*Character*
40-43	*Subchunk2 Size*	Banyak byte untuk data audio mentah (RAW)	32-bit *Integer* bertanda
44…	data	Data audio mentah	*Byte*

2.3.2. CAPTCHA

CAPTCHA merupakan suatu metode untuk memastikan suatu respon/perlakuan dilakukan oleh manusia, bukan mesin/sistem. CAPTCHA dihasilkan secara acak oleh *server* dan *server* itu pula yang menilai kebenaran masukan dari pengguna.

CAPTCHA merupakan salah satu bentuk interaksi manusia dengan komputer menggunakan suara. Seiring perkembangan teknologi, jenis CAPTCHA saat ini semakin beragam dan dilengkapi dengan fitur bantuan suara yang utamanya disediakan untuk orang dengan gangguan penglihatan. Suara yang dihasilkan oleh sistem CAPTCHA biasanya merupakan ucapan yang dipadukan dengan derau.

Derau pada ucapan CAPTCHA dibuat hanya untuk menyamarkan agar sistem pengenalan suara sulit mengenalinya. Namun tingkat gangguan ini tidak merusak fungsi CAPTCHA. Derau tersebut telah disesuaikan sedemikian rupa amplitudonya sehingga walaupun dengan derau, manusia tetap dapat mengenali ucapan CAPTCHA aslinya.

2.3.3. Fonem

Bagian terkecil yang masih menunjukkan perbedaan makna dari bentuk tulisan adalah huruf, sedangkan bagian terkecil dari bentuk suara/lisan adalah fonem (atau *phone*). Fonem tidak identik dengan huruf karena gabungan huruf "n" dan "g", misalnya, menjadi fonem "ng" yang pengucapannya berbeda dari pengucapan “n” maupun “g”. Jumlah fonem lebih banyak daripada jumlah huruf. Dari alfabet yang kita kenal, ada sumber yang mengatakan bahwa Bahasa Indonesia memiliki 35 fonem [19] dan ada juga yang mengatakan 30 [20]. Berikut ini tabel pendefinisian 30 fonem dalam bahasa Indonesia:

Tabel 2.2. Pendefinisian Fonem Bahasa Indonesia Menjadi 30 Fonem

Fonem	Contoh
a	**a**man
i	**i**kan
u	**u**ntuk
e	s**e**mua

Fonem	Contoh
au	pant**au**
p	**p**alu
b	**b**ubur
t	**t**angan

Fonem	Contoh
j	**j**auh
f	**f**ana
s	**s**arana
z	**z**aman

Fonem	Contoh
ny	**ny**awa
r	**r**asa
l	**l**ama
w	**w**ajan

e	m**e**dan	d	**d**uka	h	**h**ujan	y	ja**y**a
o	p**o**los	k	**k**apas	m	**m**akam	_	_ (diam)
ai	duh**ai**	g	**g**una	n	**n**ama		
oi	amb**oi**	c	**c**ari	ng	bu**ng**a		

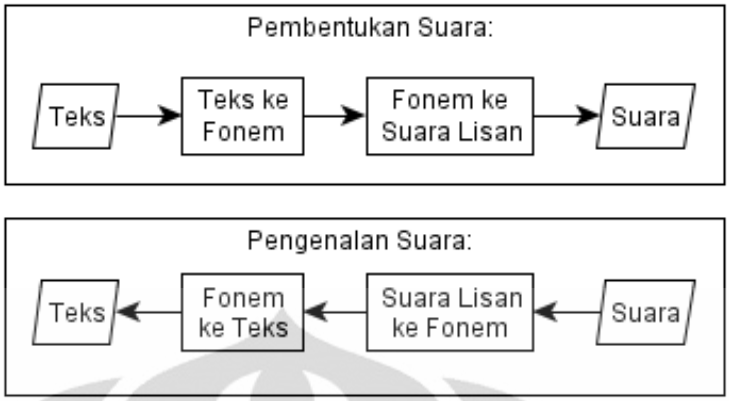
2.3.4. Difon

Difon (*diphone*) adalah dua fonem yang berurutan, dan dari semua fonem yang didefinisikan, jumlah difon yang ada merupakan kombinasi 2 dari semuanya. Dalam bahasa Indonesia, dari 35 fonem yang ada, dapat tercipta 1.296 difon, termasuk difon "diam" untuk awal dan akhir kata [21].

Metode pengolahan suara lisan secara komputer dapat menggunakan pendekatan difon. Sistem pengolah suara lisan akan mendeteksi makna dari masukan (suara pada pengenalan suara, dan teks pada sintesis suara) dengan mencocokkan antara difon dengan 2 huruf yang runtut pada basisdata. Dengan perbedaan difon pada setiap bahasa, setiap sistem pengolah suara lisan, khususnya yang menggunakan metode pengelompokan difon, harus memiliki basisdata yang berbeda untuk mampu mengolah suara lisan dari bahasa yang lain.

2.3.5. Pemrosesan Suara Lisan pada Komputer

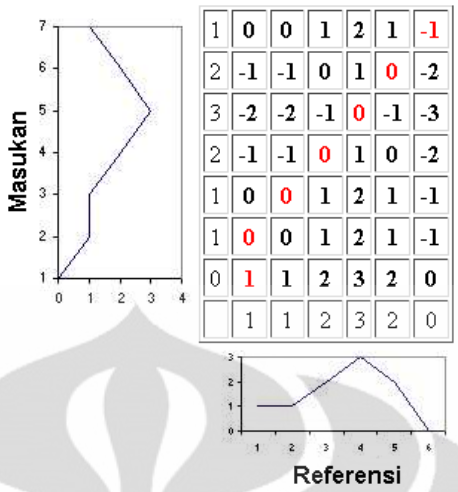
Setiap satuan suara lisan di komputer, agar dapat diproses, harus memiliki 3 parameter; pertama, kode sampa yaitu kode yang mewakili sebuah fonem atau difon. Kedua, durasi yaitu lama pengucapan, dan ketiga intonasi yaitu frekuensi, atau tinggi nada pengucapan [17]. Pengolahan secara lebih lanjut yang dilakukan secara umum melibatkan dua tahap utama, yaitu konversi teks ke fonem dan sebaliknya, serta konversi fonem ke suara dan sebaliknya.



Gambar 2.4. Tahapan Umum Pengolahan Suara Lisan pada Komputer

Untuk dapat mensintesis suara, diperlukan *database* yang mengasosiasikan setiap kata dengan fonem (bila menggunakan pendekatan fonem) atau difonnya (bila menggunakan pendekatan difon). Tentunya data ini akan berbeda untuk setiap bahasa. Selain itu, diperlukan juga sampel suara untuk setiap fonem atau difon yang ada. Sistem akan mencocokkan kata yang menjadi input dengan database tesebut, dan selanjutnya merangkai semua sampel suara sesuai fonem atau difon yang sesuai dengan kata tersebut. Penyusunan berdasarkan fonem langsung dilakukan begitu saja. Sedangkan penyusunan berdasarkan difon melibatkan pemotongan bagian-bagian tertentu. Rangkaian sampel suara ini kemudian dapat langsung diputar secara urut atau disatukan terlebih dahulu dalam satu *stream* [22]. Pemutaran langsung sesuai urutan melibatkan proses yang lebih banyak, sedangkan penyusunan menjadi *stream* membutuhkan *buffer* lebih besar.

Di sisi lain, untuk pengenalan suara, algoritma yang secara luas digunakan meliputi *Hidden Markov Model* (HMM), *Artificial Neural Network* (ANN), dan *Dynamic Time Wrapping* (DTW). Dari ketiga algoritma tersebut, DTW merupakan algoritma yang paling sederhana. DTW populer digunakan untuk pengenalan suara skala kecil, untuk perangkat tertanam, dan sistem lainnya yang tidak memiliki/mengharuskan komputasi yang rumit.



Gambar 2.5. *Path* antara Dua Gelombang pada Perhitungan DTW

DTW merupakan algoritma yang bertujuan untuk mencari kemiripan antara dua gelombang dalam domain waktu, yaitu masukan dan referensi. DTW memekarkan (*stretch*) atau mengkompresi (*compress*) gelombang masukan sesuai dengan panjang gelombang referensi [23], dan selanjutnya menghitung kemiripan keduanya berdasarkan *path*. *Path* merupakan kumpulan jarak *euclidean* terkecil antara setiap sinyal dalam kedua gelombang, yang sedekat mungkin membentuk diagonal. Selanjutnya, semua komponen yang membentuk *path* ini dijumlahkan untuk mendapatkan jarak global (*global distance*).

Pada sistem pengenalan suara, sebuah gelombang masukan dibandingkan dengan setiap gelombang referensi. Hasilnya, gelombang referensi yang menghasilkan jarak global terkecil dianggap sebagai gelombang yang sesuai dengan gelombang masukan.

2.4. Pemanfaatan Teknologi *Microsoft* untuk Pengembangan

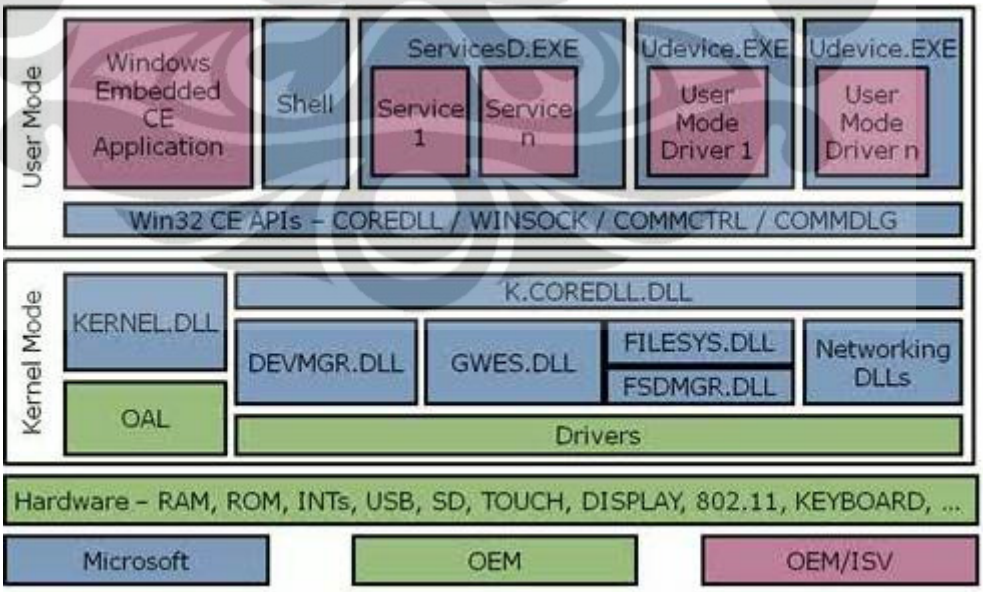
*2.4.1. Windows Compact Embedded 6.0 (*WinCE*6)*

WinCE adalah sebuah OS waktu nyata (*real-time*) untuk sistem tertanam (*embedded system*) yang sengaja dibangun dari awal, dan WinCE termasuk dalam keluarga OS *Microsoft Windows*. WinCE bukan *Windows Mobile* seperti yang dianggap oleh kebanyakan orang. *Windows Mobile* adalah OS yang dibangun di atas infrastruktur WinCE. Demikian juga, WinCE bukanlah *Windows* 95 yang disederhanakan.

WinCE dirancang untuk mewujudkan OS tertanam yang handal dan dapat menangani multiproses, mudah dikembangkan karena terdiri dari banyak modul yang bisa ditambah dan dikurangi sesuai kebutuhan, sekaligus efisien dalam pengembangan dan penggunaan *resource*. *Driver* perangkat keras disediakan melalui *Board Support Package* (BSP) yang terbuka untuk dikembangkan semua orang, termasuk produsen perangkat keras.

Microsoft menyediakan versi evaluasi dari WinCE secara gratis untuk 120 hari, dengan fungsionalitas yang lengkap. Dengan WinCE, pengembang dapat merancang OS dengan mudah, karena proses pembuatan OS berbasis WinCE dilakukan dengan *Platform Builder* yang merupakan *plugin* untuk *Visual Studio* sebagai IDE yang terdiri dari komponen-komponen. WinCE memiliki banyak konfigurasi fungsionalitas dengan *wizard* yang merupakan alat konfigurasi dengan pertanyaan dan pilihan terstruktur yang bersifat menuntun. Terdapat sekitar 600 komponen yang dapat dipilih untuk disertakan dalam OS. Dengan demikian, ukuran OS dapat dibuat lebih kecil dan efisien. Ukuran OS minimal yang dibangun dengan WinCE adalah sekitar 350 KB.

Gambar **2.6** menjelaskan arsitektur WinCE [24].



Gambar 2.6. Arsitektur WinCE6

User-mode adalah tempat dilakukannya eksekusi aplikasi, *shell* (antarmuka), dan berbagai *service*. Di antara *user-mode* dan *kernel-mode*, terdapat

Udevice yang berperan sebagai *proxy* pengatur proses yang menghubungkan *driver* di *kernel-mode* dengan aplikasi di *user-mode* Pada *kernel-mode*, *kernel* menangani akses ke perangkat keras dengan *driver-driver* yang ada. *Kernel* juga menangani *scheduler*, *interrupt*, dan I/O *exception*.

Semua fungsionalitas ini didukung oleh Microsoft, *Original Equipment Manufacture* (OEM) dan *Independen Software Vendor* (ISV) di OEM *Adaptation Layer* (OAL). Komponen OEM/ISV utamanya berperan saat *boot*, yakni menyediakan koneksi antara kernel dengan memori, CPU, *interrupt controller*, dan *clock*.

Berikut ini detil karakteristik WinCE6

- Merupakan OS 32 bit yang dapat menangani banyak proses dan memungkinkan banyak *thread* dalam suatu proses.
- Memiliki memori model *virtual protected memory* yang bukan berupa *swap file*, yang terdiri dari *user virtual memory* dan *kernel virtual memory*, dan dilengkapi perangkat manajemen sistem memori
- Memiliki *Application Programming Interface* (API) yang mirip dengan kernel Win32.
- Mendukung 4 arsitektur CPU: ARM, x86, MIPS, dan SH4.

Di luar karakterisitk tersebut, OS yang dibangun dengan WinCE6 dapat memiliki fitur-fitur berikut ini:

- Multimedia, yaitu DirectShow, Windows Media Player, dan *Digital Rights Management*
- Jaringan, yaitu UPnP, TCP/IPv6, SMB, VPN, *Cellular*, Bluetooth, WiFi, dan VoIP
- Manajemen berkas sistem, yaitu Cache, UDFS, dan *exFat*
- Keamanan, yaitu otentikasi, kriptografi, dan *credential manager*
- Internasionalisasi, yaitu layanan berbasis lokasi, antarmuka pengguna yang multibahasa, dan teks dari kanan ke kiri
- Pengembangan, yaitu .NET *Compact Framework*, *Active Template Library*, MFC8, COM, DCOM, dan *Location Framework*
- Manajemen perangkat, seperti layar penampil, baterai, printer, dll.

2.4.2. Windows Embedded Standard 2009 (WES09)

WES09 adalah OS tertanam yang lebih kompleks dari pada WinCE. Microsoft menargetkan WES09 untuk pasar OS tertanam yang membutuhkan komputasi kompleks dengan pengembangan aplikasi semudah mungkin (mempercepat siklus produksi).

WES09 dibangun berdasarkan Windows XP *Profesional Service Pack* 3. WES09 bukanlah OS waktu nyata, namun tersedia ekstensi dari pihak ke tiga untuk membuat WES09 menjadi OS waktu nyata, dan fitur-fitur tambahan lainnya yang menjadikannya cocok untuk sistem tertanam. WES09 dirancang untuk sistem tertanam yang membutuhkan fungsionalitas yang mirip dengan komputer *desktop*. Di sini, *driver* perangkat keras yang dipakai adalah sama dengan *driver* untuk Windows XP.

Secara umum, pengembangan OS berbasis WES09 hampir sama dengan WinCE. Keduanya sama sama OS yang terdiri dari komponen-komponen, dengan terdapat sekiar 12000 komponen di WES09. Pengembangan WES09 juga dilakukan dengan IDE yang disebut *Windows Embedded Studio*. Dengan IDE ini, WES09 dapat dibangun sedemikian rupa, bahkan hingga tak terlihat seperti OS jika dilihat dari sisi antarmuka pengguna.

Berikut ini detil karakteristik WES09

- Mendukung aplikasi 32-bit maupun 64-bit, multiproses, dan *multithread*.
- Mendukung arsitektur CPU Intel x86 dan x64.
- Memiliki manajemen memori berupa *swap*, manajemen daya, dan manajemen *credential*.
- Memiliki dukungan terhadap aplikasi yang tergantung pada OS Windows secara umum.
- Mendukung komponen-komponen dan fungsionalitas yang umumnnya terdapat di Windows versi *desktop*, seperti Silverlight, Flash, *Active Directory*, bahkan *Multipoint/multitouch*.

Selain karakteristik tersebut, OS yang dibangun dengan WES09 dapat memiliki fitur-fitur berikut ini:

- multimedia
- *browser* Internet
- jaringan
- akses jarak jauh (*remote*)
- manajemen berkas sistem
- manajemen perangkat
- keamanan
- fitur-fitur untuk dukungan pengembangan, seperti .NET, ATL, MFC, COM, DCOM, dan *Location Awareness Framework*

2.4.3. Teknologi Suara (*Speech*) Microsoft

Fitur suara dapat menjadi elemen penting yang menguatkan aspek antarmuka pengguna dari sebuah aplikasi. *Microsoft* sebagai salah satu produsen *software* terkemuka telah lama mengembangkan teknologi fitur suara. Bukan hanya untuk dipakai di produk-produknya saja, *Microsoft* juga menyediakan perangkat fitur suara ini untuk para pengembang aplikasi.

Ada banyak perangkat yang disediakan *Microsoft* untuk para pengembang agar dapat memperkaya aplikasinya dengan fitur suara. Berikut ini adalah produk yang bisa menjadi pilihan [25]:

a. *Speech Application Programming Interface* (SAPI) untuk pengembangan aplikasi dengan komunikasi perangkat tingkat rendah
b. .NET *System.Speech managed namespace* untuk pengembangan aplikasi dengan berbagai bahasa pada *platform* .NET, yang mulai disediakan sejak .NET versi 3.
c. *Microsoft Unified Communication managed* API 2.0 untuk pengembangan aplikasi di *server* yang melayani interaksi suara.
d. *Speech Server* untuk pengembangan aplikasi di *server* yang disediakan untuk *Windows Server* sejak 2007
e. *Tellme Studio* untuk pengembangan aplikasi dengan fitur suara yang berbasis *VoiceXML* berstandar W3C.

Pemilihan mana produk yang akan digunakan tergantung pada lingkungan pengembangan aplikasinya. Dari dua jenis produk yang disediakan untuk

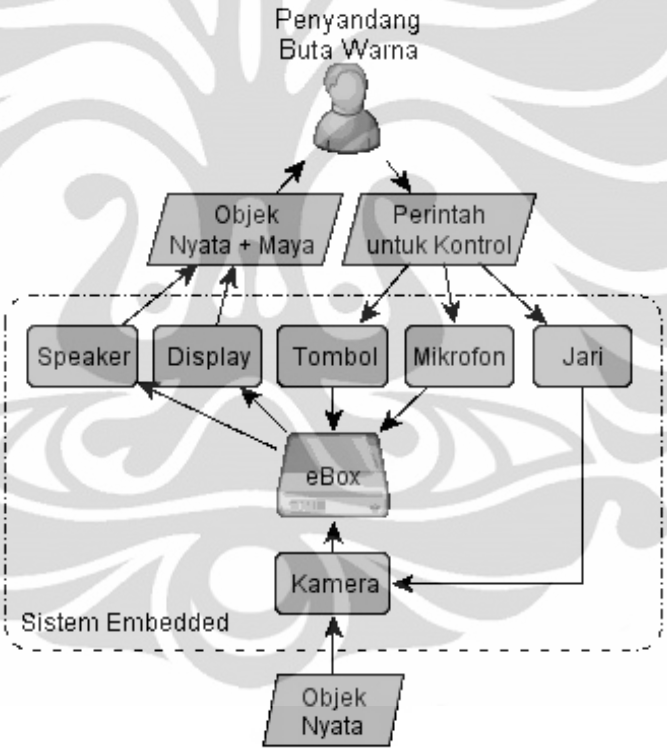
pengembangan aplikasi *desktop*, penelitian ini menggunakan .NET *System.Speech managed namespace*.

Berbeda dengan *Microsoft* SAPI yang lebih ditujukan untuk pemrograman tingkat rendah berbasis *Component Object Model* (COM) dengan bahasa pemrograman C++, *Microsoft* .NET *System.Speech managed namespace* disediakan untuk pengembangan fitur suara dengan lebih mudah pada lingkungan yang telah diatur di atas arsitektur bangunan .NET, yaitu .NET *managed environment*. Selain menawarkan pemrograman yang lebih mudah, .NET juga menjamin kompatibilitas aplikasi agar bisa dijalankan di atas semua OS *Windows* yang memiliki versi .NET yang sama [26]. *System.Speech namespace* ini sudah menyediakan semua metode untuk berinteraksi dengan suara, seperti pengenalan suara, sintesis suara, pengunaan *grammar library*, dan *speech event handler*.

BAB 3
PERANCANGAN SISTEM

3.1. Deskripsi Sistem

Sistem utama yang menjadi fokus dalam skripsi ini adalah sistem tertanam sebagai perangkat fungsi tunggal yang terdedikasi untuk membantu penyandang buta warna. Selain dikembangkan di atas perangkat yang berbeda, pengembangan aplikasi untuk perangkat tertanam ini juga memperhatikan kompatibilitasnya dengan perangkat keras yang digunakan.

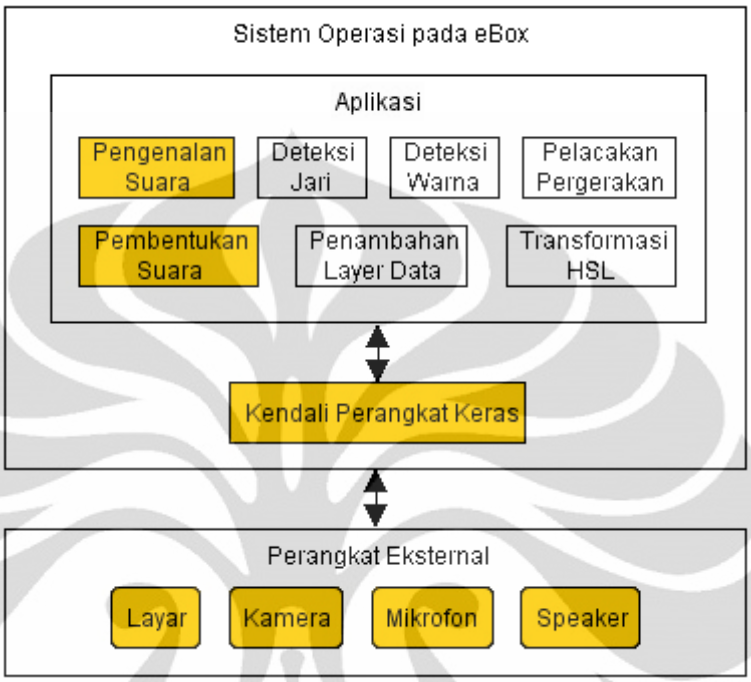


Gambar 3.1. Aliran Data pada Sistem Tertanam

Aplikasi untuk perangkat tertanam ini akan dikembangkan sebagai prototipe di atas WinCE6 dan WES09 yang berjalan pada eBox dengan pemrograman berbasis *Visual C#* (C# dengan kompatibilitas terhadap .NET).

Aplikasi yang dikembangkan akan menangani beberapa algoritma utama program, menyediakan antarmuka pengguna, dan berkomunikasi dengan OS dalam hal penggunaan perangkat keras yang ada. Selanjutnya, OS ini pun

dirancang sedemikian rupa agar mampu mengendalikan semua perangkat keras yang dibutuhkan.



Gambar 3.2. Skema Sistem Tertanam

3.2. Pemrosesan Suara pada Aplikasi

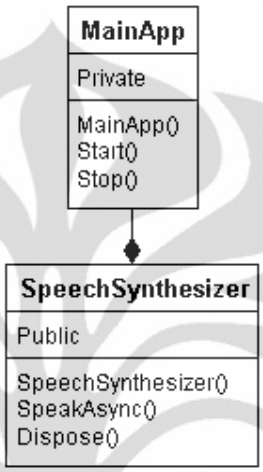
Pengembangan sistem Chromophore ini dilakukan dengan kaidah pemrograman berorientasi objek untuk mempermudah pengembangan suatu sistem besar secara bersama-sama. Pekerjaan dibagi menjadi beberapa modul dan dikerjakan secara terpisah oleh setiap anggota tim.

Pengembangan modul yang dibahas pada skripsi ini adalah modul pemrosesan suara, yangmencakup dua metode pemrosesan suara pada sistem komputer, yakni metode sintesis suara (*speech synthesis*) dan pengenalan suara (*speech recognition*).

3.2.1. Modul Sintesis Suara

Fungsionalitas sintesis suara secara kode disediakan oleh pustaka .NET 3.x (baca: 3 atau lebih), namun *namespace System.Speech* sendiri baru dapat digunakan ketika di lingkungan pengembangan sudah terpasang Microsoft *Speech*

SDK. Hasil kompilasinya baru dapat berfungsi bila di lingkungan pemasangan sudah terpasang Microsoft SAPI. Jadi, bila kebutuhan tersebut telah terpenuhi, untuk menggunakan pustaka sintesis suara dari .NET 3.x, setidaknya struktur kelas paling sederhana yang dapat dipakai adalah:



Gambar 3.3. *Class Diagram* untuk Modul Sintesis Suara

Untuk mengatasi dependensinya terhadap *Speech* SDK dan SAPI, modul sintesis suara juga dapat dibuat sendiri tanpa memanfaatkan *namespace System.Speech* dari .NET. Cara paling sederhana untuk melakukan hal ini adalah dengan tabel *lookup* fonem dari setiap kata dan sampel suara untuk setiap fonem. Program akan mencari kata masukan dalam tabel *lookup* ini dan melihat fonem apa saja yang dipakai. Setelah itu, program akan menyusun setiap sampel suara fonem sesuai urutan fonem yang ditemukan di tabel *lookup* dari langkah sebelumnya. Setelah semuanya tersusun, barulah satu-persatu sampel suara tersebut diputar.

3.2.2. Modul Pengenalan Suara

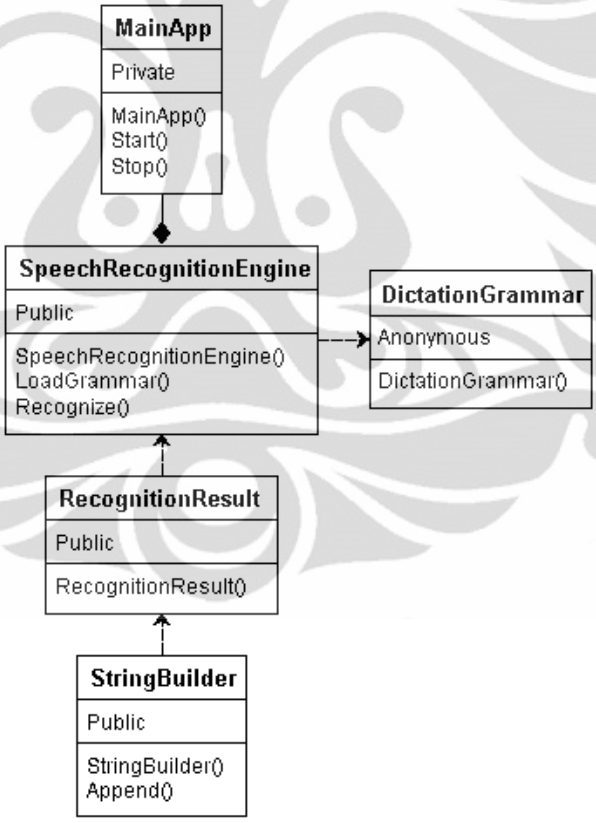
Proses pengenalan suara pada aplikasi komputer bisa jadi sangat rumit. Proses pengenalan suara lebih rumit dibandingkan sintesis suara dari data *string* yang terdefinisi [27].

Dalam skenario perintah suara (*voice command*) yang akan digunakan, terlebih dahulu didefinisikan daftar kata yang terbatas yang dapat dikenali sistem

dalam basisdatanya. Selanjutnya, *engine* pengenalan suara mencocokkan suara yang diucapkan pengguna pada daftar tersebut.

Voice command memungkinkan pengguna mengontrol aplikasi dengan berbicara melalui perangkat masukan suara tanpa menggunakan *mouse* atau *keyboard* yang biasa dipakai di komputer, maka sangat cocok untuk diimplementasikan pada alat ini.

Seperti halnya sintesis suara, fungsionalitas pengenalan suara ini membutuhkan *Speech* SDK di lingkungan pengembangan dan SAPI di lingkungan pemasangan. Bila kebutuhan tersebut telah terpenuhi, untuk menggunakan pustaka pengenalan suara dari .NET 3.x, setidaknya struktur kelas paling sederhana yang dapat dipakai adalah:



Gambar 3.4. *Class Diagram* untuk Blok Pengenalan Suara Sederhana

Untuk mengatasi dependensinya dengan *Speech* SDK dan SAPI, modul pengenalan suara juga dapat dibuat manual dengan memanfaatkan algoritma DTW. Untuk pengembangan dengan bahasa pemrograman C#, tersedia sebuah kelas, bernama *SimpleDTW*, yang dapat digunakan secara bebas untuk melakukan

perhitungan DTW [28]. Kode berikut ini memakai kelas tersebut untuk menghitung jarak global antara dua gelombang, x dan y.

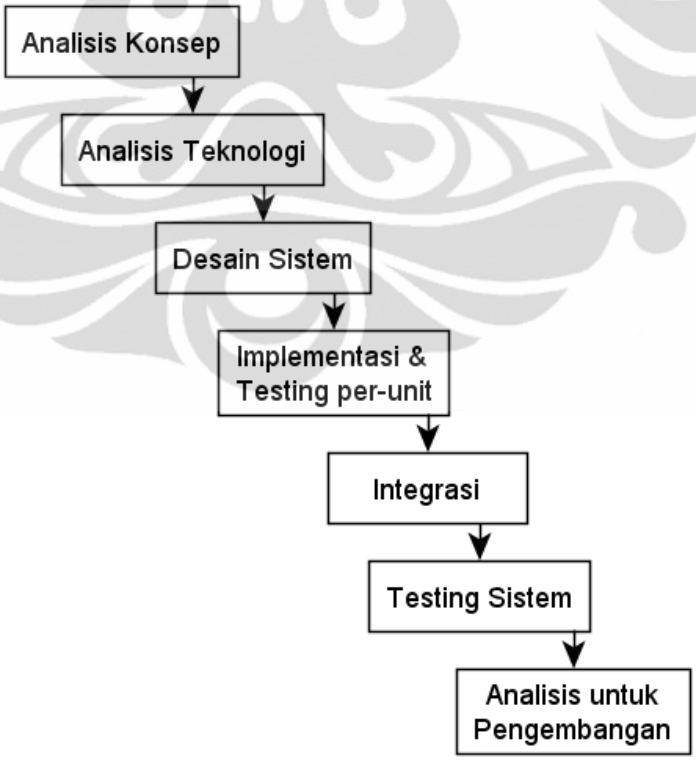
```
//gelombang pertama
double[] x = {9,3,1,5,1,2,0,1,0,2,2,8,1,7,0,6,4,4,5};
//gelombang kedua
double[] y = {1,0,5,5,0,1,0,1,0,3,3,2,8,1,0,6,4,4,5};
//instansiasi kelas SimpleDTW
SimpleDTW dtw = new SimpleDTW(x,y);
//method untuk memulai kalkulasi DTW
dtw.calculateDTW();
//method untuk mengambil hasil kalkulasi DTW
double[] jarak = dtw.getSum()
```

Gambar 3.5. Pemakaian Kelas *SimpleDTW* dalam Program

3.3. Metode dan Desain Pengembangan

3.3.1. Model Pengembangan *Waterfall*

Dalam mengembangkan sistem ini, digunakan pola pendekatan model *Waterfall* [29] sebagai berikut:



Gambar 3.6. Pendekatan Pengembangan Model *Waterfall*

3.3.2. Pemodelan Sistem dengan UML

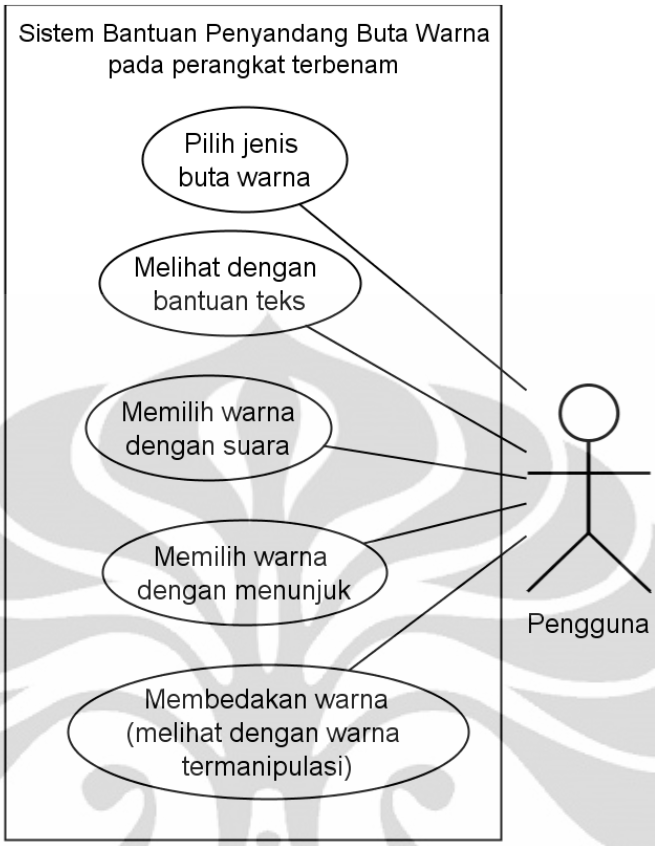
Unified Modelling Language (UML) adalah pemodelan sistem yang digunakan secara universal dalam dunia teknologi informasi (TI). UML menggunakan notasi-notasi yang standar [30] sehingga memudahkan pengembangan proyek TI skala besar yang melibatkan banyak orang, baik pengguna, pelaksana proyek/*programmer*, penganalisa, dan penguji coba.

Terdapat beberapa jenis diagram UML yang biasa digunakan dalam pengembangan perangkat lunak maupun proyek TI lainnya. Setiap diagram memiliki notasi, cara pembuatan, dan fungsinya masing-masing. Semuanya dikombinasikan untuk membangun dokumentasi yang baik.

Penulisan skripsi ini menyertakan beberapa diagram UML sebagai rancangan desain sistem yang dibuat. Diagram tersebut meliputi *use case diagram*, *activity diagram*, *sequence diagram, state diagram, object diagram, deployment diagram* dan *collaboration diagram*.

a. *Use case diagram*

Use case diagram digunakan untuk menggambarkan proses interaksi yang terjadi antara agen dengan fungsionalitas utama sistem.

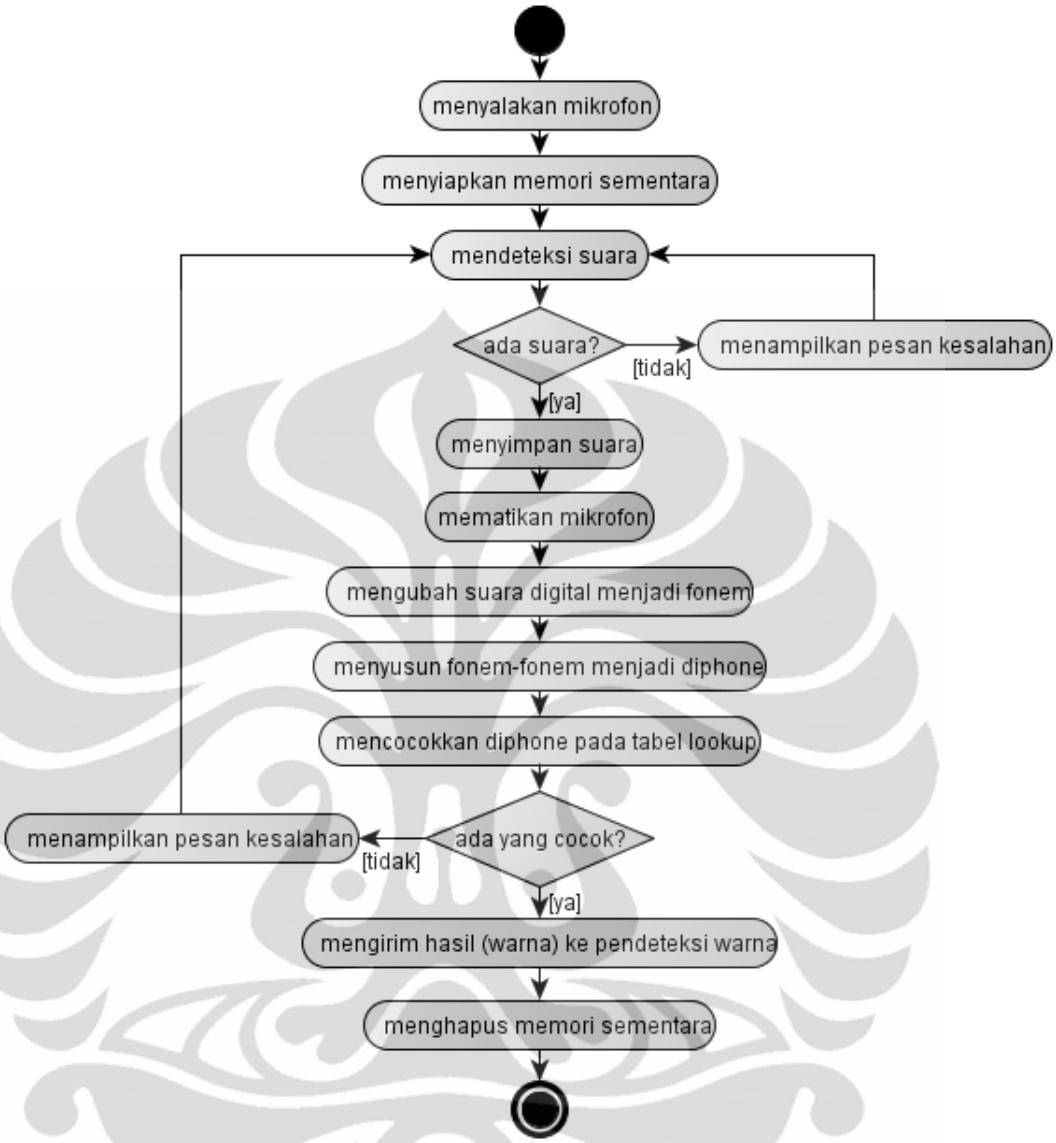


Gambar 3.7. *Use Case* Diagram Chromophore pada Perangkat Tertanam

Dari Gambar 3.7, dapat dilihat bahwa sistem yang ingin dikembangkan memiliki beberapa fungsionalitas, dari memilih jenis buta warna, hingga melihat warna dengan domain warna yang telah termanipulasi untuk dapat melihat warna dengan kontras yang tinggi, sehingga memudahkan penderita buta warna saat harus membedakan warna-warna yang samar.

b. *Activity diagram*

Activity diagram merupakan diagram UML yang digunakan untuk merepresentasikan logika, *business process*, atau alur kerja program.



Gambar 3.8. *Activity Diagram* untuk Modul Pengenalan Suara

Activity diagram pada Gambar 3.8 atas menjelaskan alur kerja dari modul pengenalan suara yang akan diimplementasikan untuk fitur *command voice*, yakni sebagai berikut:

1. Modul pengenalan suara yang awalnya berada pada kondisi *stand-by*, akan diaktifkan saat pengguna menekan tombol pemicu. Tombol pemicu ini ibarat *trigger* pada *handy talky* (HT), yang harus terus ditekan saat memasukkan perintah suara.
2. Bila tombol pemuci diaktifkan, sistem akan menyiapkan mikrofon dan memori sementara untuk penyimpanan suara yang terekam oleh mikrofon.

3. Sistem kemudian mengidentifikasi, apakah benar ada masukan suara dari pengguna. Bila memang ada, suara tersebut akan direkam, dan bila tidak, sistem akan menginformasikan pengguna (dengan *beep*) bahwa sistem tidak mendeteksi adanya suara untuk ditindaklanjuti sebagai perintah.
4. Setelah suara terrekam, untuk menghemat daya, sistem akan mematikan mikrofon.
5. Selanjutnya, algoritma pengolahan suara mulai bekerja, dengan mengubah *stream* suara ke dalam bentuk rentetan fonem yang melibatkan basisdata *Sistem.Speech* yang akan digunakan. Kemudian algoritma akan mengasosiasikan fonem menjadi difon, dan mencocokkan difon tersebut dengan tabel *lookup* yang berisi perintah-perintah yang terdefinisi.
6. Bila tidak ditemukan perintah yang cocok, sistem akan memberi peringatan (dengan *beep* yang lain) bahwa ucapan pengguna tidak dikenal sebagai perintah, dan menyarankan pengguna untuk mencoba lagi.
7. Bila ditemukan data yang cocok, modul ini akan mengirimkan perintah terjemahan dari suara yang terekam pada modul lain yang memegang fungsi utama sebagai pengolah citra, pelacak gerakan, dan pemrosesan realitas tertambah.
8. Setelah perintah terkirim, memori sementara kembali dikosongkan dan modul pengenalan suara menjadi *stand-by*.

Gambar 3.9 adalah *activity diagram* untuk modul sintesis suara.

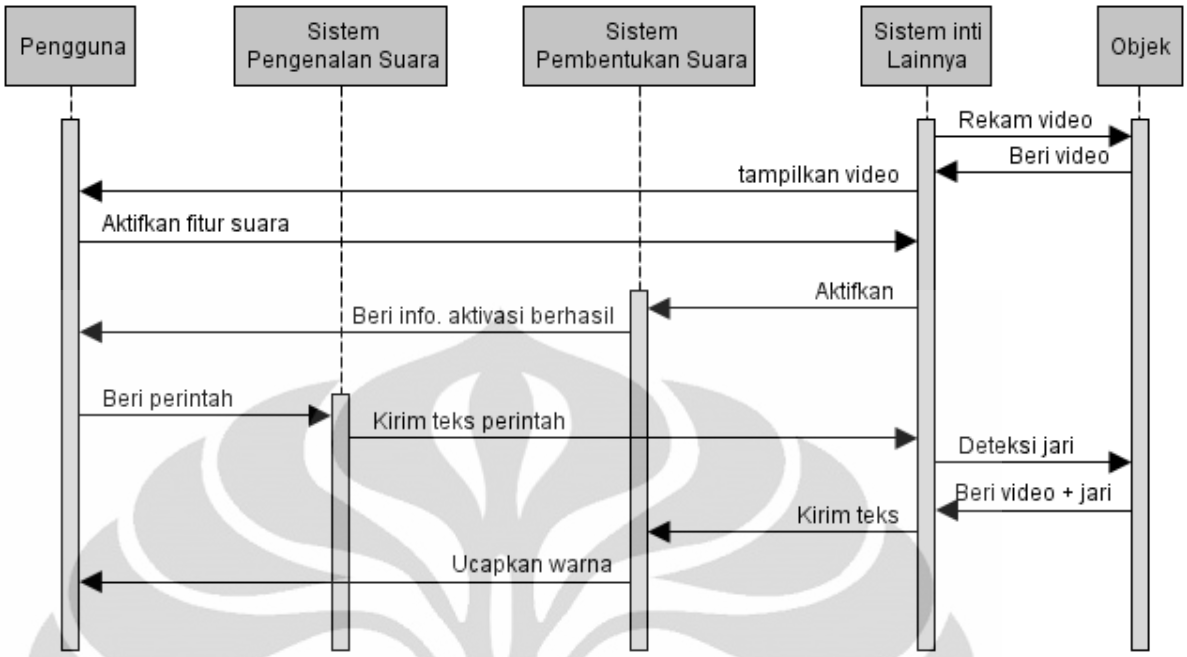
Gambar 3.9. *Activity Diagram* untuk Modul Sintesis Suara

Modul sintesis suara mulai bekerja ketika sistem menginstruksikan untuk menghasilkan suara dari data yang ada yang berupa teks. Selanjutnya, proses yang terjadi adalah:

1. Modul ini akan mengubah huruf demi huruf yang diterima menjadi fonem dan ditambahkan dengan durasi dan *pitch.*
2. Langkah selanjutnya, modul ini akan mensintesa rentetan fonem dengan durasi dan *pitch* masing-masing menjadi suara difon dan merangkainya menjadi suara per kata.
3. Suara tersebut kemudian disalurkan ke *speaker* yang selalu dalam keadaan *stand-by* selama sistem aktif.
4. Modul ini selesai melakukan tugasnya dan kembali ke keadaan *stand-by* dan menanti perintah selanjutnya.

c. *Sequence diagram*

Sequence diagram menggambarkan interaksi yang terjadi antar komponen dalam sistem, termasuk pengguna, dengan menyertakaan detil waktu keberadaan (*lifetime*) setiap komponen yang ada. Berikut ini *sequence diagram* dari sistem yang direncanakan untuk mewakili proses pada fitur *command voice* secara umum.



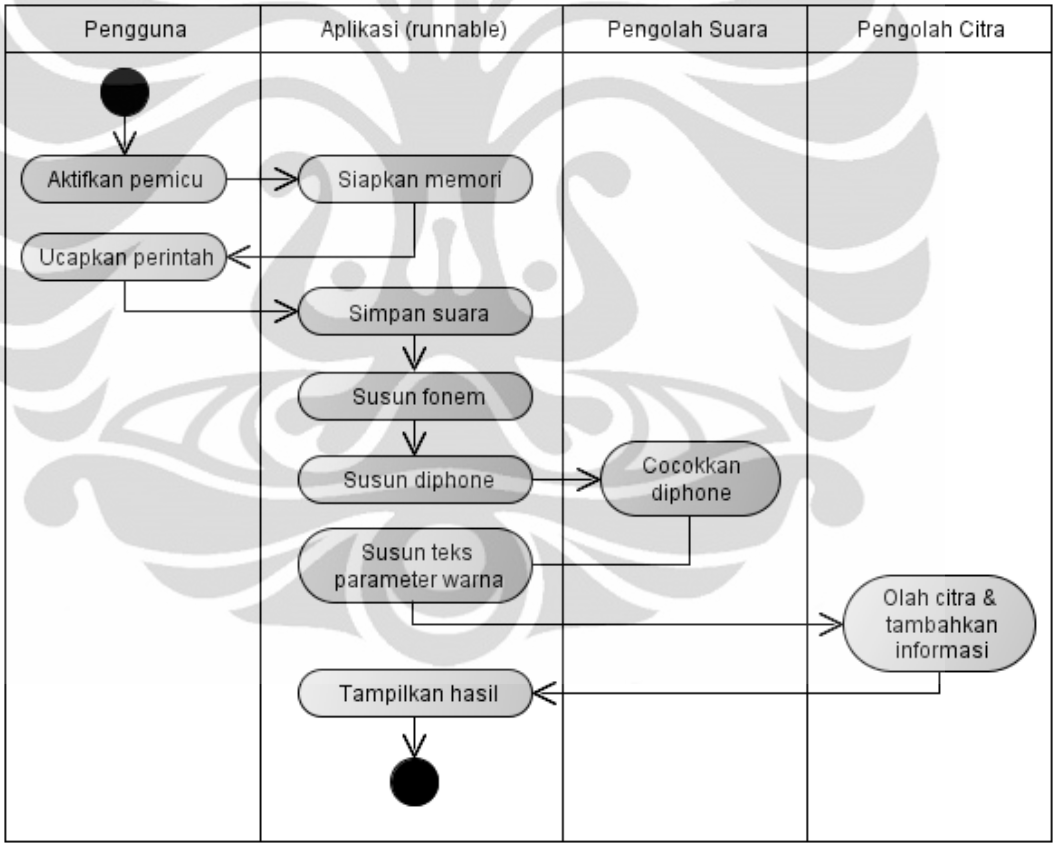
Gambar 3.10. *Sequence Diagram* untuk Fitur Suara

Gambar 3.10 mengilustrasikan interaksi yang terjadi secara urut dari atas ke bawah, sebagai berikut:

1. Pada permulaan, sistem utama akan merekam objek nyata dan dikembalikan menjadi tampilan video standar. Hal ini dilakukan dengan tambahan teks pada warna yang membingungkan untuk jenis buta warna yang dipilih di tahap sebelumnya.
2. Sistem utama akan memerintahkan fitur suara untuk hidup.
3. Saat pengguna memberikan perintah, sistem pengenalan suara memproses suara yang diterima dan mengirimkan hasil pemrosesannya dalam bentuk teks ke sistem utama.
4. Sistem utama kemudian merekam objek nyata dan mendeteksi jadi bila pengguna mengaktifkan fitur *point to sound*.
5. Hasil yang diberikan adalah video dengan tanda/*highlight* pada bagian yang ditunjuk dan teks warna. Video langsung ditampilkan di layar penampil, sedangkan teks dikirim ke sistem sintesis suara.
6. Sistem sintesis suara kemudian menterjemahkan teks tersebut menjadi suara, dan mengirimkan suara tersebut ke *speaker* untuk diucapkan.

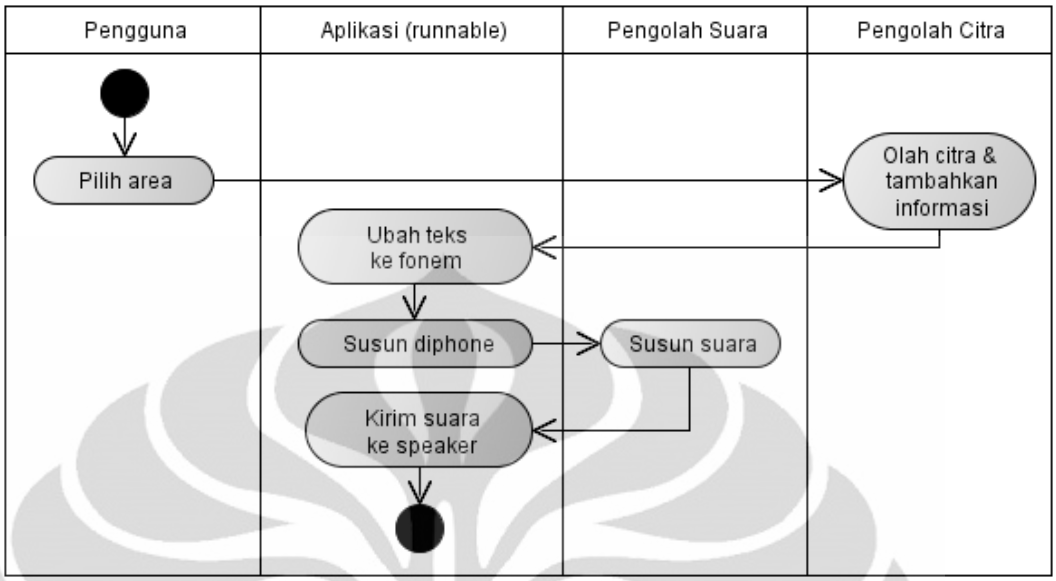
d. State Diagram

Gambar 3.11 menunjukkan *state diagram* untuk proses pengenalan suara. Pengguna harus menekan pemicu agar sistem menyiapkan *resource* yang dibutuhkan untuk pengenalan suara ini. Setelah sistem siap, barulah pengguna dapat mengucapkan perintahnya dan sistem akan mengolah suara yang masuk dan mencocokkannya dengan tabel *lookup*. Setelah itu, sistem akan menjalankan pekerjaan yang dispesifikasikan pada *record* tabel *lookup* yang dianggap cocok, dan memulai pengolahan citra. Pengolahan citra ini akan menghasilkan informasi tambahan yang kemudian ditampilkan bersama-sama dengan citra asli sebagai informasi *overlay*.



Gambar 3.11. *State Diagram* untuk Pengenalan Suara

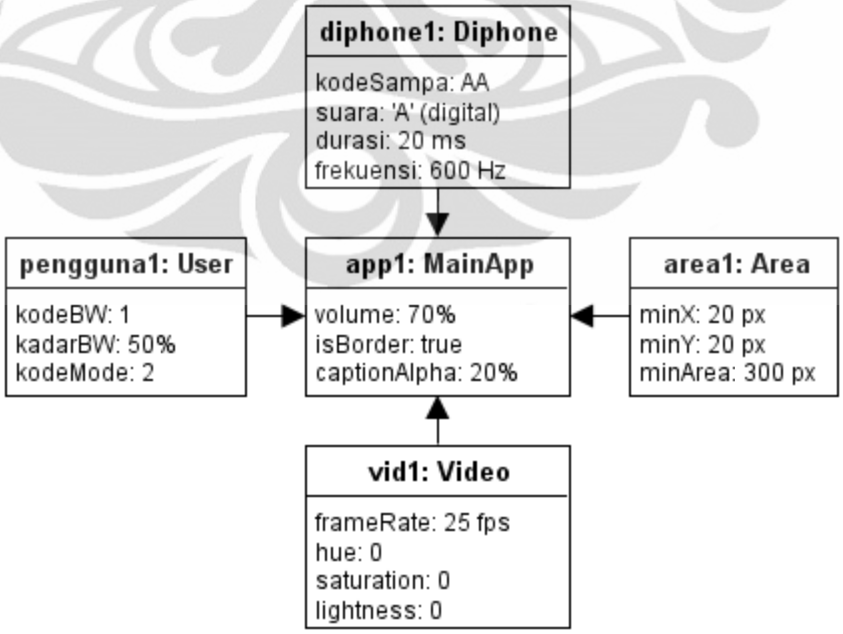
Gambar 3.12 menunjukkan *state diagram* untuk proses penyusunan suara. Setelah pengguna memilih area tertentu pada citra, modul pengolah citra akan menginformasikan warna apa yang dipilih pengguna, dan menerjemahkannya menjadi teks. Teks ini kemudian diproses oleh modul penyusun suara agar dapat dikirim ke *speaker* dan dikabarkan kepada pengguna.



Gambar 3.12. *State Diagram* untuk Penyusunan Suara

e. Object Diagram

Object diagram pada Gambar 3.13 mengilustrasikan objek apa saja yang terlibat dalam sistem serta rincian atribut yang dimiliki oleh setiap objek tersebut.



Gambar 3.13. *Object Diagram* Sistem

"pengguna1" merupakan objek dari kelas User yang memilih jenis buta warna pertama, yakni defisiensi hijau, dengan kadar 50%, dan sedang berada pada

model operasi 2, yang merupakan model untuk mengetahui nama warna dengan suara.

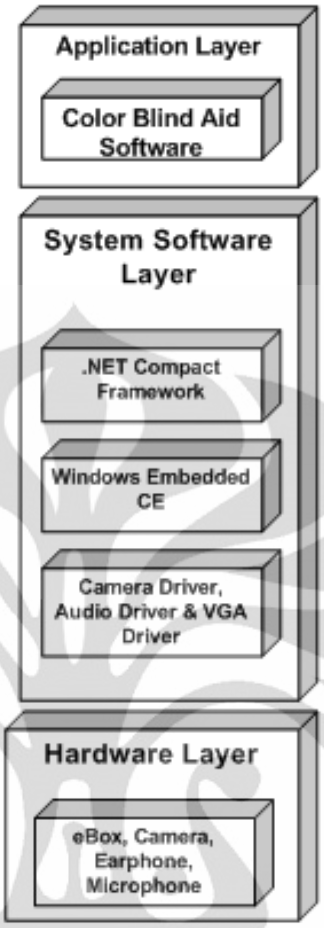
"app1" sebagai sistem yang dijalankan oleh "pengguna1" menentukan level suara sistem sebesar 70%, menampilkan *caption* dengan transparansi 20%, dan akan bekerja dengan menampilkan *border* pada area yang dipilih "pengguna1".

"difon1" merupakan instansiasi dari kelas Difon, yang akan digunakan untuk menghasilkan suara 'A' dengan durasi 20 milidetik dan frekuensi 600 Hz.

"vid1" merupakan objek dari kelas Video yang merupakan video yang akan ditampilkan kepada "pengguna1". Karena "pengguna1" memilih mode operasi 2, video yang ditampilkan tidak mengalami manipulasi. Nilai *hue*, *saturation*, dan *lightness* dari video tersebut merupakan nilai dasar, yakni nol.

f. Deployment Diagram

Deployment diagram menggambarkan konfigurasi statik dari komponen-komponen yang akan berjalan pada kondisi nyata natinya [31].

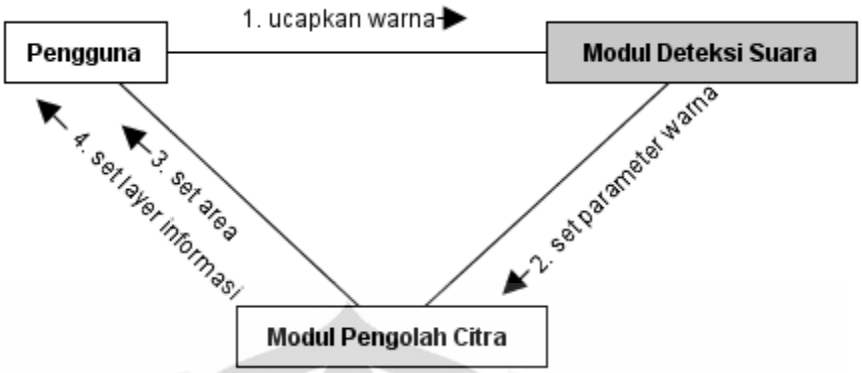


Gambar 3.14. *Deployment Diagram* Sistem

Gambar 3.14 [32] menunjukkan konfigurasi sistem yang dikembangkan. Sistem ini memiliki lapisan aplikasi yang merupakan bagian tempat berjalannya aplikasi. Lapisan selanjutnya adalah lapisan perangkat lunak sistem yang merupakan landasan bagi lapisan aplikasi karena menyediakan *library* .NET, WinCE, dan *driver* perangkat keras yang dibutuhkan. Lapisan terakhir adalah lapisan perangkat keras, yakni mencakup eBox, kamera, speaker/*earphone*, dan mikrofon.

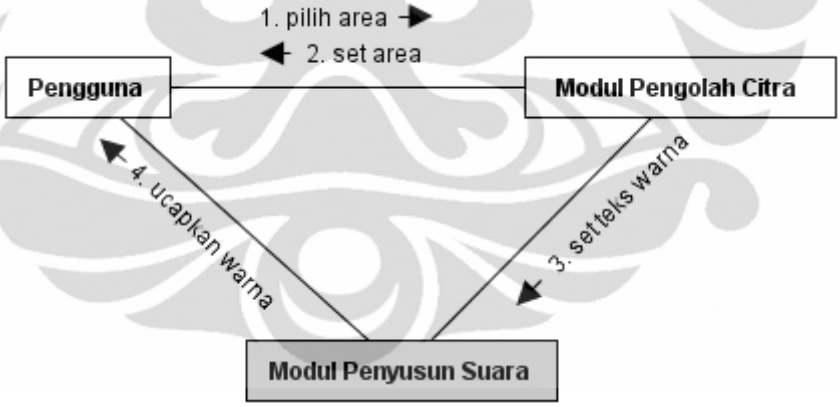
g. Collaboration Diagram

Collaboration diagram mengilustrasikan transaksi yang membentuk hubungan antarbagian dalam sistem.



Gambar 3.15. *Collaboration Diagram* untuk Pengenalan Suara

Gambar 3.15 menunjukkan *collaboration diagram* untuk proses pengenalan suara. Hal pertama yang terjadi adalah pengguna mengucapkan perintah berupa nama warna. Selanjutnya, perintah suara tersebut akan diterjemahkan oleh modul deteksi suara menjadi teks yang valid sebagai masukan proses pengolahan citra. Modul pengolah citra lalu menghasilkan parameter tertentu untuk menandai area dengan warna yang dipilih dan juga menambahkan *layer* informasi.



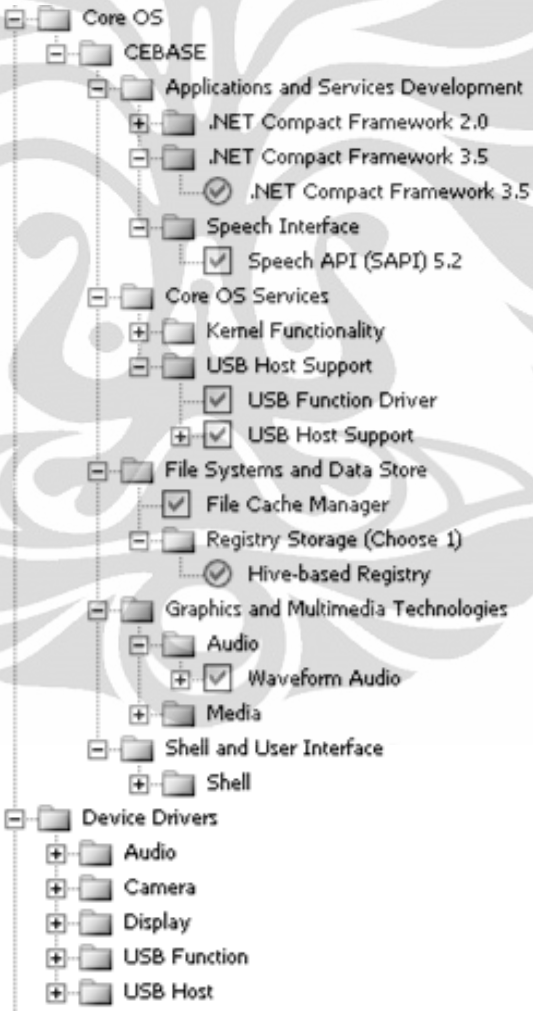
Gambar 3.16. *Collaboration Diagram* untuk Penyusunan Suara

Gambar 3.16 menunjukkan *collaboration diagram* untuk proses penyusunan suara. Setelah pengguna memilih area, modul pengolah citra akan menandai area yang dipilih dan mengirimkan teks warna dari area yang dipilih ke modul penyusun suara. Modul ini aka mengubah teks tersebut menjadi suara digital yang selanjutnya disampaikan kepada pengguna lewat *speaker*.

3.4. Perancangan Sistem Operasi

3.4.1. Sistem Operasi berbasis *Windows Compact Embedded* 6.0

OS yang ingin dirancang dengan WinCE6 adalah OS yang seringkas mungkin, namun tetap mendukung fungsionalitas aplikasi Chromophore. Hal yang pertama harus dilakukan adalah pemilihan *driver* perangkat keras agar OS dapat berkomunikasi dengan eBox. Kumpulan *driver* ini dapat diperoleh melalui BSP yang telah disediakan *Intelligent Control on Processor* (ICOP), vendor yang memproduksi eBox. Selain *driver* dari BSP, perlu juga tambahan *driver* untuk kamera.



Gambar 3.17. Komponen OS dan *Driver*

Selanjutnya, OS harus memiliki pustaka COM, ATL, dan .NET 3.5 agar set perintah yang dibutuhkan aplikasi Chromophore dapat berjalan dengan baik.

Selain komponen tersebut, yang juga ingin diimplementasikan pada OS ini adalah SAPI untuk mendukung fungsionalitas pemrosesan suara.

Setelah menyertakan komponen-komponen yang dibutuhkan oleh aplikasi, perlu juga dilakukan pemangkasan komponen-komponen lainnya yang tidak dibutuhkan, di antaranya perambah Internet, pemutar media, layanan-layanan jaringan, internasionalisasi, dan antarmuka *shell Explorer*.exe. Secara ringkas, komponen yang disertakan pada OS dapat dilihat pada Gambar 3.17.

Hal terakhir yang dilakukan adalah mengkonfigurasi OS agar segera menjalankan aplikasi Chromophore setelah *boot*, dan mencegah semua kontrol untuk mematikan aplikasi ini. Hal ini dapat dilakukan melalui pengaturan *registry*.

3.4.2. Sistem Operasi berbasis *Windows Embedded Standard* 2009

OS yang ingin dirancang dengan WES09 adalah OS yang secara arsitektur lebih mirip dengan OS *desktop*, namun diminimalisasi agar peluangnnya untuk dikembangan di perangkat tertanam dengan *resource* terbatas semakin besar. Dari segi pengembangan aplikasi, OS semacam ini akan memberikan ruang lebih luas untuk menerapkan komputasi yang lebih rumit.

Sama seperti sebelumnya, OS ini harus dirancang dengan memasukkan semua *driver* yang dibutuhkan dan pustaka pengembangan seperti COM dan .NET, dan SAPI untuk fungsionalitas pemrosesan suara. OS ini juga harus dioptimasi dengan menghilangkan komponen-komponen yang tidak dibutuhkan seperti fasilitas Internet, multimedia, jaringan, internasionalisasi, dan antarmuka *desktop*.

Optimasi lanjutan dilakukan dengan mengatur layanan apa saja yang harus berjalan, dibolehkan berjalan, atau tidak diperbolehkan sama sekali. Ini perlu dilakukan untuk menghemat sumberdaya.

BAB 4
IMPLEMENTASI SISTEM

4.1. Implementasi Fitur Suara dengan .NET *Framework*

Untuk implementasi fitur suara ini, yang pertama dilakukan adalah memulai *project* dengan *template* yang disebut *Windows Application* di *Visual Studio* 2010. Pada *project* tersebut, dibuatlah sebuah aplikasi *dummy* dengan *platform* pengembangan .NET *Framework* 4.0. Aplikasi ini disebut *dummy* karena belum mengintegrasikan semua fungsionalitas Chromophore, melainkan hanya menerapkan fitur pengenalan suara dan sintesis suara dari kelas yang telah dibuat sebelumnya.

Aplikasi ini dikembangkan dengan bahasa pemrograman Visual C#. Pada implementasi tahap ini, kemampuan/fitur aplikasi yang dirancang adalah:

- Mengenali perintah-perintah yang telah didefinisikan.
- Mengenali nama-nama warna tertentu yang telah didefinisikan.
- Mengucapkan nama-nama warna yang telah didefinisikan.

4.1.1. Sintesis Suara

Implementasi fitur sintesis suara, atau yang juga dikenal dengan istilah *text-to-speech*, dapat dilakukan dengan cukup mudah di lingkungan .NET *Framework*, karena tersedia *namespace System.Speech* yang siap digunakan di .NET *Framework* 3.5 dan 4.0.

Untuk dapat memanfaatkan *namespace* ini, yang perlu dilakukan adalah:

- menambahkan *System.Speech* ke dalam daftar referensi *project*,
- menggunakannya sebagai dependensi dengan *keyword* `using` di berkas/kelas yang akan memanggil fungsionalitas *namespace Speech*,
- membuat *instance* dari kelas *SpeechSynthesizer*. Melalui *instance* ini, pengembang dapat memanfaatkan *property* dan *method* yang telah disediakan untuk sintesis suara.

Berikut ini contoh program sederhana yang menggunakan kelas yang telah disediakan pada *namespace System.Speech.Synthesis* untuk mensintesis suara dari teks. Blok program di bawah ini akan menghasilkan suara ucapan "*test*".

```
//variabel penyimpan teks yang akan diucapkan
String kata = "test";
//membuat objek dari SpeechSynthesizer
SpeechSynthesizer SS = new SpeechSynthesizer();
//mengeluarkan teks dengan segera (asinkron)
SS.SpeakAsync(kata);
//menghapus objek
SS.Dispose();
```

Gambar 4.1. Penggunaan Kelas *SpeechSynthesizer* dalam Program

4.1.2. Pengenalan Suara

Seperti halnya sintesis suara, implementasi pengenalan suara juga cukup mudah dilakukan dengan memanfaatkan *namespace System.Speech*. Tiga hal yang harus dilakukan untuk menggunakan *namespace* ini adalah:

- menambahkan *System.Speech* ke dalam daftar referensi *project*,
- menggunakannya sebagai dependensi dengan keyword `using` di *berkas*/kelas yang akan memanggil fungsionalitas *namespace Speech*,
- membuat *instance* dari kelas *SpeechRecognizer*. Melalui *instance* ini, pengembang dapat memanfaatkan *property* dan *method* yang telah disediakan untuk pengenalan suara.

Berikut ini contoh sederhana implementasi kelas yang telah disediakan .NET setelah meng-*include System.Speech.Recognition*. Blok kode program di bawah ini mengenali kata-kata berdasarkan tabel *lookup* yang hanya mengandung kata-kata tertentu saja, yang akan digunakan di aplikasi Chromophore. Penggunaan tabel *lookup* ini membuat pengenalan lebih akurat, karena dengan demikian *SpeechRecognizer* hanya akan mencocokkan *stream* masukan dengan kata-kata yang ada di tabel ini, bukan dengan semua kata. Untuk memudahkan pengembangan, tabel *lookup* ini disimpan dalam berkas terpisah yang tidak diproses oleh *compiler*, dalam berkas grammar.dat.

```
//objek untuk menyimpan elemen-elemen dari grammar.dat
List<string> grammarList = new List<string>();
//membaca grammar.dat per baris
using (StreamReader readFile = new StreamReader("grammar.dat"))
{
    string line;
    while ((line = readFile.ReadLine()) != null)
    {
        // menyimpan tiap baris grammar.dat sebagai 1 elemen
        grammarList.Add(line);
    }
}
//membentuk array untuk transisi
Choices choices = new Choices(grammarList.ToArray());
// GrammarBuilder mengubah teks menjadi grammar
GrammarBuilder builder = new GrammarBuilder(choices);
//menyimpan grammar siap pakai
Grammar myGrammar = new Grammar(builder);

....

//membuat objek dari SpeechRecognitionEngine
SpeechRecognizer SR = new SpeechRecognizer();
//membaca tabel lookup grammar
SR.LoadGrammar(theGrammar);
//mulai mendengarkan audio dari mikrofon
SR.Enabled = true;
//berhenti mendengarkan audio dari mikrofon
//ini dilakukan setelah pengguna selesai mengucapkan kata
SR.Enabled = false;
//teks hasil pengenalan adalah SR.Result.toString()
```

Gambar 4.2. Penggunaan Kelas *SpeechRecognizer* pada Program

Pada pengembangan ini, kata-kata yang didefinisikan dalam berkas grammar.dat adalah 24 nama warna, yaitu *black, blue, brown, cream, dark blue, dark green, dark orange, dark red, dark yellow green, gold, gray, green, greenish gray, light blue, light green, magenta, maroon, orange, pink, purple, red, tosca, white, yellow*, bilangan 1 hingga 3 untuk navigasi *mode*, dan perintah "*select mode*" dan "*cancel*".

4.2. Implementasi Fitur Suara dengan .NET Compact *Framework*

Implementasi selanjutnya adalah pengembangan aplikasi dengan *platform* .NET *Compact Framework* 3.5. Implementasi tahap ini sebenarnya dilakukan karena aplikasi yang dikembangkan dengan plarform .NET *Framework*

ternyata tidak bisa dijalankan di atas OS WinCE yang hanya mendukung .NET *Compact Framework*.

Untuk target perangkat dengan OS WinCE6, pengembangan ini hanya dapat dilakukan dengan *Visual Studio* 2005 untuk .NETCF 2.0 dan *Visual Studio* 2008 untuk .NETCF 2.0 dan 3.5. Aplikasi harus dibuat berdasarkan *template* yang disebut *Smart Device Application*. Di sini, aplikasi yang dikembangkan tidak bisa merujuk pada pustaka .NET *Framework*, padahal *namespace System.Speech* yang dibutuhkan untuk pengenalan suara dan sintesis suara ada di sana.

Untuk mengatasi keterbatasan ini, dibuatlah komponen sintesis suara dan pengenalan suara yang dibutuhkan tanpa menggunakan *managed namespace System.Speech* dari Microsoft.

4.2.1. Sintesis Suara

Untuk membuat sistem sintesis suara sederhana, yang dibutuhkan adalah tabel asosiasi kata dan fonem-fonemnya, dan sampel suara untuk tiap fonem yang mengacu pada 41 fonem bahasa Inggris [33]. Terdapat 24 kata, yaitu jenis warna yang didefinisikan pada berkas teks sebagai tabel asosiasi, sebagai berikut:

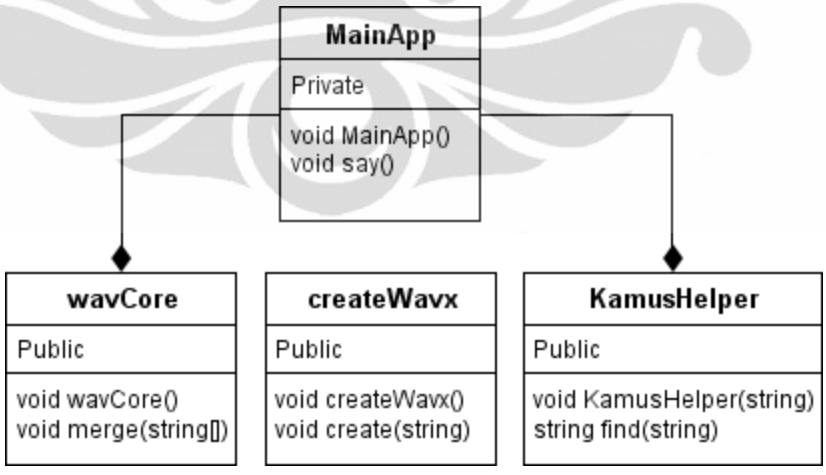
```
BLACK=B L AE K 0
BLUE=B L UW 0
BROWN=B R AW N 0
CREAM=K R IY M 0
DARK BLUE=D AA R K PAU B L UW 0
DARK GREEN=D AA R K PAU G R IY N 0
DARK ORANGE=D AA R K PAU AO R AH N JH 0
DARK RED=D AA R K PAU R EH D 0
DARK YELLOW GREEN=D AA R K PAU Y EH L OW PAU G R IY N 0
GOLD=G OW L D 0
GRAY=G R EY 0
GREEN=G R IY N 0
GREENISH GRAY=G R IY N IH SH PAU G R EY 0
LIGHT BLUE=L AY T PAU B L UW 0
LIGHT GREEN=L AY T PAU G R IY N 0
MAGENTA=M AH JH EH N T AH 0
MAROON=M ER UW N 0
ORANGE=AO R AH N JH 0
PINK=P IH NG K 0
PURPLE=P ER P AH L 0
RED=R EH D 0
TOSCA=T AO S K AH 0
WHITE=W AY T 0
YELLOW=Y EH L OW 0
```

Kode program dapat dibagi menjadi 3 kelas fungsional dan satu kelas aplikasi utama, seperti diilustrasikan pada Gambar 4.3. Kelas `createWavx` berfungsi untuk mengubah berkas-berkas audio WAV menjadi berkas yang lebih cepat diproses oleh kelas `wavCore`. Berkas ini berekstensi *.WAVX dan hanya menyimpan *waveform* saja. Kelas ini hanya digunakan sewaktu-waktu, untuk memperbarui sampel suara, dan tidak disertakan dalam aplikasi utama. Yang disertakan bersama aplikasi utama adalah berkas-berkas WAVX hasil buatan kelas ini dan disertakan dalam bentuk *embedded resource*.

Kelas `wavCore` berfungsi untuk mengambil *embedded resource* berformat WAVX untuk setiap fonem, merangkainya menjadi satu berkas WAV, dan menyerahkannya pada *System.Media.SoundPlayer* yang terdapat dalam pustaka .NETCF 3.5 untuk diputar.

Kelas `kamusHelper` berfungsi untuk mengambil tabel asosiasi fonem dari *embedded resource* dan menyimpan setiap elemen dari tabel tersebut ke dalam format *Collection* agar proses pencarian kata dapat dilakukan dengan cepat. *Method* `find` menerima masukan berupa kata dan mengembalikan keluaran berupa *array* fonem.

Kelas `mainApp` adalah aplikasi utama yang akan memanggil fungsionalitasi kelas-kelas yang lain.



Gambar 4.3. *Class Diagram* Implementasi Sistem Sintesis Suara Sederhana

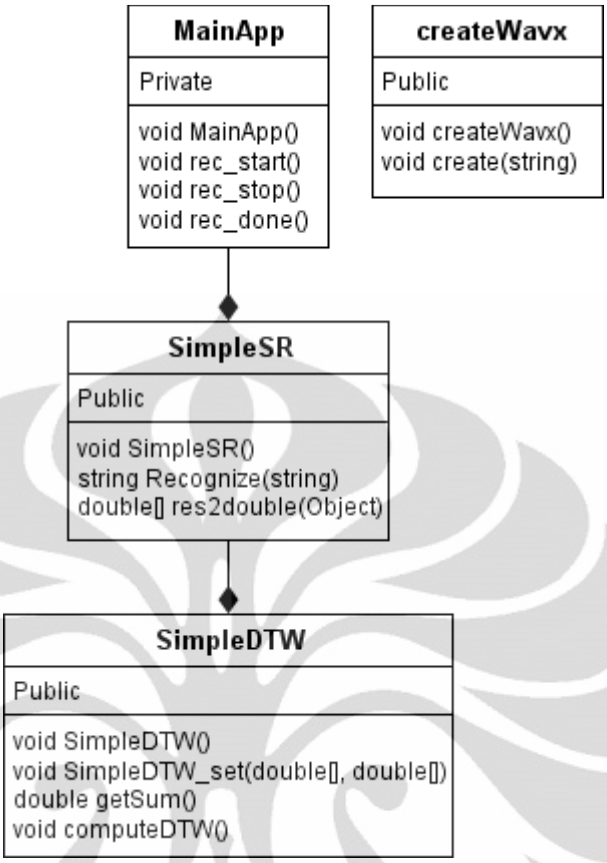
4.2.2. Pengenalan Suara

Algoritma DTW diimplementasikan untuk membuat sistem pengenalan suara. Algoritma ini dipilih mengingat Chromophore hanya membutuhkan pengenalan terhadap kata-kata yang terdefinisi saja, yang jumlahnya terbatas, dan tidak membutuhkan proses pembelajaran. Selain itu, ini dilakukan agar Chromophore memiliki komputasi yang sesederhana mungkin.

Mengacu pada konsep DTW yang telah dipaparkan pada subbab 2.3.5, untuk menerapkan algoritma ini, hal pertama yang harus ditentukan adalah kata-kata apa saja yang ingin dideteksi. Kumpulan kata ini disimpan dalam sebuah berkas teks. Untuk sistem ini, didefinisikan kata-kata yang sama seperti pada sistem pengenalan suara dengan .NET *Framework.*

Selanjutnya, dibutuhkan gelombang referensi (dalam hal ini berkas audio) untuk setiap kata yang terdefinisi, dan ini dapat diperoleh dari fasilitas audio Google *Translate*. Untuk mempercepat pengeloahan berkas ini, dilakukan pengubahan format menjadi format WAVX, dan semuanya disimpan sebagai *embedded resource*.

Untuk memungkinkan aplikasi merekam *stream* audio dari mikrofon, *namespace OpenNETCF.Media.WaveAudio* dapat digunakan. Ini merupakan pustaka yang disediakan komunitas OpenNetCF untuk melengkapi Microsoft .NETCF.



Gambar 4.4. *Class Diagram* Sistem Pengenalan Suara dengan DTW

Kelas `MainApp` mengatur perekaman suara dengan *method* `rec_start` dan `rec_stop`, dan menginisiasi kelas `SimpleSR` serta memulai pengenalan di kelas `SimpleSR` dengan *method* `rec_done` yang dieksekusi bila *stream* WAV hasil perekaman telah siap. *Method* `Recognize` milik `SimpleSR` membaca *waveform* dari *stream* hasil perekaman dan membandingkannya dengan setiap gelombang referensi dengan menginisiasi `SimpleDTW`. *Method* `getSum` dari `SimpleDTW` mengembalikan hasil perhitungan ke *method* `Recognize` milik `SimpleSR`. Pada langkah selanjutnya di *method* ini, kata yang memiliki hasil DTW terkecil dipilih sebagai hasil pengenalan.

4.3. Implementasi Sistem Operasi Windows CE 6.0

Pembuatan OS berbasis WinCE6 untuk sistem Chromophore merupakan percobaan yang pertama dilakukan guna memperoleh OS standar dan memenuhi kebutuhan aplikasi. Pembuatan OS ini dilakukan dengan lingkungan pengembangan (IDE) berupa *Visual Studio* 2005.

Untuk memulai pengembangan OS berbasis WinCE*6*, dibutuhkan beberapa tahap persiapan, khususnya yang menyangkut penyesuaian OS. Persiapan tahap pertama adalah menginstal perangkat pengembangan yang dibutuhkan, yaitu [34]:

1. *Visual Studio* 2005 sebagai IDE utama
2. WinCE6 yang akan menginstal komponen-komponen OS WinCE6 untuk pengembangan dan juga Platform Builder. *Platform Builder* 6.0 IDE pada dasarnya adalah sebuah *plug-in* berupa *project wizard* bagi *Visual Studio* 2005. *Platform Builder* 6.0 tidak bisa berjalan pada *Visual Studio* 2008.
3. *Microsoft Visual Studio* 2005 *Team Suite Service Pack 1*
4. WinCE6 *Platform Builder Service Pack 1*
5. WinCE6 R2
6. WinCE6 USB *Camera Driver* untuk tambahan *driver* kamera yang berbasis USB Video Class versi 1.1.

Selanjutnya, lewat *Visual Studio,* dibuatlah *project* dengan *template* yang bernama *Platform Builder for CE 6.0*, dan kemudian memilih *Board Support Package* (BSP). Untuk memperoleh hasil yang terbaik, dilakukan percobaan satu-persatu BSP yang telah ditambahkan pada *platfotm builder*, mereka adalah:

- **CEPC: x86** → BSP standar yang disediakan oleh Microsoft yang menyediakan *driver* untuk perangkat keras yang umum digunakan.
- **ICOP_eBox3310A_60GS: X86** → BSP yang disediakan oleh ICOP untuk *driver* WinCE untuk *device* eBox-3300/3310A.
- **ICOP_eBox3310A_60H: X86** → BSP yang disediakan oleh ICOP untuk *driver* WinCE untuk *device* eBox-3300A/3310A.

Meskipun pada tahap awal ini dapat dipilih lebih dari satu BSP sekaligus, pada akhirnya hanya boleh ada 1 BSP yang dipilih. Oleh karena itu, kombinasi BSP tidak mungkin dilakukan.

Selanjutnya, setelah sumber untuk *driver* perangkat keras yang akan disertakan dalam perancangan OS ditentukan melalui BSP, penentuan komponen/*catalog item* yang akan disertakan dilakukan secara manual. Perlu diketahui bahwa tidak ada dependensi dalam pemilihan komponen OS dengan pemilihan BSP yang telah dilakukan sebelumnya.

Pada tahap awal ini, *Platform Builder* mengharuskan untuk memilih satu jenis *template*, yang merepresentasikan set komponen yang dipilih untuk dimasukkan ke dalam OS yang akan dibuat. *Template* yang tersedia adalah:

- *Consumer Media Device* → *template* yang menyertakan *catalog item* dasar untuk memungkinkan pemutaran audio dan video yang biasanya dibutuhkan untuk keperluan perangkat hiburan.
- *Custom Device* → *template* yang tidak menyertakan *catalog item* apa-apa yang dapat dipilih untuk pengembangan perangkat sangat sederhana hingga perangkat yang kaya fitur.
- *Industrial Device* → *template* yang menyertakan *catalog item* dasar untuk pengembangan otomasi perangkat industri seperti panel *Human-machine Interface* (HMI) atau *Programmable Logic Controller* (PLC).
- *PDA Device* → *template* yang menyertakan *catalog item* dasar untuk berbagai jenis PDA atau perangkat selular dengan *keyboard*.
- *Phone Device* → *template* yang menyertakan *catalog item* dasar untuk pengembangan telepon berbasis Internet menggunakan VoIP.
- *Small Footprint Device* → *template* yang menyertakan *catalog item* dasar untuk pengembangan perangkat dengan fungsi minim.
- *Thin Client* → *template* yang menyertakan *catalog item* dasar untuk pengembangan perangkat akses *remote* dengan shell yang terbatas.

Untuk Chromophore, pengembangan OS dilakukan dengan *template Custom Device*, karena *template* lainnya akan jauh dari fungsi OS yang diharapkan untuk perangkat tertanam Chromophore.

4.3.1. Penyesuaian Komponen Katalog

Katalog memuat berbagai komponen yang dapat ditambahkan sebagai penyusun OS WinCE. Komponen-komponen pada katalog dikategorikan menjadi 14 kategori.

Selanjutnya, dilakukan pememilihan komponen-komponen yang dibutuhkan untuk fungsionalitas Chromophore. Beberapa komponen memiliki dependensi yang membuat *Visual Studio* mengikutsertakannya secara otomatis. Dari berbagai

kategori, berikut ini komponen yang dipilih manual, bukan yang ditambahkan otomatis:

- .NET Compact *Framework* 2.0 → berisi kelas-kelas siap pakai (*managed namespace*) dari .NETCF 2.0 [35].
- .NET Compact *Framework* 3.5 → berisi kelas-kelas siap pakai (*managed namespace*) dari .NETCF 3.5 [35].
- Active Template Library (ATL) → berisi dukungan untuk pengembangan *Component Object Model* (COM) [36] seperti *Object Linking and Embedding* (OLE), dan *ActiveX*.
- *Speech* API (SAPI) 5.2 → berisi dukungan untuk pengenalan suara dan sintesis suara [35].
- USB *function driver* → *driver* untuk menangani fungsionalitas spesifik USB. Ini dibutuhkan untuk menghubungkan eBox dengan kamera USB.
- *File Cache Manager* → menyediakan fungsionalitas *cache* untuk mempercepat akses *berkas* dan menghemat daya, karena mengakses *disk* membutuhkan lebih banyak daya daripada mengakses *cache* [37].
- *Hive-based registry* → memungkinkan penyimpanan konfigurasi *registry* dalam *file system*. *Hive-based registry* dipilih daripada *RAM-based registry* karena *RAM-based registry* kurang efisien untuk perangkat yang melakukan *cold reboot* yang disebut juga *hard reset*, dan memiliki *persistent storage* [38].
- *DirectDraw* → dibutuhkan untuk komunikasi dengan *driver* kamera yang disediakan Microsoft yang berbasis *DirectShow*.
- Wave/AIFF/AU/SND *File Parser* → memungkinkan pembacaan berkas suara, khususnya format WAV yang akan diigunakan dalam sintesis suara [39].
- *Waveform Audio Renderer* → memungkinkan sintesis dan *playback* suara berdasarkan *stream waveform*.
- Semua komponen dalam kategori *DirectShow*.
- Semua komponen dalam kategori *Video Codecs and Renderers.*

4.3.2. Penyesuaian *Driver*

Dalam implementasi ini, perangkat keras yang digunakan selain eBox adalah monitor standar, *keyboard* standar, *headphone* standar, dan sebuah USB *webcam* merek Prolink PCC5020 yang kompatibel dengan USB Video *Class* versi 1.1. Untuk hal itu, *driver* yang disertakan dalam OS, bukan yang disertakan otomatis oleh *Visual Studio*, meliputi:

- Ensoniq ES1371 pada kategori *Audio*.
- NULL *Camera Driver.*
- NULL (*Stub*) pada kategori *Display.*
- MJPEG *Decompression Filter* (pada kategori *Third Party*) [40].
- USB *Camera Driver* (pada kategori *Third Party*).

4.3.3. Penyesuaian Tambahan

Penyesuaian tambahan yang dilakukan adalah membuat OS segera menjalankan aplikasi Chromophore setelah *boot*, dengan kata lain, membuat aplikasi Chromophore menjadi *autorun at startup*. Untuk melakukan hal ini, ditambahkanlah REG_SZ key bernama *Launch*99 dengan nilai berupa *path* tempat aplikasi Chromophore berada. *Key* ini harus ditempatkan di *registry* HKEY_LOCAL_MACHINE/init. Penambahan ini dilakukan lewat berkas *PUBLIC*/*common*/*Parameter Files*/*common*.reg di *solution explorer* di layar kerja *Visual Studio*.

4.3.4. Pemasangan Sistem Operasi

Proses kompilasi OS memakan waktu sekitar 30 menit bila dilakukan di atas komputer pengembangan dengan prosesor ganda @ 1.86 GHz, 2 GB RAM, dan hard disk dengan kecepatan putar 5400 RPM. Setelah proses kompilasi OS selesai, *image* dari OS yang dibuat akan tersimpan di direktori hasil *debug* atau *release*, tergantung setelan *project,* sebagai sebuah *berkas* bernama NK.BIN.

Ada beberapa cara yang dapat dilakukan untuk menjalankan *image* ini di eBox. Cara yang paling umum digunakan adalah dengan koneksi kabel *twisted-pair* yang biasa digunakan untuk menghubungkan koneksi LAN *Ethernet*. Dalam hal ini, kedua jenis kabel, *cross* dan *straight* dapat digunakan. Koneksi dengan

cara ini membutuhkan peran *Connection Manager* dari *Visual Studio*, dan dapat digunakan untuk *remote debugging* OS dan aplikasi yang dijalankan secara waktu aktual di eBox.

Cara kedua adalah dengan memindahkan *image* OS secara manual dengan *removable* media, menyetel BIOS eBox agar mendeteksi *removable* media tersebut, dan mengganti NK.BIN di *local disk* eBox dengan NK.BIN baru dari *removable* media dengan *command line* eBox, yang dapat dijalankan dengan menekan F5 saat memilih sumber OS yang ingin dijalankan. Saat berjalan dalam DOS, aplikasi Chromophore dan berkas-berkas pendukungnya juga disalin ke lokasi yang telah dipilih untuk *autorun*. Setelah semuanya siap, baru dilakukan *boot* ulang eBox dengan cara biasa, yaitu *boot* dari *local disk*. Cara ini tidak bisa digunakan untuk *debug*, namun lebih cepat untuk diakukan berulang kali dibandingkan cara pertama.

WinCE *root* membutuhkan sistem *berkas* FAT (Win FAT 16 *Logical Block Addressing*/LBA) dan *berkas-berkas* berikut ini [24] sebagai *Master Boot Record* (MBR):

- Autoexec.bat → program sederhana berupa *boot selector* yang akan dijalankan oleh DOS pertama kali.
- Config.sys → menyimpan variable yang dibutuhkan saat OS berjalan.
- Eboot.bin → komponen yang dibutuhkan untuk menjalin koneksi Ethernet.
- Loadcepc.exe → komponen utama yang dibutuhkan untuk memproses NK.bin, baik di *local storage* maupun di komputer *remote* lewat koneksi Ethernet.
- NK.bin → image OS WinCE.

4.4. Implementasi Sistem Operasi *Windows Embedded Standard* 2009

Pembuatan OS berbasis WES09 ini sebenarnya dilakukan karena WinCE6 yang telah dicoba sebelumnya memiliki keterbatasan dalam hal *Framework* (lihat bagian analisis: BAB 55.4.1). WES09 mendukung *platform* .NET, sementara WinCE6 hanya mendukung .NET *Compact Framework* (CF) yang ternyata berbeda dengan .NET biasa, sehingga banyak *managed namespace* dari .NET yang tidak tersedia. Selain itu, WinCE juga tidak menyediakan *speech engine* [41],

meskipun komponen SAPI 5.2 telah dipilih pada OS desainnya. Dibutuhkan komponen pihak ketiga untuk membuat fitur suara di atas .NETCF [33], dan *speech engine* pihak ketiga yang telah mendukung *porting* ke WinCE6 tidak ada yang gratis.

Pengembangan OS berbasis WES09 harus dilakukan dengan IDE *Windows Embedded Studio*. Tahap pertama yang dilakukan untuk memulai pengembangan OS WES09 adalah dengan membuat *project* dan memilih arsitektur CPU yang sesuai, dalam hal ini, digunakan CPU berarsitektur x86.

4.4.1. Penyesuaian *Driver*

Untuk menggantikan peran BSP pada Platform Builder WinCE6, *Windows Embedded Studio* memiliki aplikasi atomik pendukung yang disebut *Target Analyzer*. *Target Analyzer* adalah aplikasi yang harus dieksekusi di perangkat target (*target device,* dalam hal ini eBox 3300) untuk mengumpulkan informasi apa saja *driver* perangkat keras yang dibutuhkan perangkat target. Setelah dijalankan, *Target Analyzer* menyimpan daftar *driver* tersebut dalam sebuah berkas berekstensi *.PMQ yang selanjutnya akan digunakan untuk menginstruksikan *Target Designer* pada *Windows Embedded Studio* untuk mengikutsertakan *driver-driver* tersebut.

Terdapat dua jenis Target Analyzer, TA.EXE dan TAP.EXE (*Target Analyzer Pro*). TA dapat berjalan meskipun dengan OS DOS, namun TA kadang tidak dapat memberikan daftar *driver* yang lengkap. TAP hanya dapat berjalan dengan OS Windows NT (Windows 2000 dan Windows XP), namun TAP dapat memberikan daftar *driver* yang lengkap, karena TAP juga mengumpulkan informasi dari OS di mana ia dijalankan. Karena WES09 mirip dengan Windows XP, untuk meminimalisasi resiko kegagalan, digunakanlah TAP di atas Windows XP yang telah dimodifikasi sehingga dapat dimuat dan dijalankan dari USB *storage* (*live USB*).

Selain menggunakan berkas PMQ dari TAP, terdapat *driver* yang harus disertakan untuk eBox 3300, namun tidak terdapat dalam katalog komponen

WES09. *Driver* ini harus ditambahkan dengan cara mengunduhnya dari ICOP, dan memasukkannya ke dalam katalog dengan *Component Database Manager.*

4.4.2. Penyesuaian Komponen Katalog

Seperti halnya *Platform Builder* untuk WinCE6, perancangan OS berbasis WES09 di *Target Designer* juga dilakukan dengan memilih komponen-komponen dari katalog. Berikut ini komponen yang ditambahkan dari katalog WES09 ke dalam OS yang dirancang khusus untuk Chromophore:

- .NET *Framework* 3.5
- *Speech* API
- *Speech Core*
- *Task Manager* untuk keperluan observasi dan optimasi OS yang dilakukan di atas perangkat target.
- *USB Boot 2.0,* dibutuhkan untuk *boot* dari USB *storage*.
- *Runtime Quick Start Helper*, dibutuhkan untuk *boot* dari USB *storage*.

Selain penambahan komponen, juga dilakukan pengurangan komponen yang tidak dibutuhkan, seperti aplikasi klien pembaca *e-mail*, pemutar media, perambah Internet, dan konektifitas jaringan.

4.4.3. Pemasangan Sistem Operasi

Setelah pemilihan *driver* dan komponen selesai dilakukan, sebelum kompilasi, *Target Designer* otomatis menjalankan pemeriksaan dependensi komponen, dan menambahkan komponen yang menjadi dependensi komponen dan *driver* yang telah dipilih sebelumnya. Proses pemeriksaan dependensi dan kompilasi memakan waktu sekitar 40 menit. *Runtime image* yang dihasilkan berukuran sekitar 300 MB, terdiri dari:

- NTDETECT.COM
- ntldr
- *boot*.ini

- WERUNTIME.INI
- Direktori *Documents and Settings*
- Direktori *Program Files*
- Direktori WINDOWS

Pemasangan OS ke dalam perangkat target, eBox 3300, dilakukan dengan menyiapkan *storage* yang akan dijadikan sumber *boot* oleh eBox, dan memindahkan *runtime image* OS yang telah dirancang ke *storage* tersebut. Persiapan *storage* sebenarnya dilakukan untuk menyiapkan partisi *boot* yang sesuai dengan *boot file* yang secara *default* dihasilkan saat pembuatan *runtime image*. Partisi ini harus dibuat dengan tepat agar BIOS dapat memperlakukannya sebagai *Master Boot Record* (MBR).

Awalnya berkas-berkas OS ditempatkan di *storage* lokal eBox. Namun karena ukurannya yang hanya 488 MB, *storage* ini tidak mencukupi, karena *First Boot Agent* (FBA). FBA berjalan otomatis saat WES09 *boot* pertama kali di perangkat target, untuk penyesuaian dan registrasi *driver* serta komponen OS membuat ukuran OS mendekati 700 MB (lihat bagian BAB 55.4.2).

Akhirnya diputuskan untuk memindahkan *storage* ke USB *flash disk* berukuran 4 GB. Karena penggantian storage tersebut, diperluka dua lagi komponen tambahan, yaitu *USB Boot 2.0* dan *Runtime Quick Start Helper*, sehingga mengharuskan kompilasi ulang, penggunaan aplikasi *UFDPrep* yang disediakan *Windows Embedded Studio* untuk membuat MBR di *flash disk*, penyalinan semua berkas runtime image ke *flash disk* tersebut, dan pengaturan *boot sequence* di BIOS eBox dan menjadikan *flash disk* yang digunakan sebagai satu-satunya *drive* yang dapat di*boot*.

4.4.4. Modifikasi Tambahan

Modifikasi tambahan dilakukan saat OS sudah berjalan di atas perangkat target, dengan bantuan *System Configuration Utility* (msconfig.exe), *Services*.msc, *Registry Editor* (regedit.exe) dan *Task Manager* (taskmgr.exe). Modifikasi yang dilakukan meliputi:

- Menyalin aplikasi Chromophore dan menambahkannya ke daftar aplikasi *Startup*.
- Menonaktifkan semua aplikasi *Startup* kecuali Chromophore.
- Melumpuhkan (membuatnya menjadi *disable*) *service-service* yang tidak dibutuhkan, dan mengatur *service-service* yang dibutuhkan agar dijalankan secara otomatis.
- Meniadakan aplikasi *shell*.

BAB 5
UJI COBA DAN ANALISIS

Terdapat 2 sistem sintesis suara dan 2 sistem pengenalan suara yang dibuat. Keterbatasan perangkat keras pada eBox membuat pengujian terhadap 4 sistem tersebut dilakukan di atas komputer pengembangan untuk mempercepat proses pengujian. Alternatif ini dipilih karena keempat sistem tersebut bekerja berdasarkan sampel suara yang akan tetap sama di manapun mereka dijalankan. Pengujian terhadap kedua sistem operasi tetap dilakukan di atas eBox.

5.1. Analisa Implementasi Sintesis Suara

5.1.1. Responden

Untuk pengujian modul ini, diambil 10 responden yang merupakan mahasiswa Departemen Teknik Elektro FTUI dengan kisaran umur responden antara 20 hingga 25 tahun, asal daerah/logat yang berbeda-beda, dan kemampuan bahasa Inggris yang berbeda-beda.

5.1.2. Metode Pengujian

Setiap responden akan terlebih dahulu mendengar kata-kata yang nantinya akan diucapkan sistem, dengan *audio* yang bersumber dari Google *Translate.* Hal ini dilakukan untuk meminimalisasi variabel bebas, yakni kemampuan berbahasa Inggris dari responden yang berbeda-beda.

Setiap responden akan mencoba mengenali kata-kata yang diucapkan oleh masing-masing sistem, tiap kata akan diucapkan maksimal hingga 3 kali, dan dilakukan pencatatan pada pengulangan ke berapa responden berhasil mengenali kata yang bersangkutan dengan benar. Sistem A adalah sistem yang menggunakan *namespace System.Speech*, sedangkan Sistem B adalah sistem yang dibuat tanpa menggunakan *namespace System.Speech*. Kata-kata yang diujikan adalah sebagian (dipilih secara acak) dari kata yang akan digunakan pada sistem Chromophore.

Selain itu, untuk melihat kepuasan responden terhadap implementasi fitur sintesis suara dengan .NET dan SAPI pada Chromophore, responden diminta

memberikan komentar terhadap pernyataan "suara yang diucapkan jelas bagi saya" setelah beberapa waktu mencoba menggunakan Chromophore.

Setiap responden menjalani pengujian dengan menggunakan *headphone*, sedangkan panduan dan umpan balik diberikan secara lisan. Berikut ini spesifikasi *headphone* yang digunakan:

Merek	: SPCPhone Bravo Series
Mikrofon	: 9X7/58dB+2dB
Jangkauan frekuensi	: 30-13500 Hz
Sensitivitas speaker	: 105 dB
Impedansi speaker	: 32 ohm
Panjang *chord*	: ± 2,5 m
Daya masukan maksimal	: 100 mW

5.1.3. Hasil Pengujian dan Analisis

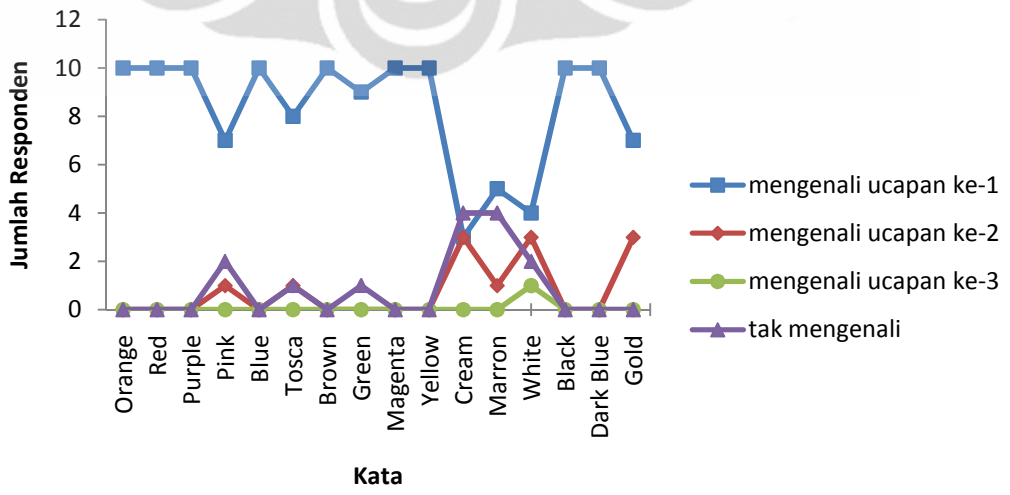
Tabel 5.1. Hasil Pengujian Sistem Sintesis Suara

No	Kata	Jumlah Responden							
		Sistem A				Sistem B			
		Ke-1	Ke-2	Ke-3	Gagal	Ke-1	Ke-2	Ke-3	Gagal
1	*Black*	10	0	0	0	9	0	1	0
2	*Blue*	10	0	0	0	3	1	0	6
3	*Brown*	10	0	0	0	5	2	3	0
4	*Cream*	3	3	0	4	3	0	0	7
5	*Dark Blue*	10	0	0	0	7	1	1	1
6	*Gold*	7	3	0	0	3	2	1	4
7	*Green*	9	0	0	1	9	0	1	0
8	*Magenta*	10	0	0	0	8	1	1	0
9	*Marron*	5	1	0	4	6	0	0	4
10	*Orange*	10	0	0	0	9	1	0	0
11	*Pink*	7	1	0	2	1	0	0	9
12	*Purple*	10	0	0	0	0	2	0	8
13	*Red*	10	0	0	0	7	2	1	0
14	*Tosca*	8	1	0	1	7	1	0	2
15	*White*	4	3	1	2	2	0	1	7
16	*Yellow*	10	0	0	0	9	1	0	0

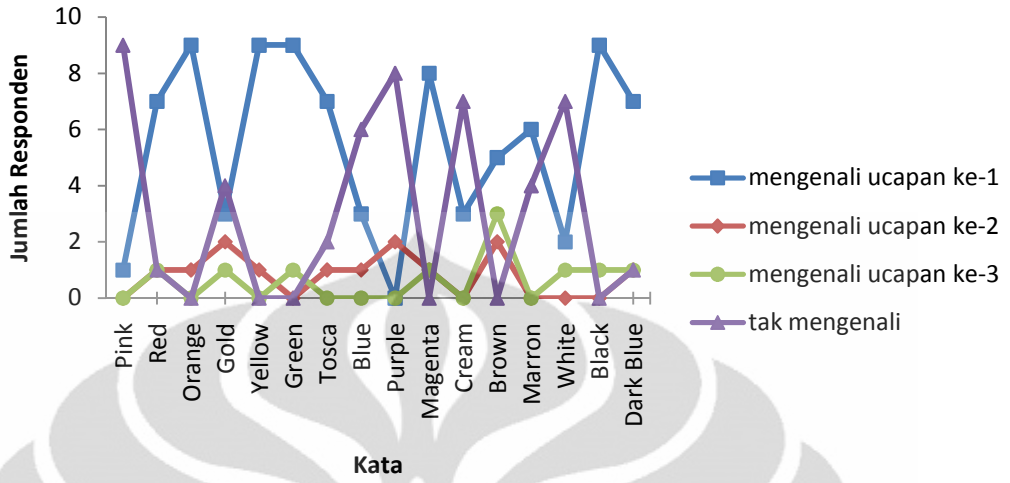
Tabel 5.1 menyajikan data hasil uji coba, yakni jumlah orang yang dapat dengan benar mengenali kata yang diucapkan sistem. Pada tabel tersebut, pengujian terhadap kedua sistem direpresentasikan dengan kolom **Sistem A** untuk

sistem yang menggunakan *namespace System.Speech*, dan kolom **Sistem B** untuk sistem yang tidak menggunakan *namespace System.Speech*. Setiap kolom ini kemudian dibagi menjadi 4 bagian. Kolom **ke-1** untuk jumlah responden yang berhasil mengenali kata yang bersangkutan pada pengucapan pertama. Kolom **ke-2** untuk jumlah responden yang berhasil mengenali kata yang bersangkutan pada pengucapan kedua. Kolom **ke-3** untuk jumlah responden yang berhasil mengenali kata yang bersangkutan pada pengucapan ketiga. Kolom **Gagal** untuk jumlah responden yang tidak berhasil mengenali kata yang bersangkutan. Misalkan, angka **9** di bawah kolom **ke-1** di **Sistem A** pada baris kata **Green** menunjukkan bahwa ada 9 orang responden yang dapat dengan tepat mengenali kata "*green*" yang diucapkan oleh sistem sintesis suara yang dibuat dengan memanfaatkan *namespace System.Speech* pada pengucapan pertama.

Gambar 5.1 menunjukkan distribusi 10 responden berdasarkan pengenalannya terhadap setiap kata yang diucapkan sistem A. Gambar 5.2 mengelompokkan responden berdasarkan pengenalannya terhadap setiap kata yang diucapkan sistem B. Pada kedua grafik ini, sumbu X menunjukkan kata-kata yang diucapkan sistem, yang disusun urut berdasarkan urutan pengujiannya. Sumbu Y menunjukkan jumlah responden di setiap kejadian untuk setiap kata. Empat plot garis menunjukkan 4 kejadian yang menggambarkan pengenalan responden terhadap kata yang bersangkutan, yakni mengenali ucapan ke-1, mengenali ucapan ke-2, mengenali ucapan ke-3, dan tidak mengenali.



Gambar 5.1. Distribusi Responden berdasarkan Pengenalannya terhadap Setiap Kata di Sistem A



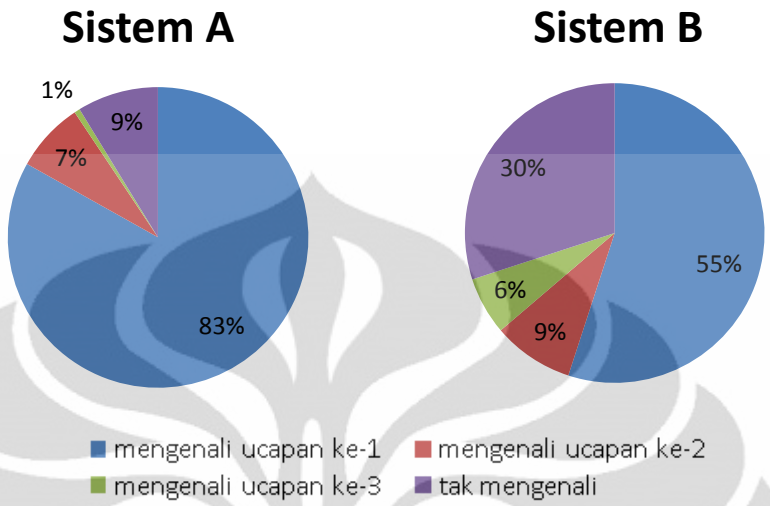
Gambar 5.2. Distribusi Responden berdasarkan Pengenalannya terhadap Setiap Kata di Sistem B

Dari Gambar 5.1, terlihat bahwa sebagian besar responden dapat mengenali semua kata saat baru diucapkan 1 kali. Hanya berapa responden yang membutuhkan pengucapan lebih dari 1 kali, dan itu pun hanya untuk beberapa kata. Meskipun demikian, ada 1 kata yang memiliki tingkat pengenalan relatif rendah, yakni kata "*cream*", yang mungkin karena kemiripannya dengan kata "green" yang mungkin lebih familiar bagi responden.

Dari Gambar 5.2, terlihat bahwa kejadian yang sering muncul adalah responden berhasil mengenali kata di ucapan ke-1, dan tidak berhasil mengenali sama sekali. Kata pertama, "*pink*", hanya dikenali 1 orang responden. Karena sistem B diuji setelah sistem A yang secara umum menghasilkan suara yang lebih mudah dikenali, ini terjadi karena responden belum terbiasa dengan sistem B, sebagaimana salah satu responden ada yang mengatakan bahwa dirinya sudah mulai terbiasa di tengah pengujian terhadap sistem B, dan satu lagi responden yang mengatakan bahwa dirinya kurang cepat menyesuaikan diri di akhir pengujian sistem B.

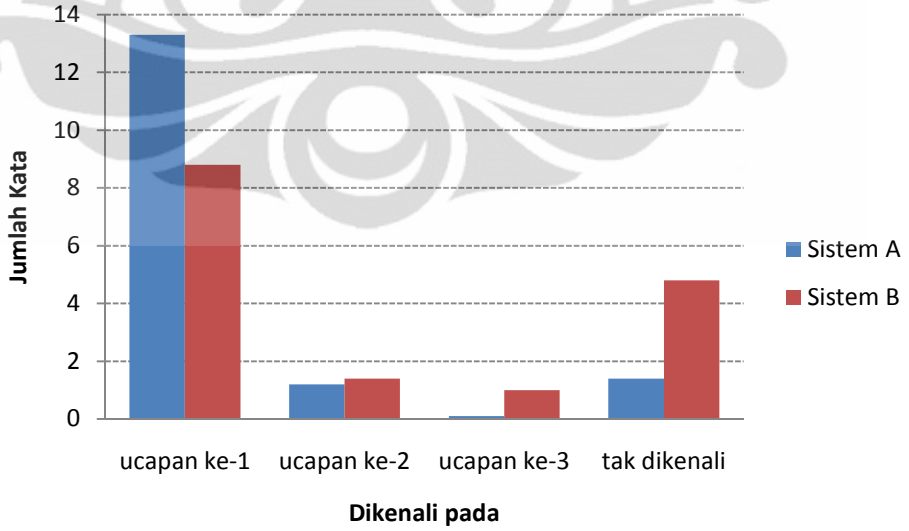
Gambar 5.3 merepresentasikan distribusi responden berdasarkan pengenalannya terhadap sebuah kata, atau distribusi kata berdasarkan kemudahannya dikenali oleh setiap responden. Di sistem A, rata-rata, 83% responden dapat mengenali satu kata di ucapan ke-1, atau dengan kata lain, 83% kata dapat dikenali satu responden di ucapan ke-1. Di sistem B, rata-rata, hanya

55% responden yang dapat mengenali satu kata di ucapan ke-1, dan rata-rata, 30% responden tidak mengenali satu kata.



Gambar 5.3. Distribusi Responden atau Kata untuk Setiap Kejadian

Berdasarkan data dari dua sistem tersebut, Gambar 5.4 menampilkan perbandingan kedua sistem dalam hal berapa kata yang dapat langsung dikenali oleh rata-rata responden di ucapan ke-1, harus diulangi hingga 2 kali pengucapan, 3 kali pengucapan, dan yang tidak bisa dikenali meskipun telah diulangi sebanyak 3 kali.



Gambar 5.4. Kemudahan Kata untuk Dikenali Tiap Responden

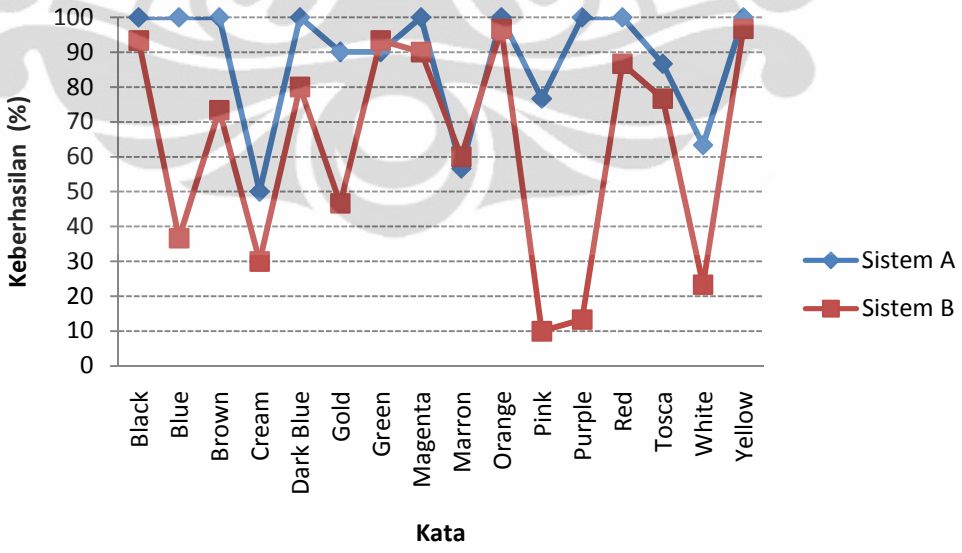
Untuk memberikan nilai kuantitatif terhadap keberhasilan masing-masing sistem, diberlakukan pembobotan pada 4 kejadian yang ada, seperti tercantum pada Tabel 5.2 di bawah ini.

Tabel 5.2. Pembobotan Tiap Kejadian

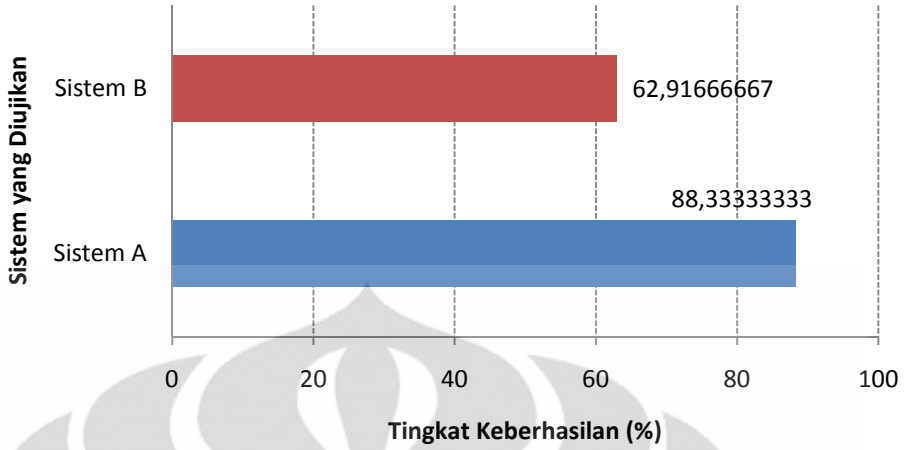
No.	Kejadian	Bobot
1	keberhasilan pengenalan pada pengucapan ke-1	3
2	keberhasilan pengenalan pada pengucapan ke-2	2
3	keberhasilan pengenalan pada pengucapan ke-3	1
4	kata tidak berhasil dikenali meskipun telah diulang pengucapannya hingga tiga kali	0

Melalui pembobotan tersebut, total nilai terbaik bagi sebuah sistem adalah 480, yakni untuk sistem yang semua responden (10 responden) dapat mengenali semua kata (16 kata) di ucapan ke-1 (bobot = 3).

Gambar 5.5 menunjukkan perbandingan kuantitatif keberhasilan sistem, dengan representasi nilai yang telah diubah menjadi bentuk persen untuk setiap kata (nilai 30 = 100%). Gambar 5.6 menunjukkan perbandingan kuantitatif dari keberhasilan kedua sistem untuk seluruh kata yang diujikan. Dari kedua grafik ini, dapat disimpulkan bahwa untuk menghasilkan kata yang dapat dikenali dengan mudah, sistem A lebih baik.

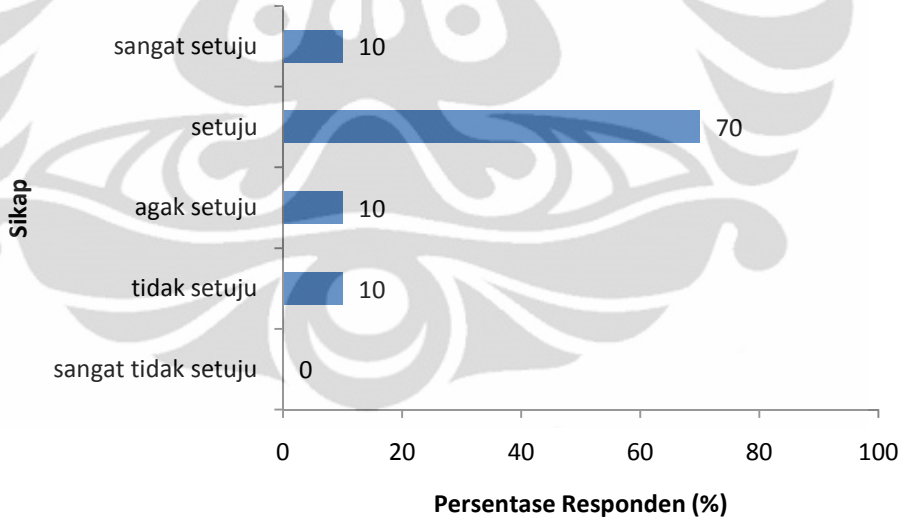


Gambar 5.5. Persentase Keberhasilan Kedua Sistem untuk Setiap Kata



Gambar 5.6. Perbandingan Keberhasilan Kedua Sistem

Untuk diintergrasikan dalam chormophore, sistem yang digunakan adalah sistem A. Tentang komentar responden atas pernyataan "suara yang diucapkan jelas bagi saya", dari keseluruhan pengujian tersebut, 5 responden menjawab agak setuju, 4 responden menjawab setuju, dan 1 responden menjawab sangat setuju.



Gambar 5.7. Sikap Responden atas Pernyataan "Suara yang Diucapkan Jelas bagi Saya"

5.2. Analisa Implementasi Pengenalan Suara dengan .NET dan SAPI

5.2.1. Responden

Untuk pengujian modul ini, diambil 10 responden yang merupakan mahasiswa Departemen Teknik Elektro FTUI dengan kisaran umur antara 20

hingga 25 tahun, asal daerah/logat yang berbeda-beda, dan kemampuan bahasa Inggris yang berbeda-beda.

5.2.2. Metode Pengujian

Pada uji coba ini, setiap satu responden akan menguji sistem pengenalan suara, dengan mengucapkan kata-kata tertentu yang sudah ada dalam *database*/*grammar* sistem. Untuk meminimalisasi variasi yang tidak diinginkan, sebelum pengujian dimulai, responden terlebih dahulu diperdengarkan bagaimana pengucapan kata-kata yang ditentukan dalam bahasa Inggris yang benar, yang diambil dari Google *translate*. Setiap sistem, diuji dengan 2 kondisi, kondisi tenang dan kondisi bising, dan dalam setiap kondisi, pengucapan kata dilakukan maksimal hingga 3 kali pengucapan.

Kondisi bising penilis buat dengan memutar suara-suara tak berarti dengan volume tertentu, sehingga suara tersebut masuk ke mikrofon yang digunakan responden. Responden diupayakan untuk mengucapkan kata yang diujikan dengan volume yang tetap, relatif terhadap volume suara-suara bising yang ada. Volume suara bising ini adalah antara 10% hingga 20% dari volume rata-rata ucapan responden. Pengukuran volume ini dilakukan dengan aplikasi Audacity.

Angka 10% dan 20% ini dipilih berdasarkan observasi acak terhadap 3 sistem CAPTCHA yang menyediakan fitur audio (dengan sampling *convenience*, volume derau tertinggi yang ditemui adalah 22,22%, lihat Tabel 5.3), dan berdasarkan penelitian sebelumnya [3] yang telah menguji performa beberapa *engine* pengenalan suara (SAPI5.1 termasuk di dalamnya) dengan perbandingan sinyal terhadap derau (SNR) maksimal sebesar 20 dB. Berdasarkan persamaan

$$SNR_{dB} = 10\log_{10}\left(\frac{P_{speech}}{P_{noise}}\right) = 20\log_{10}\left(\frac{A_{speech}}{A_{noise}}\right)$$

nilai SNR tersebut setara dengan volume derau sebesar 10% dari volume ucapan.

Tabel 5.3. Level Derau dari Beberapa Sistem CAPTCHA

Nama situs	Volume CAPTCHA:Volume Derau			Persentase derau rata-rata (%)
	Sampel 1	Sampel 2	Sampel 3	
Captcha.biz	0,6:0,25	0,6:0,05	0,6:0,1	22,22
reCaptcha.net	0,35:0,1	0,3:0,05	0,3:0,03	18,42
Facebook.com	0,3:0,01	0,3:0,01	0,3:0,01	3,33

Selain melakukan pengujian terhadap modul ini, di akhir, responden juga diminta untuk mencoba menggunakan Chromophore, dan diminta untuk memberikan komentarnya atas pernyataan "memerintah dengan suara lebih menyenangkan".

5.2.3. Hasil Pengujian dan Analisis

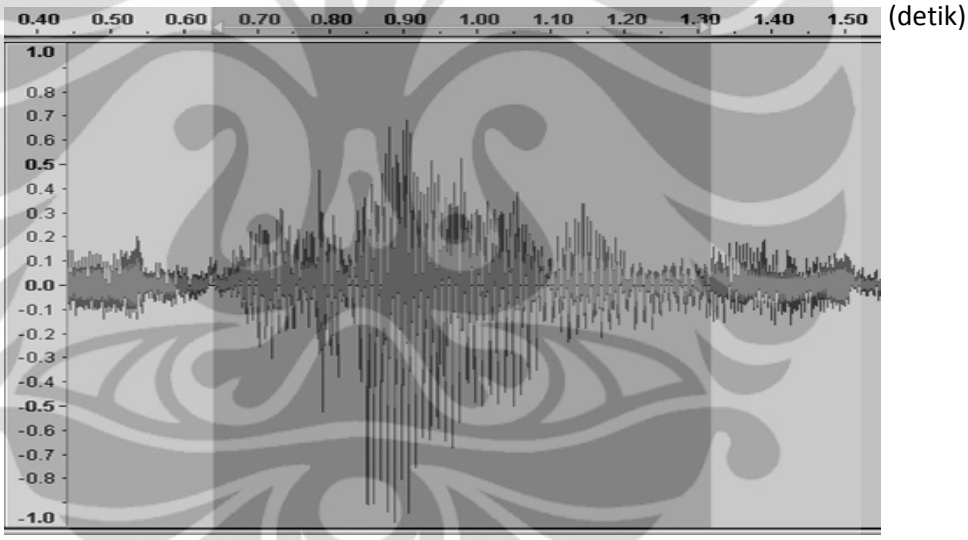
Tabel 5.4. Hasil Pengujian Sistem Pengenalan Suara

No	Kata	Jumlah Responden							
		Tenang				Bising			
		ke-1	ke-2	ke-3	Gagal	ke-1	ke-2	ke-3	Gagal
1	*Red*	3	2	1	4	6	0	1	3
2	*Orange*	2	5	0	3	6	2	0	2
3	*Yellow*	6	2	2	0	8	0	2	0
4	*Light Green*	3	4	2	1	8	1	0	1
5	*Green*	7	3	0	0	8	2	0	0
6	*Tosca*	5	3	0	2	6	1	1	2
7	*Light Blue*	5	3	1	1	6	3	0	1
8	*Blue*	9	1	0	0	8	1	0	1
9	*Purple*	3	3	0	4	5	1	2	2
10	*Magenta*	4	2	0	4	7	0	1	2
11	*Dark Green*	7	2	0	1	6	1	0	3
12	*Dark Orange*	4	4	1	1	5	1	1	3
13	*Dark Red*	5	3	0	2	4	1	2	3
14	*Dark Yellow Green*	5	3	0	2	5	0	1	4
15	*Greenish Gray*	7	0	1	2	4	2	2	2
16	*select mode*	6	2	1	1	5	2	1	2
17	*help*	6	4	0	0	7	1	1	1
18	*close*	10	0	0	0	8	1	1	0
19	*one*	9	1	0	0	8	1	0	1
20	*two*	9	1	0	0	8	1	0	1
21	*three*	8	2	0	0	9	0	0	1

Pada Tabel 5.4, kolom **Tenang** merepresentasikan hasil pengujian untuk sistem kondisi tenang, kolom **Bising** merepresentasikan hasil pengujian untuk sistem kondisi bising. Setiap kondisi dibagi lagi menjadi 4 kolom. Kolom **ke-1** berisi jumlah responden yang perintahnya dapat dikenali sistem pada ucapan pertama, kolom **ke-2** berisi jumlah responden yang perintahnya dapat dikenali

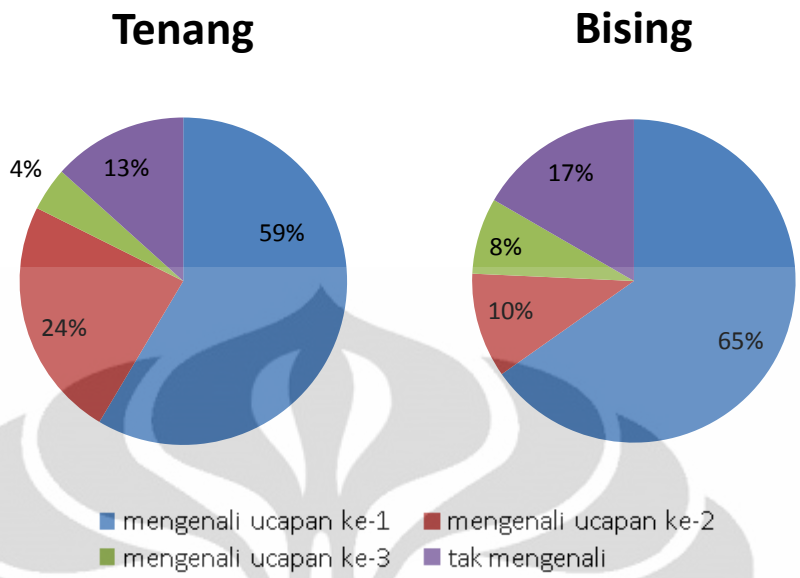
sistem pada ucapan kedua, kolom **ke-3** berisi jumlah responden yang perintahnya dapat dikenali sistem pada ucapan ketiga, dan kolom **Gagal** berisi jumlah responden yang perintahnya tidak dapat dikenali. Misalkan, angka **3** di bawah kolom **ke-1** di kolom **Tenang** pada baris kata **Red** menunjukkan bahwa ada 3 orang responden yang ucapannya atas kata "*red*" dapat dikenali sistem pada ucapan mereka yang pertama pada kondisi tenang.

Pada kondisi bising, pengukuran volume ucapan dan derau dilakukan dengan Audacity. Pada Gambar 5.8, di seluruh bagian terdapat *waveform* derau, sementara hanya di bagian yang berlatar lebih gelap terdapat *waveform* ucapan "*green*" dari responden. Pada gambar tersebut, derau berfluktuasi namun rata-rata ada di bawah amplitudo 0,2, sedangkan amplitudo ucapan responden bernilai maksimal 1.



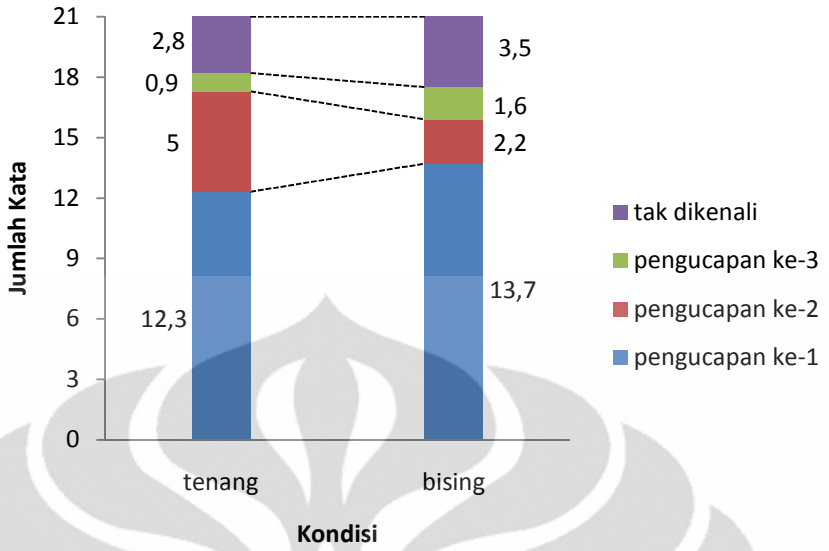
Gambar 5.8. *Waveform* Kata "*green*" pada Kondisi Bising

Gambar 5.9 merepresentasikan distribusi responden berdasarkan kemudahan ucapannya atas sebuah kata untuk dikenali, atau distribusi semua kata yang diucapkan setiap responden berdasarkan kemudahannya dikenali. Pada kondisi tenang, rata-rata ada 59% responden yang ucapannya atas satu kata berhasil dikenali dengan benar pada pengucapan ke-1, atau dengan kata lain 59% kata yang diucapkan satu responden dapat dikenali dengan benar pada pengucapan ke-1. Di kondisi bising, persentase keberhasilan pada pengucapan ke-1 justru meningkat, namun kegagalan pengenalan juga meningkat.



Gambar 5.9. Distribusi Responden atau Kata untuk Setiap Kejadian

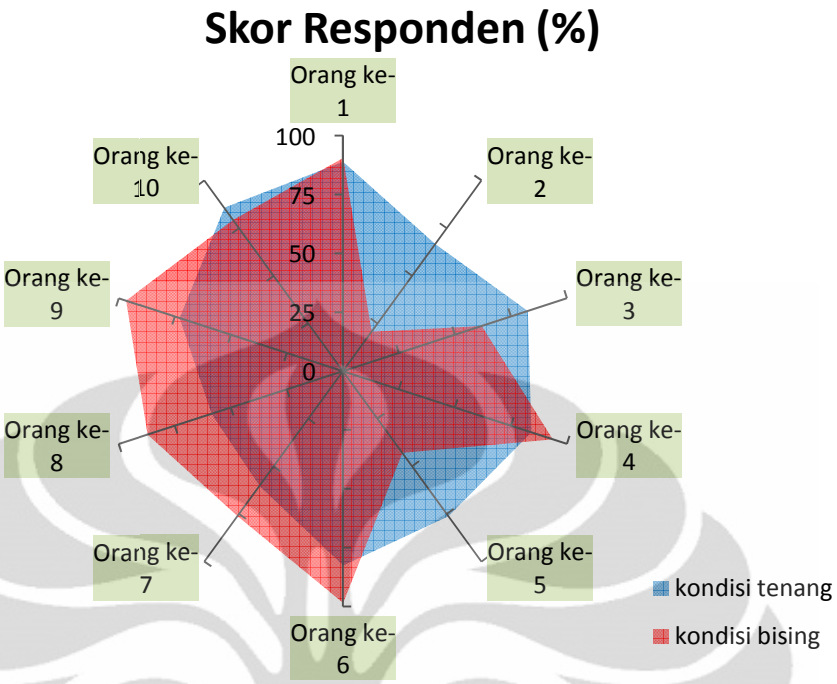
Gambar 5.10 menampilkan perbandingan antara kedua kondisi dalam hal jumlah kata di setiap kejadian yang ditampilkan secara kumulatif untuk mempermudah pengamatan transisi antarkondisi. Hal yang unik di sini, pada kondisi bising, justru lebih banyak kata yang terdeteksi dengan benar di ucapan ke-1. Bila dilihat dari teknis pelaksanaan pengujian, pengujian untuk kondisi bising dilakukan setelah pengujian untuk kondisi tenang. Urutan pengujian ini bisa jadi berpengaruh; membuat responden lebih lancar mengucapkan kata-kata yang diujikan (artikulasi). Responden menjadi terbiasa dengan sistem dalam hal pewaktuan penggunaan tombol *trigger*, kecepatan, dan intonasi pengucapan tiap fonem. Di pengujian ke dua, pengujian dalam kondisi bising, beberapa kata yang pada pengujian pertama harus membutuhkan pengulangan menjadi dapat dikenali dengan benar di ucapan ke-1.



Gambar 5.10. Distribusi Kata secara Kumulatif untuk Setiap Kejadian

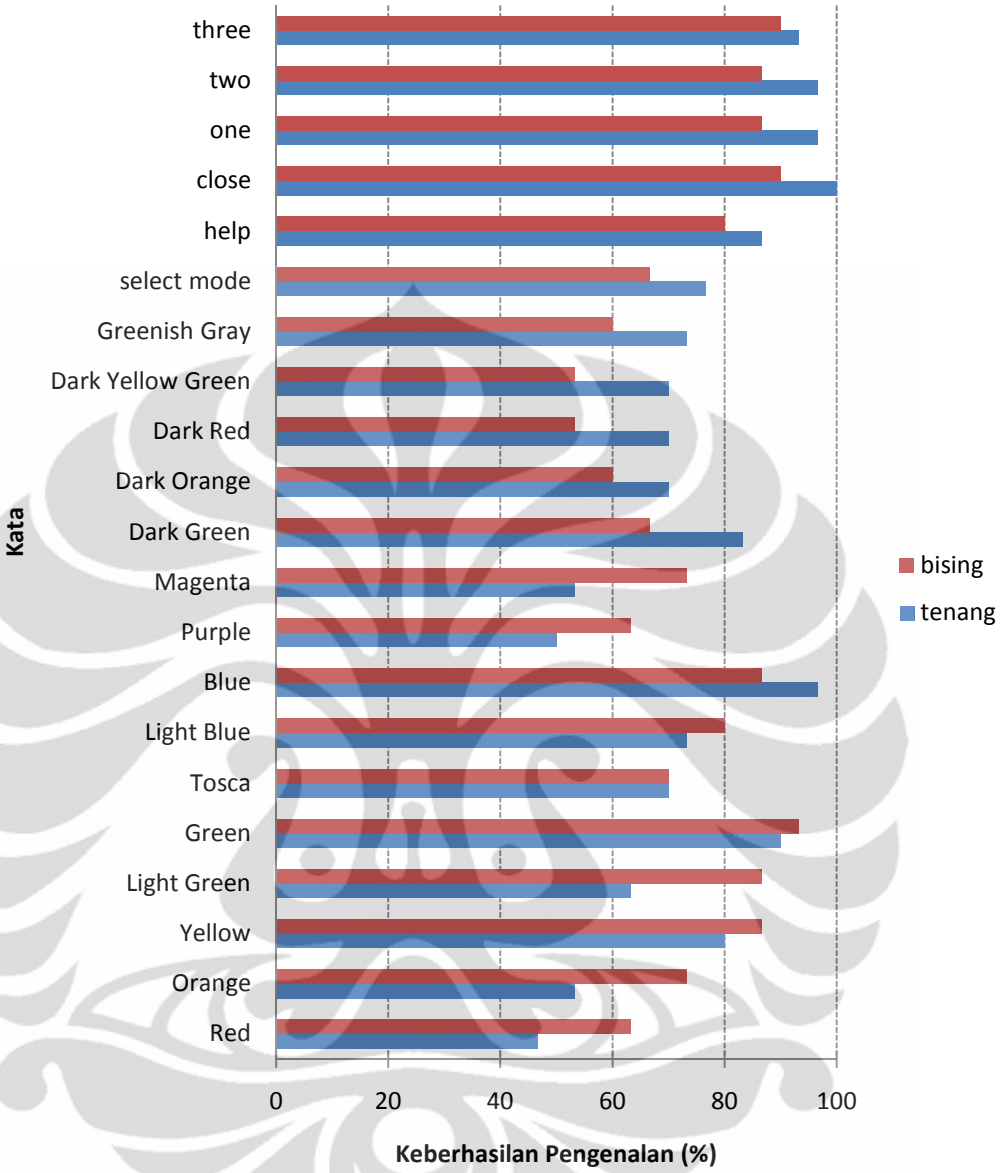
Untuk dapat memberikan skor kuantitatif terhadap performa sistem di dua kondisi tersebut, diberlakukan pembobotan terhadap setiap kejadian yang ada, seperti tercantum pada Tabel 5.2. Berdasarkan pembobotan tersebut, skor tertinggi yang dapat diperoleh seorang responden adalah 63, yakni ketika ia berhasil mengucapkan semua kata (21 kata) dan berhasil dikenali sistem pada pengucapan ke-1 (3 poin).

Gambar 5.11 menampilkan skor setiap responden dalam bentuk persentase untuk kedua kondisi. Dari sini, dapat diamati bahwa ada dua responden nomor 2 (Orang ke-2) dan 5 (Orang ke-5) memunculkan data pencilan saat kondisi bising. Namun tetap dilakukan perhitungan terhadap data ini karena 2 dari 10 data adalah angka yang cukup signifikan.

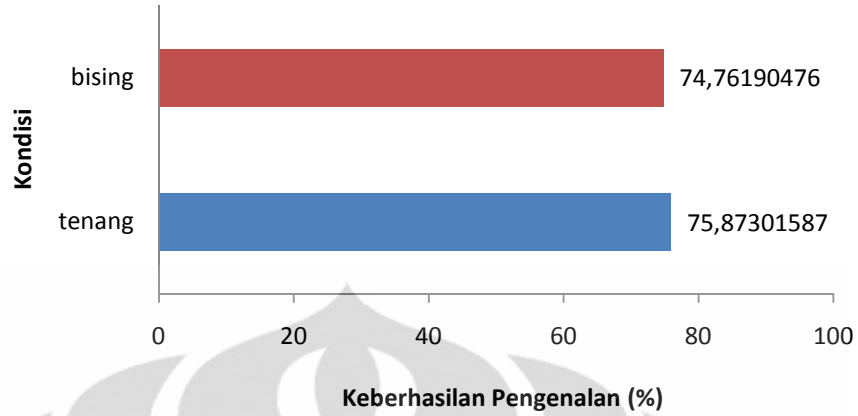


Gambar 5.11. Skor Tiap Responden (dalam Bentuk Persentase)

Kemudahan setiap kata untuk dikenali dapat direpresentasikan berdasarkan pembobotan sama seperti sebelumnya. Semakin tinggi skornya, semakin mudah kata tersebut untuk dikenali. Skor tertinggi yang dapat diperoleh sebuah kata adalah 30, yakni ketika kata tersebut berhasil dikenali di ucapan ke-1 (3 poin) oleh semua responden (10 orang). Untuk menggambarkan kemudahan setiap kata untuk dikenali sistem, Gambar 5.12 menggambarkan skor setiap kata dalam wujud persentase keberhasilan dalam bentuk persen dari skor pada setiap kondisi. Selanjutnya, sebagai pembanding akhir, Gambar 5.13 menggambarkan kinerja sistem, sebagai akumulasi persentase keberhasilan setiap kata, pada kedua kondisi yang diujikan.

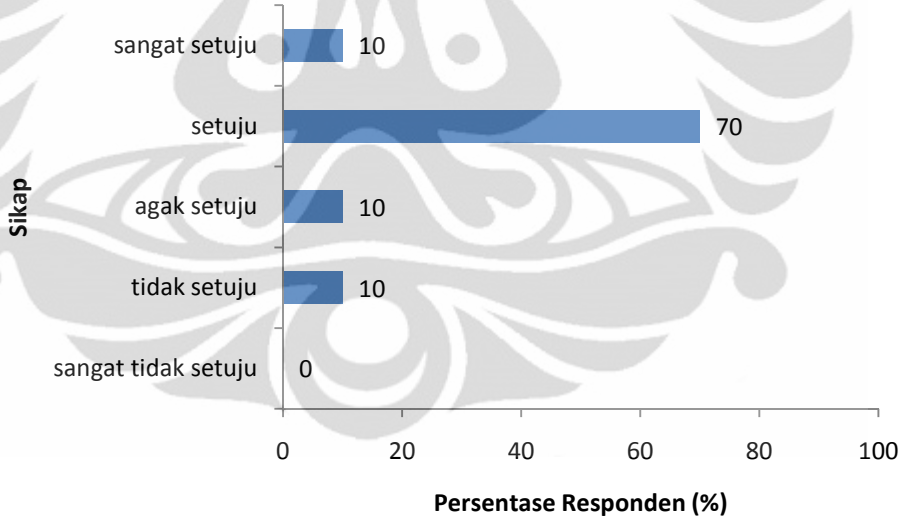


Gambar 5.12. Persentase Keberhasilan Pengenalan atas Setiap Kata



Gambar 5.13. Kinerja Sistem dalam Dua Kondisi

Adapun tentang pernyataan "memerintah dengan suara lebih menyenangkan", dari 10 responden yang telah mencoba aplikasi Chromophore, satu responden menyatakan tidak setuju, 1 responden menyatakan agak setuju, 7 responden menyatakan setuju, dan 1 responden menyatakan sangat setuju.



Gambar 5.14. Sikap Responden atas Pernyataan "Memerintah dengan Suara Lebih Menyenangkan"

5.3. Analisa Implementasi DTW untuk Pengenalan Suara

5.3.1. Metode Pengujian

Uji coba ini dilakukan dengan memutar suara pengucapan kata-kata yang berasal dari Google *Translate*, kemudian diperdengarkan ke mikrofon untuk

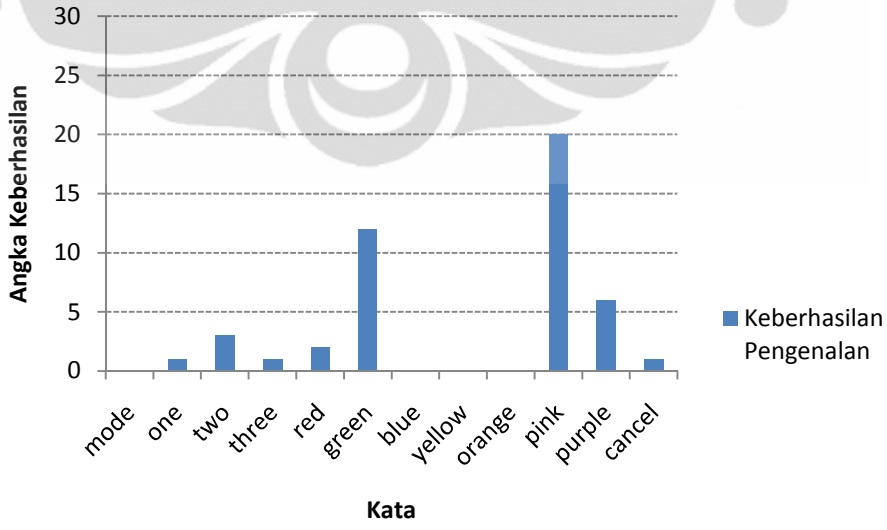
dideteksi oleh program yang berjalan dengan .NETCF 3.5 dan menggunakan DTW sebagai algoritma pengenalan suara. Sebelas kata yang diucapkan merupakan bagian dari 29 kata yang akan terdapat dalam sistem, yang diambil secara acak. Setiap kata diulangi hingga 30 kali, dan dilakukan perhitungan berapa kali kata tersebut berhasil dikenali oleh program.

5.3.2. Hasil Pengujian dan Analisis

Melalui pengujian yang cukup sederhana ini, diperoleh hasil sebagai berikut

Tabel 5.5. Tingkat Keberhasilan Pengenalan Setiap Kata

Kata	Keberhasilan	
	Angka	Persentase (%)
mode	0	0
one	1	3,333333
two	3	10
three	1	3,333333
red	2	6,666667
green	12	40
blue	0	0
yellow	0	0
orange	0	0
pink	20	66,66667
purple	6	20
cancel	1	3,333333



Gambar 5.15. Angka Keberhasilan Pengenalan Setiap Kata

Berdasarkan hasil tersebut, terlihat bahwa algoritma DTW semata belum cukup untuk diterapkan di sistem pengenalan suara. Meskipun telah diuji menggunakan *audio* yang berasal dari Google *Translate* yang sama seperti gelombang referensi yang ada, ternyata DTW masih belum dapat mengenali gelombang yang sama bila gelombang tersebut mendapatkan tambahan derau, perubahan panjang *byte*, dan amplitudo.

5.4. Analisa Implementasi Sistem Operasi Tertanam

5.4.1. Pengaruh Pemilihan BSP terhadap Kinerja WinCE6

Analisis dilakkukan terhadap variasi kinerja OS yang diakibatkan oleh BSP yang dipilih, sedangkan komponen dan *driver* yang disertakan dibuat seragam seperti yang telah dijelaskan pada bagian perancangan. Kinerja OS ini diamati melalui tiga parameter:

1. Ukuran berkas *image* dari OS pada saat *runtime*. Ukuran berkas ini dilihat dari informasi *used space* pada *local storage* di eBox, setelah OS dimuat, dikurangi berkas NK.bin dan berkas-berkas pendukung *booting* lainnya. Ukuran OS mmemiliki arti penting dalam pengembangan sistem tertanam dengan *resource* yang terbatas.
2. Waktu yang dibutuhkan untuk *cold boot-up* OS rata-rata. Waktu ini diukur dengan stopwatch digital, dimulai sejak BIOS menjalankan loadcepc.exe, hingga lampu indikator akses disk di eBox berhenti dan OS dimuat seluruhnya. Adapun untuk pengujian ini, setiap *image* disimpan di *local storage* eBox, semua *image* dirancang tanpa menabahkan aplikasi *startup*, dan pengujian dilakukan sebanyak 5 kali untuk setiap *image*.
3. Kelengkapan dukungan terhadap kebutuhan aplikasi Chromophore. Pengamatan ini dilakukan dengan mencoba menjalankan aplikasi Chromophore, menginventarisasi pesan kesalahan maupun peringatan yang muncul, dan menganalisis dependensinya secara teoritis dengan sumber pustaka yang ada.

Pengujian dilakukan atas hasil implementasi OS dengan tiga BSP. Tabel di bawah ini menyajikan data perbandingan antara ketiganya dalam hal ukuran

berkas *runtime image*, waktu yang dibutuhkan untuk *cold boot-up*, dan kekurangan dukungan perangkat lunak untuk kebutuhan aplikasi Chromophore.

Tabel 5.6. Pengaruh BSP terhadap Kinerja OS

BSP	Ukuran *Image* (KB)	Waktu boot-up (detik)	Kekurangan *Software*
CEPC: x86	24.909	13,82	- *Driver audio* - *Driver Webcam* - System.Windows.Forms, Version 2.0.0.0
ICOP_eBox 3310A _60GS: X86	26.071	19.03	- *Driver Webcam* - *System.Speech*, Version=3.0.0.0
ICOP_eBox 3310A _60H: X86	25.278	18.57	- *Driver Webcam* - *System.Speech*, Version=3.0.0.0

Dari tabel di atas, terlihat bahwa BSP CEPC memiliki ukuran dan waktu *boot-up* yang paling kecil. Di sisi lain, BSP ICOP_60GS dan ICOP_60H memiliki dukungan yang lebih dari pada BSP CEPC, yakni dengan mendukung *driver* audio. Hal ini wajar karena ICOP merupakan vendor yang memproduksi eBox, sehingga ia dapat menyediakan BSP yang mendukung semua perangkat kerasnya. Meskipun demikian, terlihat bahwa tidak ada BSP yang mendukung *driver webcam* yang akan digunakan.

Ukuran BSP ICOP_60GS yang lebih besar dan waktu *boot-up*nya yang lebih lama mungkin dikarenakan BSP ini mendukung RAM 512 MB. Karena eBox yang digunakan memiliki ukuran RAM sebesar 256 MB, setelan yang kurang tepat ini membuat OS mencoba menyiapkan pengalamatan untuk RAM berukuran 512 MB [24]. Meskipun gagal, hal ini membutuhkan tambahan waktu. Walaupun demikian, ini tidak berpengaruh pada kinerja OS saat OS telah dijalankan.

Untuk melanjutkan analisis, dibuatlah aplikasi yang tidak membutuhkan peran kamera. Saat aplikasi ini diuji, dari tiga jenis BSP tersebut, terdapat 2 jenis pesan kesalahan:

1. *File or assembly name 'System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089', or one of its dependencies, was not found.*

2. *File or assembly name 'System.Speech, Version=3.0.0.0, Culture=neutral, PublicKeyToken= 31BF3856AD364E35', or one of its dependencies, was not found.*

Kesalahan jenis pertama disebabkan karena aplikasi mencoba mengakses fungsionalitas dari *namespace System.Windows.Form* versi 2, namun dari pustaka .NET (*PublicKeyToken* mengindikasikan hal tersebut), bukan dari .NETCF [42]. Karena OS WinCE6 hanya mendukung *platform* .NETCF, aplikasi gagal dijalankan.

Kesalahan jenis kedua disebabkan karena aplikasi mencoba mengakses fungsionalitas dari *namespace System.Speech* versi 3, dengan pustaka .NET. Selain WinCE6 tidak mendukung *platform* .NET, OS juga tidak dapat memberikan fungsionalitas *Speech* karena tidak ada *speech engine*.

Pesan kesalahan yang muncul merupakan kesalahan yang pertama terdeteksi oleh OS, berdasarkan urutan kode program dieksekusi. Boleh jadi, bukan hanya dua kesalahan di atas yang muncul karena aplikasi Chromophore menggunakan banyak fungsionalitas dari pustaka .NET, seperti pengolahan gambar dan video yang dibutuhkan modul transformasi dan pendeteksi warna, dan pendeteksi ujung jari.

5.4.2. Kinerja WES09 untuk eBox dan Chromophore

Analisis berikutnya adalah untuk mengetahui seberapa cocok WES09 untuk diimplementasikan sebagai OS pada perangkat tertanam seperti eBox, dan seberapa cocok WES09 menangani aplikasi Chromophore.

OS WES09 jauh lebih besar dari pada WinCE6, apalagi saat ditambahkan komponen .NET yang dibutuhkan untuk menjalankan aplikasi. Bila dilihat dari ukuran saat selesai kompilasi, 300 MB adalah ukuran yang cukup untuk ditempatkan di eBox dengan *storage* sebesar 488 MB. Namun untuk dapat digunakan, WES09 harus menjalankan FBA terlebih dahulu, dan ini membuat ukuran berkas OS menjadi hampir 700 MB. Dengan ukuran sebesar ini, tentunya pembuatan sistem tertanam dengan menggunakan WES09 harus dipertimbangkan kembali.

Setelah mencoba mengurangi komponennya dengan lebih hati-hati, diketahui bahwa ukuran yang besar ini karena komponen .NET dan segala

dependensinya. Dengan demikian, komponen ini tetap dibutuhkan untuk menjalankan Chromophore dan memilih untuk mengganti *storage* menjadi USB *flash disk* yang berukuran 4 GB. Penggantian ini membuat proses baca dan tulis berkas menjadi lebih lama. Dari pengamatan selanjutnya, .NET juga menjalankan proses tersembunyi bernama mscorsvw.exe untuk menuntaskan instalasi .NET pada saat FBA. Proses ini berjalan sekitar 15 menit dan membuat CPU bekerja hingga 100%, namun hanya dijalankan sekali saja, yakni tepat setelah *reboot* setelah instalasi .NET [43]. Dari sini, dapat disimpulkan bahwa produk dengan WES09 dan .NET tidak boleh langsung dilepas ke pasar untuk digunakan *end-user*, karena proses *boot* pertamanya adalah FBA dan *boot* keduanya otomatis menjalankan mscorsvw.exe.

Setelah *boot* untuk yang ketiga kalinya, WES sudah berjalan normal. Mulai saat tersebut, modifikasi tambahan untuk optimasi, pemasangan USB kamera, dan eksekusi aplikasi Chromophore dapat dijalankan.

Tidak ada pesan kesalahan saat Chromophore dijalankan, namun kecepatan responnya dalam menampilkan gambar dari kamera, dan juga dalam pengenalan suara masih tergolong lambat dan kurang nyaman untuk digunakan dengan sering. Pengamatan dengan *task manager* menunjukkan bahwa Chromophore hampir selalu membuat prosesor bekerja dengan beban 100%.

5.4.3. Pertimbangan dalam Pemilihan Sistem Operasi Tertanam

WES09 cocok untuk digunakan pada sistem tertanam yang memang membutuhkan komputasi rumit dan fungsionalitas yang luas. Dengan WES09, pengembangan sistem tertanam yang mirip dengan aplikasi *desktop* jadi lebih mudah. Pada posisi ini, prioritas harus dipilih. Apakah ingin mudah mengembangkan aplikasi dan merelakan penggunaan *resource* yang besar demi OS, atau mengusahakan pengembangan aplikasi dengan lebih rumit agar penggunaan *resource*nya sedikit.

Pada percobaan ini, dipilih alternatif pertama, dengan memberikan tambahan storage berupa USB flash disk pada eBox agar pengembangan aplikasi tetap dapat dilakukan dengan mudah dan cepat. Pilihan kedua sebenarnya telah dicoba lebih dahulu, yakni dengan mengembangkan aplikasi fitur suara yang tidak

membutuhkan SAPI dan .NET 3.5, serta mengembangkan OS dengan WinCE6. Namun usaha ini menemui kendala dari segi dukungan *driver webcam* dan performa aplikasi dalam hal performa pengenalan dan sintesis suara yang tidak cukup baik.

BAB 6
PENUTUP

Kesimpulan dari penulisan skripsi ini adalah:

1. Sistem sintesis suara yang dibuat dengan penggabungan fonem memiliki tingkat keberhasilan 62,92%. Di sisi lain, sistem pengenalan suara yang dibuat hanya dengan metode DTW memiliki kinerja yang buruk dengan menyisakan banyak kata tidak dikenali sama sekali.
2. Fitur suara yang dibuat dengan memanfaatkan *namespace System.Speech* dari Microsoft memiliki tingkat keberhasilan 88,33% untuk sintesis suara, 75,87% untuk pengenalan suara di lingkungan tenang, dan 74,76% untuk pengenalan suara di lingkungan bising. Fitur suara ini mengharuskan OS memiliki pustaka .NET 3,5 dan SAPI sebagai *speech engine*nya.
3. WinCE6 cocok untuk pengembangan sistem tertanam asalkan terdapat dukungan pustaka dan *driver*. WinCE6 hanya memiliki pustaka .NETCF dan SAPI tidak bisa ditambahkan ke dalam WinCE6. WES09 lebih cocok digunakan untuk perangkat fungsi tunggal yang rumit daripada disebut sebagai perangkat tertanam. WES09 mendukung pustaka .NET 3,5 dan menerima *driver* dan komponen yang bisa ditambahkan ke Windows XP, termasuk SAPI. Pemilihan antara WinCE6 dan WES09 sebaiknya memperhatikan kebutuhan aplikasi dan batasan perangkat keras.
4. Implementasi Chromophore sebagai sistem bantuan penderita buta warna dengan interaksi suara dapat diterima pengguna dengan baik. Masalah yang tersisa tinggal bagaimana pemilihan OS dan perangkat keras yang akan digunakan.

Untuk pengkajian ke depan, dapat dilakukan penelitian berikut ini.

1. Pembuatan pustaka (*library*) pemrograman untuk implementasi fitur-fitur suara di sistem tertanam yang kemudian disediakan secara terbuka.
2. Pengujian kompatibilitas fitur suara dengan *platform* pengembangan perangkat tertanam lainnya.

DAFTAR REFERENSI

[1] Ohkubo, T., & Kobayashi, K. (2008). A Color Compensation Vision System for Color-blind People. *SICE Annual Conference* (p.1-3). Tokyo: IEEE.

[2] Turley, J. (2005, May 24). *Embedded systems survey: Operating systems up for grabs*. May 30, 2011. http://www.eetimes.com/discussion/other/4025539/Embedded-systems-survey-Operating-systems-up-for-grabs

[3] Lynch, R. L. (2003). *Speech Recognition Engine Comparison.* New Hampshire: Project54, University of New Hampshire.

[4] Poret, S., Jony, R. D., & Gregori, S. (2009). Image Processing for Colour Blindness Correction. *Science and Technology for Humanity (TIC-STH)* (p.1-3). Toronto: IEEE.

[5] Hoffman, P. S. (1999). Accommodating Color Blindness. *Cognetics Corporation Reprinted from Usability Interface* , p.12.

[6] McDowell, J. (2008). Design of a color sensing system to aid the color blind. *POTENTIALS, IEEE* , p.34-39.

[7] Rowe, M. H. (2002). Trichromatic color vision in primates. *News in Physiological Sciences , 17* (3), p.93-98.

[8] Sakurambo. (2007, July 23). *Cone Response*. December 18, 2010. http://en.wikipedia.org/wiki/File:Cone-response.svg

[9] *What Is Colorblindness and the Different Types?* (n.d.). December 9, 2010. http://colorvisiontesting.com/color2.htm

[10] Ananto, B. S. (2010). *Sistem Bantuan Penderita Buta Warna: Perancangan Antarmuka Pengguna dan Integrasi Hasil Pengolahan Sistem Realitas Tertambah.* Depok: Universitas Indonesia.

[11] Bailey, G., & Haddrill, M. (2009, October). *How Color Blindness Affects Your Daily Life*. December 15, 2010. http://www.allaboutvision.com/conditions/colordeficiency.htm

[12] Wicaksana, B. A. (2010). *Sistem Bantuan Penderita Buta Warna: Pendeteksian Warna, Pelacakan Gerakan (Motion Tracking), dan Tampilan Informasi Warna dengan Platform .NET dan EMGUCV Library.* Depok: Universitas Indonesia.

[13] Yoh, M.-S. (2001). The Reality of Virtual Reality. *Seventh International Convverence on Virtual Systems and Multimedia* (p. 666 - 674). California: IEEE.

[14] Bimber, Oliver, & Ramesh, R. (2006). *Spatial Augmented Reality.* Massachusetts: A K Peters.

[15] Smith, T. (2001). Concise Oxford English Dictionary (Tenth Edition). Oxford.

[16] daktari3. (2010, November 29). *Basic concepts of speech*. December 8, 2010. http://cmusphinx.sourceforge.net/wiki/tutorialconcepts

[17] Wijaya, D. T. (2010). *Penggunaan Voicebank Berbasis KVK untuk Pembuatan Suara Sintesis dengan Menggunakan Speech Synthesizer dan Singing Synthesizer berbasis UTAU*. Depok: Universitas Indonesia.

[18] Essa, E. M. (2007, Mar 10). *Concatenating Wave Files Using C# 2005*. April 23, 2011. http://www.codeproject.com/KB/audio-video/Concatenation_Wave_Files.aspx

[19] Akhmad Arman, A. (2004, August 23). *Teknologi Pemrosesan Bahasa Alami sebagai Teknologi Kunci untuk Meningkatkan Cara Interaksi antara Manusia dengan Mesin*. Februari 17, 2011. http://www.itb.ac.id/focus/focus_file/Pidato%20Ilmiah%20pada%20Sidang%20Terbuka%20PMB%202004.pdf

[20] *Konversi dari teks ke ucapan (text to speech)* . (2009, May 6). December 15, 2010. http://www.ittelkom.ac.id/library/index.php?view=article&catid=15%3Apemrosesan-sinyal&id=553%3Akonversi-dari-teks-ke-ucapan-text-to-speech&option=com_content&Itemid=15

[21] *Ketika Teks Tidak Lagi Bisu*. (2005, Maret 23). December 15, 2010. http://www.dudung.net/teknologi-informasi/ketika-teks-tidak-lagi-bisu.html

[22] casey. (2004, April 11). *Homemade Speech Recognition with .NETcf*. April 13, 2011. http://www.brains-n-brawn.com/default.aspx?vDir=noreco

[23] Rosa, L. (2006, October 1). *Fast and Robust Speech Recognition Based on Dynamic Time-Warping*. May 7, 2010. http://www.advancedsourcecode.com/dtwspeech.asp

[24] Intelligent Control on Processor. (2009). *eBox-3310A-MSJK Windows Embedded CE 6.0 R3 Jump Start Kit*. December 8, 2010. http://www.embeddedpc.net/eBox3310AMSJK/

[25] Microsoft. (2008). *Developing Speech Application*. December 8, 2010. http://www.microsoft.com/speech/developers.aspx

[26] Srinivas, S. (2004, May 18). *Speech Recognition using C#*. December 8, 2010. http://www.c-sharpcorner.com/UploadFile/ssrinivas/SpeeechRecognitionusingCSharp11222005054918AM/SpeeechRecognitionusingCSharp.aspx

[27] Brown, R. (2006). *Exploring New Speech Recognition And Synthesis APIs In Windows Vista*. January 13, 2011. http://msdn.microsoft.com/en-us/magazine/cc163663.aspx

[28] *A simple implementation of DTW(Dynamic Time Warping) in C#* . (2008, July 23). May 7, 2011. http://data-matters.blogspot.com/2008/07/simple-implementation-of-dtwdynamic.html

[29] Sari, R. F. (2008). Process. Depok, Jawa Barat.

[30] Sari, R. F. (2008). Introduction to Unified Modelling Language (UML). Depok, Jawa Barat.

[31] Ambler, S. W. (n.d.). *UML 2 Deployment Diagrams*. December 23, 2010. http://www.agilemodeling.com/artifacts/deploymentDiagram.htm

[32] Manaf, A. S. (2010). *Sistem Bantuan Penderita Buta Warna: Perancangan Sistem Tertanam berbasis Konsep Realitas Tertambah Suara dengan Metode Interaksi Langsung Pengguna dengan Objek Warna.* Depok: Universitas Indonesia.

[33] Chesnut, C. (2004, November 3). *Homemade Text-To-Speech with .NET*. April 13, 2011. http://www.generation5.org/content/2004/ttSpeech.asp

[34] MSDN. (2010). *Windows Embedded CE Downloads*. December 8, 2010. http://msdn.microsoft.com/windowsembedded/ce/dd430902.aspx

[35] Microsoft. (2011). *Applications and Services Development Catalog Items*. April 13, 2011. http://msdn.microsoft.com/en-us/library/ee485307.aspx

[36] Microsoft. (2011). *ATL Reference*. April 13, 2011. http://msdn.microsoft.com/en-us/library/t9adwcde.aspx

[37] Microsoft. (2011). *File and Disk Caching*. April 13, 2011. http://msdn.microsoft.com/en-us/library/ee489972.aspx

[38] Microsoft. (2011). *RAM-Based Registry*. April 13, 2011. http://msdn.microsoft.com/en-us/library/ee490547.aspx

[39] Microsoft. (2011). *Encoded Media Catalog Items*. April 13, 2011. http://msdn.microsoft.com/en-us/library/ee493972%28v=WinEmbedded.60%29.aspx

[40] Kumar, P. (2011, April 11). *How to Get USB Camera Working in WinCE 6.0 ARM Kit*. June 6, 2011. http://e-consystems.com/blog/how-to-get-usb-camera-working-in-wince6-arm-kit/

[41] Fairbairn, C. (2008, December 13). *Speech recognition and synthesis on Win CE 6.0*. March 20, 2011. http://social.msdn.microsoft.com/Forums/en-US/microsoftdeviceemu/thread/58233466-7c85-4b8a-ac7c-d822a4dd5068/

[42] Bargaoanu, L. (2006, June 30). *File or assembly name not found error*. April 30, 2011. http://social.msdn.microsoft.com/Forums/en-US/netfxbcl/thread/546a7be5-1140-4303-9b0f-382a274e6b7f/

[43] Notario, D. (2005, April 27). *What is mscorsvw.exe and why is it eating up my CPU? What is this new CLR Optimization Service?* May 12, 2011. http://blogs.msdn.com/b/davidnotario/archive/2005/04/27/412838.aspx