Jurnal Teknik Elektro, Vol. 11, No. 1, Maret 2018, 1-6
ISSN 1411-870X
DOI: 10.9744/jte.11.1.1-6

Kendali Suara Berbahasa Indonesia Untuk Automasi Rumah Tinggal

Rafael Bogdan, Petrus Santoso, Resmana Lim
Program Studi Teknik Elektro,Universitas Kristen Petra
Jl.Siwalankerto 121-131, Surabaya 60236, Indonesia
E-Mail: bogx94@gmail.com ; petrus@peter.petra.ac.id ; resmana@peter.petra.ac.id

***Abstrak*— Pembuatan sistem kendali suara berbahasa Indonesia ini bertujuan untuk mengaplikasikan bahasa Indonesia ke dalam sistem pengendali perangkat rumah tangga menggunakan perintah suara. Hal ini dibuat guna memberikan kemudahan bagi masyarakat Indonesia yang memiliki kemampuan terbatas dalam berbahasa asing.**

Fokus pembuatan sistem ini menggunakan Raspberry Pi sebagai "otak" dari keseluruhan sistem. Untuk mengenali perintah suara digunakan *Speech-To-Text* (STT) *engine* Julius. Pembuatan model bahasa dan model akustik menggunakan metode Hidden Markov Model (HMM), dengan bantuan perangkat lunak Hidden Markov Model Toolkit (HTK). Perekaman suara yang digunakan untuk *training* model akustik dilakukan dengan perangkat lunak Audacity. Penangkapan suara sistem menggunakan *webcam* USB dengan mikrofon *built-in*.

Metode perancangan dan pembuatan dimulai dari studi literatur, perancangan dan pembuatan sistem, pengujian sistem, dan penarikan kesimpulan. Hasil akhir, didapatkan model *training* terbaik, yang memiliki jumlah *sample training* sebanyak 26 *sample*, dan dapat mengenali 11 perintah. Dengan menggunakan model tersebut, sistem dapat mengenali perintah yang diberikan pengguna baru dengan kesuksesan sebesar 49,09% sebelum melakukan *training*, dan 63,63% sesudah melakukan *training* (sebanyak 4 *sample*). Perintah dari pengguna lama dapat dikenali dengan kesuksesan sebesar 89,09% sesudah ditambahkannya *training* dari pengguna baru. Batas maksimal intensitas kebisingan yang ditangkap mikrofon agar sistem dapat mengenali perintah adalah 69 dB. Semakin jauh jarak pengguna saat memberikan perintah, dan semakin tinggi intensitas kebisingan yang ditangkap, menyebabkan turunnya kemampuan sistem dalam mengenali perintah.

***Kata Kunci*— Raspberry Pi, *Speech-To-Text* (STT) *engine* Julius, Hidden Markov Model, Hidden Markov Model Toolkit, Audacity.**

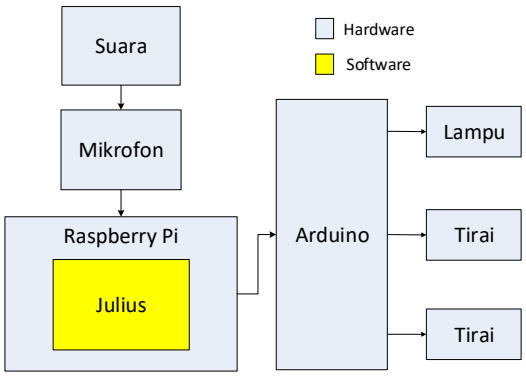
I. PENDAHULUAN

SISTEM KONTROL telah menjadi salah satu bagian dari teknologi yang berkembang dengan sangat pesat. Tidak hanya digunakan di dalam pabrik atau perindustrian saja, melainkan sistem kontrol telah digunakan pula pada perangkat-perangkat yang terdapat pada suatu rumah tinggal [1]. Namun penggunaan sistem kontrol dengan kendali suara pada perangkat-perangkat di dalam rumah tinggal masih tergolong minim. Kendali suara mampu meningkatkan efisiensi dalam sebuah sistem kontrol, sekaligus menjadi solusi khususnya bagi orang-orang yang memiliki keterbatasan fisik.

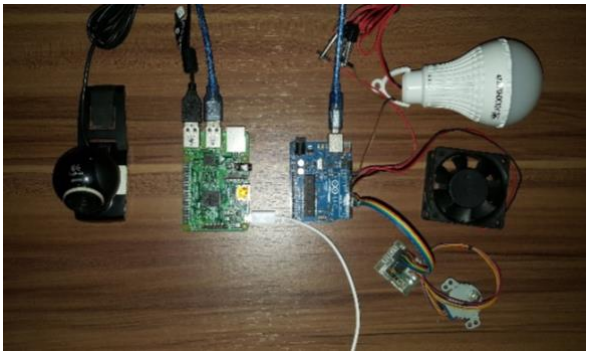
Beberapa perusahaan elektronik telah mengembangkan perangkat yang memungkinkan pengguna untuk mengontrol perangkat rumah tinggal dengan kendali suara [2]. Perangkat tersebut diantaranya adalah "Vocca" dari ActiVocal, "Echo" dari Amazon, "HomeKit" dari Apple, "Homey" dari Athom, "Wi-Fi Smart Thermostat with Voice Control" dari Honeywell, "Ivee Sleek" dari Interactive Voice, dan "Ubi" dari Unified Computer Intelligence Corporation. Namun, perangkat-perangkat tersebut memiliki beberapa kekurangan, yaitu terbatasnya bahasa yang dapat dikenali oleh sistem, menyebabkan hanya pengguna yang mampu menggunakan bahasa yang dikenali oleh sistem saja yang dapat menggunakan perangkat tersebut. Perangkat kendali suara tersebut belum dapat mengenali bahasa Indonesia, padahal banyak masyarakat Indonesia yang kurang mahir dalam berbahasa Inggris. Dengan adanya kekurangan tersebut, maka dibutuhkan perangkat sistem kontrol dengan kendali suara yang mampu mengenali perintah dalam bahasa Indonesia.

II. PERANCANGAN SISTEM

Pada bagian ini akan dijelaskan desain sistem kendali suara berbahasa Indonesia secara garis besar. Gambar 1 menunjukkan blok diagram sistem, dan Gambar 2 menunjukkan perangkat keras yang digunakan pada sistem.



Gambar 1. Blok diagram sistem



Gambar 2. Perangkat keras sistem (dari kiri: *webcam* USB dengan mikrofon *built-in*, Raspberry Pi, Arduino, lampu LED, kipas DC, *driver* dan motor *stepper*)

Sistem ini dibuat menggunakan beberapa perangkat keras, yaitu mikrofon (menggunakan *webcam* USB dengan mikrofon *built-in*) untuk menangkap suara yang menjadi *input* bagi sistem, Raspberry Pi yang menjadi "otak" dari sistem, Arduino

yang akan menerima perintah dari Raspberry Pi dan menjalankan perangkat, serta Lampu LED, kipas DC (pengganti AC) dan motor *stepper* (penggerak tirai) beserta *driver*-nya sebagai *output* sistem. Mikrofon (*webcam*) dihubungkan dengan Raspberry Pi menggunakan kabel USB. Raspberry Pi dan Arduino juga dihubungkan dengan kabel USB. Lampu LED, kipas DC, dan *driver* motor *stepper* dihubungkan pada I/O dari Arduino. Pada Raspberry Pi digunakan STT (*Speech-To-Text*) *engine* Julius, untuk mengenali perintah suara yang akan ditangkap oleh mikrofon.

III. DESAIN PERANGKAT LUNAK

A. *Desain Model Bahasa*

Dalam pembuatan model bahasa, *file* .grammar dan *file* .voca merupakan dua hal utama yang harus disusun terlebih dahulu. *File* .grammar berisi susunan golongan kata yang akan dikenali oleh sistem Berikut adalah isi dari *file* .grammar yang telah dibuat.

```
S : DIAM_AWAL DENGAR DIAM_AKHIR
DENGAR: PERINTAH1_V BENDA_N
DENGAR: BENDA_N PERINTAH2_V
DENGAR: BEJO
```

Gambar 3. *File* .grammar

Seperti yang tampak pada Gambar 3, yang berada di sebelah kiri titik dua (:) adalah simbol nonterminal, yaitu simbol yang dapat digantikan oleh simbol lain. Sedangkan di sebelah kanan titik dua adalah simbol terminal, yaitu simbol yang mewakili suatu nilai konstan [3]. Dalam Julius, simbol terminal mewakili "kategori kata", dan harus didefinisikan dalam *file* .voca.

File .voca berisi semua definisi kategori kata yang telah didaftar di dalam *file* .grammar. Definisi kategori kata yang dimaksudkan adalah membuat sebuah daftar kata yang tergolong dalam setiap kategori kata beserta dengan fonem penyusun tiap kata. Kata-kata yang merupakan kategori kata ditandai dengan tanda % (persen). Isi dari *file* .voca yang telah dibuat dapat dilihat pada Gambar 4.

```
% DIAM_AWAL
<s>        sil

% DIAM_AKHIR
</s>       sil


% PERINTAH1_V
HIDUPKAN        h iy d uw p k aa n
MATIKAN        m aa t iy k aa n
BUKA        b uw k aa
TUTUP        t uw t uw p

% PERINTAH2_V
TAMBAH        t aa m b aa h
KURANG        k uw r aa ng

% BENDA_N
LAMPU        l aa m p uw
AC        aa s ee
TIRAI        t iy r ay

% BEJO
BEJO        b eh jj oh
```

Gambar 4. *File* .voca

File .grammar dan .voca yang telah dibuat tidak dapat dibaca langsung oleh Julius, melainkan harus di-*compile* sehingga didapatkan *file* .dfa, .term, dan .dict. Proses *compile* dilakukan dengan menjalankan *script* mkdfa.jl yang telah disediakan oleh VoxForge [4].

B. *Desain Model Akustik*

Untuk membuat model akustik, diperlukan beberapa *file*, yaitu *file* VoxForgeDict.txt, prompts.txt, dan hasil rekaman *training*. *File* VoxForgeDict.txt adalah sebuah *file* kamus kata-kata disertai dengan fonem penyusunnya masing-masing. Kata-kata yang digunakan dalam sistem harus dicantumkan ke dalam *file* ini, serta kata-kata lain dengan tujuan untuk melengkapi fonem bahasa Indoensia yang ada, sehingga pengembangan sistem bisa menjadi lebih mudah. Berikut adalah potongan isi *file* VoxForgeDict.txt.

```
ABU          [ABU]     aa b uw
AC           [AC]    aa s ee
AIR          [AIR]     ay r
AKU          [AKU]     aa k uw
ALMARI       [ALMARI]  aa l m aa r iy
AMPAS        [AMPAS]   aa m p aa s
BAGAI        [BAGAI]   b aa g ay
BAGAIMANA [BAGAIMANA] b aa g ay m aa n aa
BAHASA       [BAHASA]  b aa h aa s aa
BEJO         [BEJO]    b eh jj oh
BELAKANG  [BELAKANG]  b eh l aa k aa ng
BELERANG  [BELERANG]  b eh l ae r aa ng
BENAR        [BENAR]   b eh n aa r
BETUL        [BETUL]   b eh t uw l
BIJAKSANA [BIJAKSANA] b iy jj aa k s aa n aa
BINTANG      [BINTANG] b iy n t aa ng
BUKA         [BUKA]    b uw k aa
BUMBU        [BUMBU]   b uw m b uw
CAPUNG       [CAPUNG]  c aa p uw ng
CENDEKIAWAN [CENDEKIAWAN]  c eh n d eh k iy aa w aa n
CERI         [CERI]    c ae r iy
COBA         [COBA]    c oh b aa
DALAM        [DALAM]   d aa l aa m
DAPUR        [DAPUR]   d aa p uw r
DELAPAN      [DELAPAN] d eh l aa p aa n
```

Gambar 5. *File* VoxForgeDict.txt

File prompts.txt berisi daftar kata-kata yang akan dilatih/*train*. Daftar ini terdiri dari beberapa *sample* yang masing-masing *sample*-nya tersusun dari beberapa kata. Kata-kata yang harus dilatih adalah yang akan digunakan dalam sistem, serta kata-kata lain untuk melengkapi *training* fonem-fonem bahasa Indonesia lainnya. Berikut adalah penggalan *file* prompts.txt.

```
*/sample1 HIDUPKAN LAMPU MATIKAN LAMPU HIDUPKAN LAMPU
MATIKAN LAMPU
*/sample2 HIDUPKAN AC MATIKAN AC HIDUPKAN AC MATIKAN
AC
*/sample3 BUKA TIRAI TUTUP TIRAI BUKA TIRAI TUTUP
TIRAI
*/sample4 LAMPU TAMBAH LAMPU KURANG TIRAI TAMBAH TIRAI
KURANG
*/sample5 BEJO BUKA TIRAI BEJO TUTUP TIRAI
*/sample6 BEJO TIRAI TAMBAH BEJO TIRAI KURANG
*/sample7 NOL NOL(2) SATU DUA TIGA EMPAT LIMA
*/sample8 ENAM TUJUH DELAPAN SEMBILAN SEPULUH
*/sample9 AKU ABU AIR AMPAS ALMARI
*/sample10 BENAR BETUL BIJAKSANA BUMBU
```

Gambar 6. *File* prompts.txt

Perekaman suara dilakukan menggunakan Audacity, dan mengikuti daftar yang telah disusun pada *file* prompts.txt. Properti yang harus diperhatikan saat merekam adalah *sample rate format* dalam 16000Hz, *sample format* dalam 16-bit, *channel* dalam pengaturan mono, dan *file* hasil *export* dalam format WAV (Microsoft 16 bit PCM).

File .wav yang telah didapatkan harus diubah ke dalam format MFCC sehingga dapat digunakan oleh Julius. Untuk pengubahan ini, perlu dibuat *file* codetrain.scp terlebih dahulu.

```
../train/wav/sample1.wav ../train/mfcc/sample1.mfc
../train/wav/sample2.wav ../train/mfcc/sample2.mfc
../train/wav/sample3.wav ../train/mfcc/sample3.mfc
../train/wav/sample4.wav ../train/mfcc/sample4.mfc
../train/wav/sample5.wav ../train/mfcc/sample5.mfc
../train/wav/sample6.wav ../train/mfcc/sample6.mfc
../train/wav/sample7.wav ../train/mfcc/sample7.mfc
../train/wav/sample8.wav ../train/mfcc/sample8.mfc
../train/wav/sample9.wav ../train/mfcc/sample9.mfc
../train/wav/sample10.wav ../train/mfcc/sample10.mfc
```

Gambar 7. *File* codetrain.scp

Proses yang harus dijalankan berikutnya adalah melakukan *training* dari *file-file* yang telah dibuat. Untuk melakukan *training*, dibutuhkan beberapa *script* Julia dari VoxForge untuk menjalankan beberapa fungsi. *Script* yang harus disalin beserta deng,an fungsinya masing-masing adalah sebagai berikut [5]:

- prompts2wlist.jl untuk membuat daftar kata dari *file* prompts.txt
- prompts2mlf.jl untuk membuat *master label file* (berisi daftar label dari setiap baris pada *file* prompts.txt)
- mktrihed.jl, fixfulllist.jl, dan mkclscript.jl untuk membuat *tied-state triphone* (rangkaian tiga fonem)
- trainAM.jl untuk *training* model akustik

Proses berikutnya adalah menjalankan script trainAM.jl. Script trainAM.jl akan melakukan proses *training* model akustik dengan cara menjalankan seluruh *script* dan perintah-perintah HTK secara otomatis. Perintah HTK dijalankan untuk membuat model statistik HMM.

Setelah proses *training* selesai, maka akan didapatkan *file* hmmdefs dan *file* tiedlist. *File* hmmdefs berisi data lengkap mengenai model statistik HMM, sedangkan *file* tiedlist berisi fonem-fonem beserta kombinasi fonem yang berpengaruh dan akan dikenali oleh Julius. Dua file inilah yang menjadi model akustik bagi Julius.

IV. PENGUJIAN SISTEM DAN ANALISA

A. *Pengujian Model Bahasa dan Model Akustik*

Tujuan dari pengujian ini adalah untuk mengetahui model *training* yang terbaik bagi sistem. Pengujian dilakukan dengan cara menyiapkan beberapa model yang akan diuji, menyiapkan skenario urutan perintah, dan menyiapkan rekaman suara untuk pengujian. Setelah itu memainkan rekaman suara sesuai dengan skenario perintah sebanyak sepuluh kali dengan jarak satu meter dari mikrofon, dan intensitas kebisingan seminimal mungkin (lebih kurang tertangkap oleh mikrofon sebesar 55 dB). Seluruh model dalam pengujian ini merupakan *training* dari satu orang pengguna, begitu pula dengan rekaman suara yang akan digunakan untuk pengujian merupakan rekaman suara dari satu orang pengguna.

1. Model Satu

Model ini memiliki 48 *sample training*, dengan jumlah perintah yang dikenali sebanyak 41 perintah. Berikut adalah daftar skenario perintah beserta hasil pengujian model satu.

Tabel 1. Hasil Pengujian Model Satu

Urutan Perintah (10 kali)	Dikenali (kali)	Kesuksesan (%)
BEJO	10	100
NYALAKAN LAMPU	0	0
MATIKAN LAMPU	3	30
LAMPU MENYALA	0	0
LAMPU MATI	4	40
HIDUPKAN LAMPU	5	50
LAMPU HIDUP	2	20
TERANGKAN LAMPU	0	0
REDUPKAN LAMPU	0	0
NYALAKAN AC	0	0
MATIKAN AC	9	90
AC MENYALA	4	40
AC MATI	8	80
HIDUPKAN AC	7	70
AC HIDUP	6	60
BUKA TIRAI	9	90
TUTUP TIRAI	8	80
TIRAI TERBUKA	1	10
TIRAI MENUTUP	0	0
TERANGKAN TIRAI	1	10
REDUPKAN TIRAI	2	20
NYALAKAN RUANGAN	0	0
RUANGAN MENYALA	0	0
HIDUPKAN RUANGAN	0	0
RUANGAN HIDUP	0	0
MATIKAN RUANGAN	0	0
RUANGAN MATI	0	0
TERANGKAN RUANGAN	0	0
REDUPKAN RUANGAN	0	0
RUANGAN NOL PERSEN	0	0
LAMPU NOL(2) PERSEN	0	0
LAMPU SEPULUH PERSEN	0	0
TIRAI DUAPULUH PERSEN	0	0
RUANGAN TIGAPULUH PERSEN	3	30
LAMPU EMPATPULUH PERSEN	4	40
TIRAI LIMAPULUH PERSEN	0	0
RUANGAN ENAMPULUH PERSEN	0	0
LAMPU TUJUHPULUH PERSEN	2	20
TIRAI DELAPANPULUH PERSEN	0	0
RUANGAN SEMBILANPULUH PERSEN	0	0
TIRAI SERATUS PERSEN	2	20
Persentase Kesuksesan Rata-rata		**21.95121951**

Dari hasil pengujian yang didapatkan, ternyata persentase kesuksesan sistem dalam mengenali perintah dari model satu hanya sebesar 21,95%. Nilai yang sangat jauh dari baik ini bisa disebabkan karena *training* yang kurang bagus, baik kualitas maupun jumlah *sample*. Jumlah *sample* yang terlalu banyak akan menyebabkan pemilihan kualitas *training* menjadi kurang selektif. Selain itu, terlalu banyaknya jumlah kata serta kombinasi perintah yang digunakan bisa menyebabkan sistem menjadi lebih sukar mengenali kata-kata secara spesifik. Adanya kata-kata tertentu yang susah dikenali oleh sistem juga dapat menyebabkan sistem menjadi tidak dapat mengenali perintah secara utuh. Hal lain yang tampak dari Tabel 1 adalah kurang dikenalinya perintah dengan kata-kata yang menggunakan suara *nasal*, seperti fonem 'ny' dan 'ng', serta untuk perintah *leveling.* Pada pengujian model berikutnya akan menggunakan model bahasa sama dengan yang digunakan pada model satu, namun dilakukan penambahan jumlah *sample training*.

2. Model Dua

Model dua memiliki jumlah perintah yang dikenali sama dengan model satu, yaitu sebanyak 41 perintah. Untuk jumlah *sample training* dilakukan penambahan dari model satu, yaitu sebanyak 7 *sample.* Model diuji dengan data rekaman yang sama dengan model satu. Berikut adalah hasil yang didapatkan dari pengujian model dua.

Tabel 2. Hasil Pengujian Model Dua

Urutan Perintah (10 kali)	Dikenali (kali)	Kesuksesan (%)
BEJO	10	100
NYALAKAN LAMPU	2	20
MATIKAN LAMPU	5	50
LAMPU MENYALA	1	10
LAMPU MATI	7	70
HIDUPKAN LAMPU	6	60
LAMPU HIDUP	4	40
TERANGKAN LAMPU	3	30
REDUPKAN LAMPU	1	10
NYALAKAN AC	0	0
MATIKAN AC	9	90
AC MENYALA	6	60
AC MATI	8	80
HIDUPKAN AC	8	80
AC HIDUP	8	80

BUKA TIRAI	5	50
TUTUP TIRAI	8	80
TIRAI TERBUKA	3	30
TIRAI MENUTUP	1	10
TERANGKAN TIRAI	3	30
REDUPKAN TIRAI	3	30
NYALAKAN RUANGAN	0	0
RUANGAN MENYALA	0	0
HIDUPKAN RUANGAN	5	50
RUANGAN HIDUP	0	0
MATIKAN RUANGAN	4	40
RUANGAN MATI	0	0
TERANGKAN RUANGAN	0	0
REDUPKAN RUANGAN	0	0
RUANGAN NOL PERSEN	0	0
LAMPU NOL(2) PERSEN	3	30
LAMPU SEPULUH PERSEN	6	60
TIRAI DUAPULUH PERSEN	0	0
RUANGAN TIGAPULUH PERSEN	0	0
LAMPU EMPATPULUH PERSEN	7	70
TIRAI LIMAPULUH PERSEN	0	0
RUANGAN ENAMPULUH PERSEN	0	0
LAMPU TUJUHPULUH PERSEN	5	50
TIRAI DELAPANPULUH PERSEN	0	0
RUANGAN SEMBILANPULUH PERSEN	0	0
TIRAI SERATUS PERSEN	4	40
Persentase Kesuksesan Rata-rata		**32.9268**

Dari hasil pengujian, didapatkan persentase kesuksesan rata-rata sebesar 32,9268%. Nilai ini menunjukkan adanya peningkatan dari nilai yang didapatkan pada model satu. Dengan kata lain, penambahan jumlah *sample training* akan memberikan pengaruh terhadap sistem, namun tidak signifikan. Selain itu, dari hasil pengujian model dua, ternyata perintah dengan kata-kata yang menggunakan suara *nasal* dan perintah *leveling* masih sukar untuk dikenali. Karena itu pada model berikutnya akan dilakukan pengurangan duplikasi perintah serta pengubahan terhadap bentuk perintah *leveling*.

3. Model Tiga

Model ketiga memiliki jumlah *sample training* sama dengan jumlah *sample training* model kedua, yaitu sebanyak 55 *sample*. Perintah yang akan dikenali sistem pada model tiga lebih sedikit daripada model sebelumnya, berjumlah 30 perintah, dengan meniadakan perintah dalam kalimat pasif, serta mengubah bentuk perintah *leveling* (tidak menggunakan persentase, melainkan dengan tingkatan pencahayaan). Hasil pengujian model ketiga ditunjukkan pada Tabel 3.

Tabel 3. Hasil Pengujian Model Tiga

Urutan Perintah (10 kali)	Dikenali (kali)	Kesuksesan (%)
BEJO	10	100
NYALAKAN LAMPU	4	40
MATIKAN LAMPU	8	80
HIDUPKAN LAMPU	7	70
TERANGKAN LAMPU	3	30
REDUPKAN LAMPU	3	30
NYALAKAN AC	4	40
MATIKAN AC	9	90
HIDUPKAN AC	8	80
BUKA TIRAI	7	70
TUTUP TIRAI	8	80
TERANGKAN TIRAI	5	50
REDUPKAN TIRAI	4	40
NYALAKAN RUANGAN	0	0
MATIKAN RUANGAN	4	40
HIDUPKAN RUANGAN	3	30
TERANGKAN RUANGAN	2	20
REDUPKAN RUANGAN	2	20
LAMPU TINGKAT NOL	1	10
TIRAI TINGKAT NOL(2)	5	50
RUANGAN TINGKAT SATU	0	0
LAMPU TINGKAT DUA	3	30
TIRAI TINGKAT TIGA	4	40
RUANGAN TINGKAT EMPAT	0	0
LAMPU TINGKAT LIMA	6	60
RUANGAN TINGKAT ENAM	0	0
TIRAI TINGKAT TUJUH	7	70
LAMPU TINGKAT DELAPAN	4	40
TIRAI TINGKAT SEMBILAN	5	50
RUANGAN TINGKAT SEPULUH	0	0
Persentase Kesuksesan Rata-rata		**42**

Dari hasil pengujian perintah pada model tiga, didapatkan persentase yang cukup baik, yaitu 42%. Nilai yang didapatkan ini lebih baik daripada model-model sebelumnya dikarenakan jumlah kata dan kombinasi perintah yang lebih sedikit daripada model sebelumnya. Namun untuk membuat sebuah sistem dengan efisiensi dan kemampuan tinggi, persentase tersebut masih tidaklah cukup. Dari Tabel 3, dapat dilihat bahwa perintah dengan kata-kata yang menggunakan suara *nasal*, serta perintah *leveling* masih sukar dikenali oleh sistem. Karena itu pada model empat, perintah dengan kata-kata bersuara *nasal* dan perintah *leveling* akan ditiadakan.

4. Model Empat

Pada model empat, digunakan susunan model bahasa dan model akustik yang lebih sederhana dan lebih minimalis daripada model-model sebelumnya, yaitu dengan meniadakan perintah dengan suara *nasal*, dan perintah *leveling*. Jumlah perintah yang akan dikenali adalah 11 perintah, sedangkan jumlah *sample training* adalah sebanyak 26 *sample*. Hasil pengujian ditunjukkan pada Tabel 4.

Tabel 4. Hasil Pengujian Model Empat

Urutan Perintah (10 kali)	Dikenali (kali)	Kesuksesan (%)
BEJO	10	100
HIDUPKAN LAMPU	10	100
MATIKAN LAMPU	10	100
HIDUPKAN AC	10	100
MATIKAN AC	10	100
BUKA TIRAI	10	100
TUTUP TIRAI	10	100
LAMPU TAMBAH	10	100
LAMPU KURANG	10	100
TIRAI TAMBAH	10	100
TIRAI KURANG	10	100
Persentase Kesuksesan Rata-rata		**100**

Hasil pengujian perintah model empat menunjukkan persentase kesuksesan yang sempurna. Dengan kata lain, model *training* yang lebih sederhana dan efisien akan membuat kemampuan pengenalan perintah oleh sistem menjadi lebih baik. Model keempat inilah yang akan diimplementasikan ke dalam sistem.

Dari semua pengujian terhadap empat model di atas, dapat diambil kesimpulan bahwa jumlah kata, kombinasi perintah, serta *training* yang selektif akan menghasilkan model *training* yang lebih sederhana dan membuat sistem menjadi jauh lebih efisien dan lebih fungsional.

B. *Pengujian Pendeteksian Perintah dalam Kondisi Kebisingan yang Berbeda-beda*

Pengujian ini bertujuan untuk mengetahui kemampuan sistem mengenali perintah dengan kondisi kebisingan ruangan yang bermacam-macam, serta untuk mengetahui batas maksimal kebisingan ruangan agar sistem dapat mengenali perintah. Tingkat kebisingan ruangan dikondisikan dalam empat kisaran tingkat (*level*), dengan cara memainkan sebuah *track* lagu dan mengatur *volume playback*, dan akan diukur menggunakan *smartphone* Sony Z1 pada posisi mikrofon. Model yang digunakan dalam pengujian ini adalah model 4. Hasil pengujian dipresentasikan pada Tabel 5.

Tabel 5. Hasil Pengujian Tingkat Kebisingan

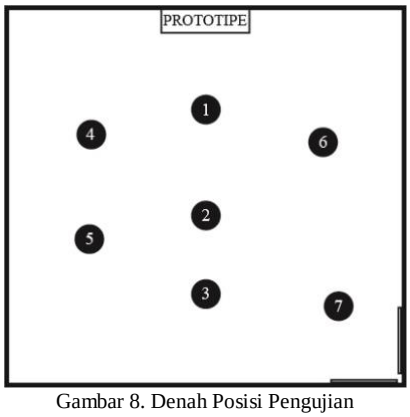
Perintah (10 kali)	Kesuksesan (%)			
	55-59 dB	60-64 dB	65-69 dB	70-74 dB
BEJO	100	100	100	0
HIDUPKAN LAMPU	100	90	80	0

MATIKAN LAMPU	100	70	70	0
HIDUPKAN AC	100	100	100	0
MATIKAN AC	100	100	100	0
BUKA TIRAI	100	100	80	0
TUTUP TIRAI	100	100	100	0
LAMPU TAMBAH	100	90	80	0
LAMPU KURANG	100	80	60	0
TIRAI TAMBAH	100	80	70	0
TIRAI KURANG	100	70	70	0
Rata-rata	**100**	**89.0909**	**82.7273**	**0**

Dari Tabel 5, dapat dilihat bahwa semakin tinggi tingkat kebisingan ruangan saat pengguna memberikan perintah, maka kemampuan sistem dalam menangkap perintah akan semakin turun. Selain itu, sistem sama sekali tidak dapat mengenali perintah jika sistem menangkap tingkat kebisingan lebih dari 69 dB. Dengan kata lain, jika sistem menangkap suara apapun lebih dari 69 dB maka ia akan menganggap suara tersebut sebagai perintah. Sehingga jika 69 dB yang ditangkap adalah kebisingan, maka perintah yang diberikan pengguna tidak akan dikenali sama sekali.

C. *Pengujian Jarak dan Posisi Pengguna*

Pengujian ini dilakukan untuk mengetahui pengaruh jarak dan posisi pengguna saat memberikan perintah. Model yang digunakan untuk pengujian adalah model 4. Gambar 8 menunjukkan denah serta posisi pengguna yang akan diuji, dan hasil pengujian ditunjukkan pada Tabel 6.



Gambar 8. Denah Posisi Pengujian

Tabel 6. Hasil Pengujian Jarak dan Posisi Pengguna

Urutan Perintah (10 kali)	Kesuksesan (%)			
	Posisi1 (1,5m)	Posisi2 (3,3m)	Posisi3 (4,8m)	Posisi4 (2,4m)
BEJO	100	100	100	100
HIDUPKAN LAMPU	100	100	90	100
MATIKAN LAMPU	100	90	90	100
HIDUPKAN AC	100	100	100	100
MATIKAN AC	100	100	100	100
BUKA TIRAI	100	100	100	100
TUTUP TIRAI	100	100	100	100
LAMPU TAMBAH	100	90	80	90
LAMPU KURANG	100	100	90	100
TIRAI TAMBAH	100	100	80	100
TIRAI KURANG	100	90	80	100
Rata-rata	**100**	**97.27**	**91.82**	**99.09**

Tabel 6. Hasil Pengujian Jarak dan Posisi Pengguna (sambungan)

Urutan Perintah (10 kali)	Kesuksesan (%)		
	Posisi5(4,1m)	Posisi6(2,5m)	Posisi7(6m)
BEJO	100	100	100
HIDUPKAN LAMPU	90	100	70
MATIKAN LAMPU	90	100	80
HIDUPKAN AC	100	100	100
MATIKAN AC	100	100	100
BUKA TIRAI	100	100	100
TUTUP TIRAI	100	100	100
LAMPU TAMBAH	80	80	80
LAMPU KURANG	80	90	70
TIRAI TAMBAH	90	90	80
TIRAI KURANG	90	100	60
Rata-rata	**92.73**	**96.36**	**85.45**

Hasil di atas menunjukkan bahwa semakin jauh jarak pengguna saat memberikan perintah, maka persentase pengenalan perintah akan berkurang. Hal ini disebabkan karena intensitas suara yang ditangkap sistem ikut berkurang. Namun selama pengguna mampu mengeluarkan suara sedemikian keras, dan mikrofon dapat menangkap intensitas suara minimal sebesar 70 dB, maka sistem akan menangkap suara tersebut sebagai perintah bagi dirinya.

D. *Pengujian Pengguna Baru Sebelum Melakukan Training*

Pengujian ini bertujuan untuk mengetahui kemampuan sistem dalam mengenali perintah yang diberikan oleh pengguna baru yang tidak dikenali oleh sistem. Pengujian dilakukan menggunakan model empat, tanpa adanya penambahan dari *training* pengguna baru. Berikut adalah hasil dari pengujian ini.

Tabel 7. Hasil Pengujian Pengguna Baru Sebelum Melakukan *Training*

Urutan Perintah (10 kali)	Dikenali (kali)	Kesuksesan (%)
BEJO	7	70
HIDUPKAN LAMPU	5	50
MATIKAN LAMPU	3	30
HIDUPKAN AC	6	60
MATIKAN AC	7	70
BUKA TIRAI	6	60
TUTUP TIRAI	7	70
LAMPU TAMBAH	3	30
LAMPU KURANG	4	40
TIRAI TAMBAH	2	20
TIRAI KURANG	4	40
Persentase Kesuksesan Rata-rata		**49.0909091**

Dari pengujian tersebut didapatkan persentase kesuksesan rata-rata sebesar 49,09%. Hal ini menunjukkan bahwa perintah yang diberikan oleh pengguna baru dapat dikenali oleh sistem, namun tidak sempurna. Karena itu, pada pengujian berikutnya akan dilakukan setelah pengguna baru melakukan *training.*

E. *Pengujian Pengguna Baru Sesudah Melakukan Training*

Sebelum pengujian, dilakukan pengubahan terhadap *sample training* terlebih dahulu, dengan menambahkan hasil *training* dari pengguna baru ke *training* pengguna lama (menggunakan model 4). Jumlah *sample training* yang ditambahkan adalah sebanyak 4 *sample*, sehingga jumlah total menjadi 30 *sample*. Berikut adalah hasil dari pengujian ini.

Tabel 8. Hasil Pengujian Pengguna Baru Sesudah Melakukan *Training*

Urutan Perintah (10 kali)	Dikenali (kali)	Kesuksesan (%)
BEJO	8	80
HIDUPKAN LAMPU	6	60
MATIKAN LAMPU	7	70
HIDUPKAN AC	8	80
MATIKAN AC	8	80
BUKA TIRAI	8	80
TUTUP TIRAI	7	70
LAMPU TAMBAH	4	40
LAMPU KURANG	6	60
TIRAI TAMBAH	4	40
TIRAI KURANG	4	40
Persentase Kesuksesan Rata-rata		**63.6363636**

Persentase kesuksesan rata-rata dari pengujian ini, yaitu sebesar 63,63%, menunjukkan adanya peningkatan dari hasil pengujian sebelumnya. Peningkatan ini menunjukkan bahwa *training* pengguna baru yang telah ditambahkan memberikan pengaruh terhadap kemampuan sistem dalam mengenali perintah dari pengguna tersebut, namun masih tidak sempurna. Hal tersebut bisa dikarenakan *training* pengguna baru yang kurang baik, atau adanya pengaruh dari *training* oleh pengguna lama. Pengujian berikutnya akan dilakukan untuk melihat

pengaruh penambahan *training* pengguna baru terhadap pengguna lama.

F. *Pengujian Pengguna Lama dengan Kombinasi Training Pengguna Lama dan Pengguna Baru*

Pengujian dilakukan dengan menggunakan model *training* dari pengujian sebelumnya, yaitu dengan *training* dari pengguna lama dan pengguna baru, dengan jumlah *sample* 30 buah. Berikut ini adalah hasil yang didapatkan dari pengujian ini.

Tabel 9. Hasil Pengujian Pengguna Lama dengan Kombinasi *Training* Pengguna Lama dan Pengguna Baru

Urutan Perintah (2 kali)	Dikenali (kali)	Kesuksesan (%)
BEJO	10	100
HIDUPKAN LAMPU	9	90
MATIKAN LAMPU	8	80
HIDUPKAN AC	10	100
MATIKAN AC	10	100
BUKA TIRAI	10	100
TUTUP TIRAI	8	80
LAMPU TAMBAH	7	70
LAMPU KURANG	8	80
TIRAI TAMBAH	9	90
TIRAI KURANG	9	90
Persentase Kesuksesan Rata-rata		**89.090909**

Dari hasil pada Tabel 9, terlihat adanya penurunan persentase kesuksesan bila dibandingkan dengan pengujian pengguna lama menggunakan model empat. Penurunan ini menunjukkan bahwa penambahan *training* pengguna baru akan memberikan dampak terhadap pengenalan perintah dari pengguna lama. Hal ini serupa dengan model akustik yang tersusun dengan jumlah *sample training* yang terlalu banyak, sehingga mengakibatkan pemilihan *training* menjadi kurang selektif. Dengan adanya kekurangan tersebut, maka sebaiknya pemilihan hasil *training* tetap dilakukan dengan selektif walaupun jumlah *sample training* menjadi lebih banyak, begitu juga pada sistem dengan pengguna lebih dari satu orang.

V. KESIMPULAN

Kesimpulan yang dapat diambil dari hasil pembuatan sistem kendali suara berbahasa Indonesia ini adalah sebagai berikut:

- Pemilihan kata yang digunakan pada model bahasa dan penjagaan kualitas *training* model akustik memberikan pengaruh yang sangat besar terhadap kemampuan sistem untuk mengenali perintah. Perubahan yang terjadi pada salah satu dari kedua model tersebut akan memberikan pengaruh yang besar terhadap hasil akhir.
- Pengguna yang tidak melakukan *training* dapat dikenali oleh sistem jika memiliki pelafalan yang sama dengan pengguna yang telah melakukan *training*. Penambahan *training* pengguna baru akan memperbaiki pengenalan perintah dari pengguna baru, namun akan mengurangi kemampuan pengenalan perintah dari pengguna lama.
- Sistem dapat mengenali perintah yang diberikan pengguna dengan batas maksimal intensitas kebisingan ruangan yang ditangkap oleh mikrofon adalah sebesar 69 dB.
- Sistem akan semakin sukar mengenali perintah jika jarak pengguna semakin jauh, atau intensitas kebisingan yang ditangkap semakin tinggi.

DAFTAR PUSTAKA

[1] Ingle, A. J., & Gawali, B. W. (2011). Status of wireless technologies used for designing home automation system - A Review. *IJACSA - International Journal of Advanced Computer Science and Applications*, *2*(7), 142–146. Retrieved from www.ijacsa.thesai.org

[2] Wollerton, M. (2014). Voice control comes to the forefront of the smart home. Retrieved May 25, 2015, from http://www.cnet.com/news/voice-control-roundup/

[3] Step 1 - Task grammar - voxforge.org. (n.d.-a). Retrieved January 16, 2016, from http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/tutorial/data-prep/step-1

[4] Step 1 - Task grammar - voxforge.org. (n.d.). Retrieved December 4, 2015, from http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/how-to/data-prep/step-1

[5] Run acoustic model creation script - voxforge.org. (n.d.). Retrieved December 4, 2015, from http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/how-to/script