RANCANG BANGUN ALAT PENDETEKSI SUARA PANGGILAN MANUSIA BERBAHASA INDONESIA UNTUK TUNARUNGU MENGGUNAKAN *LIBRARY* POCKETSPHINX BERBASIS *EMBEDDED SYSTEM*

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Mukhammad Wildan Alauddin
NIM: 125150300111049

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

Rancang Bangun Alat Pendeteksi Suara Panggilan Manusia Berbahasa Indonesia Untuk Tunarungu Menggunakan *Library* Pocketsphinx Berbasis *Embedded System*

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Mukhammad Wildan Alauddin
NIM: 125150300111049

Skripsi ini telah diuji dan dinyatakan lulus pada
12 Agustus 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Wijaya Kurniawan, S.T, M.T	Budi Darma Setiawan, S.Kom, M.Cs
NIK: 19820125 201504 1 002	NIP: 19841015 201404 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 12 Agustus 2016

Mukhammad Wildan Alauddin
NIM: 125150300111049

KATA PENGANTAR

Puji syukur penulis panjatkan kahadirat Tuhan Yang Maha Esa, karena Rahmat-Nya lah penulis dapat menyelesaikan skripsi ini. Adapun maksud penyusunan skripsi ini adalah untuk memenuhi salah satu persyaratan dalam menempuh ujian Sarjana Komputer pada Fakultas Ilmu Komputer. Judul skripsi yang disusun adalah:"Rancang Bangun Alat Pendeteksi Suara Panggilan Manusia Berbahasa Indonesia Untuk Tunarungu Menggunakan *Library* Pocketsphinx Berbasis *Embedded System*."

Banyak kesulitan dan hambatan yang dialami oleh penulis dalam menyusun skripsi ini, tetapi semua itu telah dapat diatasi dengan baik berkat dukungan dan bantuan dari berbagi pihak. Untuk itulah pada kesempatan ini penulis menyampaikan rasa hormat dan terima kasih kepada:

1. Bapak Abdul Madjid dan ibu Sholihah, selaku kedua orang tua kandung penulis yang selalu mendoakan dan membina penulis sejak lahir hingga masa yang akan datang serta mbak Majida Noviyanti dan seluruh keluarga besar Bani Wiro'i dan Bani Syafi'i yang selalu mendoakan penulis.
2. Bapak Wijaya Kurniawan, S.T, M.T. selaku dosen pembimbing I skripsi dan bapak Budi Darma Setiawan, S.Kom, M.Cs. selaku dosen pembimbing II skripsi yang selalu memberi bimbingan dan arahan sehingga penulis dapat menyelesaikan skripsi ini.
3. Bapak Sabriansyah Rizqika Akbar selaku Ketua Program Studi Teknik Komputer yang selalu menjadi teladan dan memberikan motivasi selama masa studi S1 ini.
4. Seluruh rekan mahasiswa Program Studi Teknik Komputer angkatan 2011 hingga 2014 yang selalu memberikan motivasi, dukungan, doa serta bantuan sampai terselesaikannya skripsi ini.
5. Seluruh rekan kerja LKMTIIK yang selalu mengingatkan penulis agar tidak melupakan kegiatan akademik tanpa mengesampingkan kegiatan kelembagaan.
6. Syahroni, Husni, Iman, Kiky, Fathi, Farid, Syafiq, Ali, Mafrizal, Arif dan teman-teman Kos Pak Hari yang selalu memberikan bantuan moral dan material kepada penulis.

Penulis menyadari bahwa proposal skripsi ini jauh dari sempurna, oleh karena itu untuk segala kritik dan saran yang membangun penulis ucapkan terima kasih. Penulis mengharapkan semoga skripsi ini dapat bermanfaat bagi yang membutuhkan dan menggunakannya.

Malang, 12 Agustus 2016

Penulis
mesutwildan@gmail.com

ABSTRAK

Komunikasi adalah salah satu hal terpenting dalam kehidupan manusia. Namun, tidak semua manusia dapat menggunakan indera pendengarannya secara sempurna. Manusia yang mengalami disabilitas indera pendengaran disebut sebagai tunarungu. Disabilitas tunarungu dapat disebabkan oleh banyak hal antara lain faktor keturunan dan kecelakaan.

Penyembuhan indera pendengaran bukanlah satu-satunya hal yang efektif sebagai solusi atas permasalahan ini. Hal ini disebabkan oleh tidak semua disabilitas pendengaran dapat disembuhkan, terutama akibat faktor keturunan. Selain itu, biaya untuk penyembuhan sangat mahal sehingga tidak semua penyandang tunarungu dapat melakukannya.

Atas beberapa pertimbangan tersebut penulis melakukan sebuah penelitian memanfaatkan teknologi untuk mengatasi permasalahan komunikasi antara manusia normal dengan para penyandang disabilitas tunarungu. Penulis mengaplikasikan sebuah *mini PC* bertipe Raspberry Pi 2 B+ sebagai inti dari sistem yang akan dirancang memanfaatkan *library* PocketSphinx sebagai perangkat lunak utama dalam proses konversi ucapan menjadi tulisan (*speech to text*) untuk selanjutnya diseleksi agar sistem hanya merespon apabila *input* panggilan sesuai dengan nama pengguna sistem. Sistem tersebut akan dirancang dengan menggunakan *mini* mikrofon sebagai sensor utama dan memanfaatkan *library* PocketSphinx yang menggunakan metode *Hidden Markov Model* (HMM) untuk diterapkan oleh sistem dalam mengenali karakteristik pengucapan manusia.

Alur kerja sistem ini diawali oleh *training* terhadap sistem agar mampu mengenali berbagai bola ucapan untuk suatu nama tertentu. Hasil akhir dari proses *training* ini berupa *acoustic model* yang merupakan hasil pencocokan data suara pada saat perekaman (*audio file*) dengan input pelafalan (*transcription file*). Selain itu juga perlu adanya *file dictionary* untuk mendata daftar kata yang nantinya akan dikenali oleh sistem dan juga *language model* untuk mengenali ciri ucapan seseorang. Apabila nama panggilan ketika proses *testing* sesuai dengan nama panggilan pada saat *training*, maka sistem merespon dengan memberikan getaran melalui *vibrator* dan indikator LED yang menyala.

Pada penelitian kali ini, peneliti dapat menilai bahwa penggunaan *mini PC* dapat memungkinkan sistem dapat digunakan secara *mobile*. Namun diperlukan *mini PC* dengan spesifikasi lebih tinggi untuk meningkatkan performa sistem dan mempercepat eksekusi program dan *library yang* digunakan. Hal ini juga berlaku untuk penggunaan *mini* mikrofon agar sistem dapat mendeteksi sumber suara yang lebih jauh dan daya tangkap yang lebih tinggi.

Kata Kunci: Tunarungu, Raspberry, PocksetSphinx, *Hidden Markov Model*

ABSTRACT

Communication is one of the most important things in human life. However, not all people can use their sense of hearing perfectly. People who have auditory disability are called deaf. Disability hearing impairment can be caused by many things, among others heredity and accidents.

Healing sense of hearing is not the only thing that is effective as a solution to this problem. This is due to not all auditory disabilities can be cured, especially due to hereditary factors. In addition, the cost for recovery is very expensive and not all deaf can do.

On some of these considerations the author conducted a research using technology to overcome the problems of communication between normal humans with the deaf. The author applied a mini-PC with the type of Raspberry Pi 2 B+ as the core of the system to be designed utilizing the PocketSphinx library as the primary software for the conversion of speech into writing (speech to text) to be selected by system so the syste only give respond while the name that called by people the same as the name which called while training session.

The system will be designed by using a mini mikrofon as the primary sensor and library PocketSphinx that implemented method of Hidden Markov Model (HMM) to be applied by the system to recognize the characteristics of human speech. If the name which called at the testing process in accordance with the name at training, the system responds by giving vibration through the vibrator and LED indicator lights up.

In this research, the author were able to assess that the use of the mini PC can enable the system to be used in mobile. However, it needed mini PC with higher specifications to improve system performance and speed up execution of programs and libraries used. This also applies to the use of mini mikrofon for the system to detect the sound source further and catch higher power.

Keywords: *Deaf, Raspberry, PocksetSphinx, Hidden Markov Model*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GRAFIK

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Pendengaran merupakan alat sensoris utama untuk berbicara dan berbahasa. Kehilangan pendengaran sejak lahir atau akibat unsur kecelakaan menyebabkan seseorang kesulitan dalam berbicara dan berkomunikasi dengan orang lain secara lisan. Kehilangan pendengaran pada seseorang akan berpengaruh pada perkembangan fungsi kognitifnya, karena penderita tunarungu mengalami kesulitan dalam memahami informasi yang bersifat verbal terutama konsep-konsep yang bersifat abstrak yang memerlukan penjelasan.

Melalui situs resminya (http://www.who.int/pbd/deafness/estimates/en/), Organisasi Kesehatan Dunia atau *World Health Organization* (WHO) merilis data hasil survei pada tahun 2012 tentang jumlah kasus disabilitas gangguan pendengaran. Survei tersebut didasarkan pada review dari 42 studi kasus berbasis sampel populasi yang dilakukan hingga 2010. Data hasil survei tersebut menyatakan bahwa ada 360 juta orang di dunia dengan disabilitas gangguan pendengaran (5,3% dari populasi dunia), 328 juta (91%) di antaranya adalah orang dewasa (183 juta laki-laki, 145 juta perempuan) 32 (9%) juta tersebut adalah anak-anak.

Menurut hasil Survei Sosial Ekonomi Nasional (Susenas) yang dilaksanakan Biro Pusat Statistik (BPS) tahun 2012, jumlah penyandang disabilitas di Indonesia sebanyak 6.008.661 orang. Dari jumlah tersebut sekitar 1.780.200 orang adalah penyandang disabilitas netra, 472.855 orang penyandang disabilitas rungu wicara,402.817 orang penyandang disabilitas grahita/intelektual, 616.387 orang penyandang disabilitas tubuh, 170.120 orang penyandang disabilitas yang sulit mengurus diri sendiri, dan sekitar 2.401.592 orang mengalami disabilitas ganda.

Saat ini jumlah permasalahan yang timbul akibat kekurangpedulian masyarakat semakin banyak. Salah satu contohnya ketika kita mngetahui jumlah sukarelawan pendamping tunarungu yang sedikit dan minimnya masyarakat Indonesia yang mampu memahami bahasa isyarat. Hal ini semakin meminimalisir peluang komunikasi antara manusia yang normal dengan manusia berkebutuhan khusus atau biasa kita sebut dengan penyandang disabilitas, khususnya penyandang tunarungu.

Mahalnya biaya terapi dan penyembuhan membuat para pengembang teknologi merancang suatu sistem alternatif untuk membantu tunarungu. Sistem tersebut dapat mengubah bentuk suatu *input* yang berupa ucapan menjadi tulisan maupun sebaliknya. Salah satu metode yang digunakan untuk mengubah bentuk *input* tersebut adalah *Hidden Markov Model* (HMM). *Hidden Markov Model* (HMM) merupakan model statistik di mana suatu sistem yang dimodelkan diasumsikan sebagai *markov process* dengan kondisi yang tidak terobservasi. Suatu HMM dapat dianggap sebagai jaringan Bayesian dinamis yang sederhana. (Prasetyo, 2010).

Pengembangan teknologi yang mengimplementasikan *Hidden Markov Model* sudah sering kita jumpai di kehidupan sehari-hari, khususnya yang memanfaatkan salah satu dari *library*-nya, yaitu *library* Pocketsphinx. *Library* ini mendukung implementasi fitur *speech recognition* guna mendeteksi suara panggilan manusia kepada pengguna sistem ini. Untuk mengoptimalkan kinerja *library* ini, penulis berencana menerapkannya pada Raspberry Pi dengan bantuan Sistem Operasi Raspbian yang digunakan dalam mikrokomputer Raspberry Pi.

Berdasarkan uraian permasalahan yang telah dipaparkan di atas, dan dengan perkembangan teknologi yang ada serta ketersediaan perangkat keras untuk mengimplementasikan teknologi tersebut, diharapkan bisa menjadi solusi dari masalah yang telah dipaparkan pada paragraf sebelumnya. Oleh karena itu disini penulis mengangkat sebuah skripsi dengan judul **"Rancang Bangun Alat Pendeteksi Suara Panggilan Manusia Berbahasa Indonesia Untuk Tunarungu Menggunakan *Library* Pocketsphinx Berbasis *Embedded System*"**. Sistem ini dirancang berdasarkan pengukuran dan analisa untuk mengetahui kelayakannya serta dapat memberikan acuan untuk penerapan dalam studi kasus sehingga fitur *speech recognition* sebagai pendeteksi suara panggilan manusia berbahasa Indonesia ini dapat diimplementasikan dengan baik.

1.2 Rumusan Masalah

1. Bagaimana cara membuat sistem yang dapat mengolah suara untuk membantu penyandang disabilitas tunarungu mengetahui bahwa ada orang lain yang memanggilnya?
2. Bagaimana kondisi mendukung agar sistem ini dapat bekerja secara maksimal?

1.3 Tujuan

1. Membuat purwarupa alat untuk membantu penyandang disabilitas tunarungu mengetahui bahwa ada orang yang memanggilnya.
2. Mengetahui kondisi yang mendukung sistem ini agar berkerja secara maksimal.

1.4 Manfaat

a. Bagi Penulis

   1. Menambah pengalaman selama pengimplementasian sistem berdasarkan materi yang telah dipelajari selama perkuliahan.
   2. Memberikan analisa permasalahan terhadap sistem-sistem serupa yang pernah dirancang sebelumnya.

b. Bagi Pengguna

1. Memperoleh informasi yang dapat diterima oleh reseptor alat indera selain pendengaran.
2. Memudahkan pengguna sistem untuk menerima respon suara dari orang lain yang memanggilnya.

1.5 Batasan Masalah

Pada penelitian ini, penulis membatasi penelitiannya dalam beberapa hal.

1. Jenis suara yang nantinya akan dibaca oleh sensor mikrofon hanya untuk suara panggilan terhadap nama penyandang disabilitas pengguna alat ini.
2. Sensor mikrofon hanya akan mendeteksi suara audiosonik dengan *samprate* 8 *kHz*, Lowerf (1 Hz, 150 Hz) dan Upperf (3500 Hz, 4000 Hz).
3. Jenis sensor mikrofon yang kompatibel dengan Raspberry Pi 2.
4. Jarak minimal dan maksimal antara sumber suara dengan pengguna sistem yang bisa dideteksi oleh sensor mikrofon.
5. Tipe HMM yang diimplementasikan pada sistem ini adalah Gaussian HMM (HMM Standar).

1.6 Sistematika Pembahasan

Penyusunan skripsi ini menggunakan kerangka pembahasan yang ditulis secara berurutan sebagai berikut:

BAB 1 PENDAHULUAN

Bab ini merupakan dasar dari penyusunan skripsi ini yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

BAB 2 TINJUAN PUSTAKA DAN DASAR TEORI

Bab ini berisi kajian pustaka, referensi mengenai perangkat-perangkat yang digunakan dalam mendukung pembuatan aplikasi atau sumber-sumber yang berhubungan dengan permasalahan dalam skripsi yang meliputi: Penggunaan *voice recognition* menggunakan *library* PocketSphinx, metode *Hidden Markov Model* serta Raspberry Pi.

BAB 3 METODOLOGI PENELITIAN

Bab ini menjelaskan bagaimana metodologi penelitian, memberikan gambaran umum sistem, perancangan untuk melakukan *training* suara dan *testing* suara pada *library* PocketSphinx dengan menggunakan Bahasa Indonesia.

BAB 4 PERANCANGAN

Bab ini menjelaskan secara rinci proses perancangan sistem yang meliputi deskripsi umum dari sistem, pemenuhan kebutuhan sistem, dan batasan implementasi sistem.

BAB 5 IMPLEMENTASI

Bab ini menjelaskan implementasi sistem penelitian berupa hasil *testing* dan *training*, serta batasan implementasi sistem.

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini membahas hasil uji coba dari sistem yang dibuat dengan melihat kualitas proses yang dilakukan oleh sistem dan evaluasi untuk mengetahui kekurangan dari sistem yang telah selesai dirancang.

BAB 7 PENUTUP

Bab ini digunakan untuk menyampaikan kesimpulan dari percobaan dan analisis yang dilakukan serta saran untuk pengembangan sistem pada penelitian yang serupa untuk peneliti selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini membahas kajian pustaka dan dasar teori yang digunakan untuk menunjang penulisan skripsi "Rancang Bangun Alat Bantu Pendeteksi Suara Panggilan Manusia Berbahasa Indonesia Untuk Tunarungu Berbasis *Embedded System*."

2.1 Tinjauan Pustaka

Penelitian yang dilakukan oleh (Mokhammad Hilman Fatah, 2015) yang berjudul *Implementasi Library Pocketsphinx Untuk Pengenalan Voice Command Berbahasa Indonesia Secara Offline.* Pada penelitian tersebut telah dilakukan implementasi *library* PocketSphinx untuk merealisasikan sebuah sistem *voice command* yang nantinya akan diterapkan pada *smarthome*. Sistem tersebut dapat dijalankan dalam keadaan *offline* sehingga pengguna lebih mudah menggunakannya jika dibandingkan dengan sistem yang harus dioperasikan secara *online*. Selain itu, sistem ini juga sudah mampu mendeteksi fonem Indonesia sebagai input dari pemberi *voice command*. Pada penelitian ini juga telah dilakukan pengubahan nilai dari beberapa parameter yang digunakan untuk menemukan kisaran batas atas dan batas bawah frekuensi sinyal suara terbaik dimana informasi paling banyak dijumpai. Namun hal ini diimplementasikan pada PC sehingga belum dapat langsung diterapkan pada kondisi *smarthome* yang sesungguhnya.

Penelitian serupa juga dilakukan oleh (Febriawan Ariful Fikri, 2016) dengan judul *Metode Query Matching Untuk Sistem Penjawab Pertanyaan Dengan Speech Recognition Memanfaatkan Library Pocketsphinx*. Dalam penelitian tersebut telah dilakukan implementasi *library* PocketSphinx untuk merealisasikan sistem penjawab pertanyaan untuk memudahkan mahasiswa dalam menemukan ruangan atau bagian tertentu dari Fakultas Ilmu Komputer Universitas Brawijaya, misalnya ruang dosen. Hasil analisa dari penelitian yang dilakukan oleh Mokhammad Hilman Fatah sebelumnya, Febriawan Ariful Fikri menyepurnakan metode *Hidden Markov Model* (HMM) yang hanya dapat mengonversi ucapan menjadi tulisan (*speech to text*) agar dapat melakukan interaksi dengan memberikan *feedback* berupa jawaban atas pertanyaan yang telah diberikan. Penyemurnaan tersebut dilakukan dengan mengombinasikan hasil konversi *speech to text* dengan metode *query matching* sebagai penentu respon yang paling tepat atas setiap pertanyakan yang diajukan sebagai input sistem.

2.2 Pengenalan Ucapan

Pengenalan suara merupakan kemampuan sebuah alat/komputer dalam menerjemahkan suara menjadi bentuk teks ataupun perintah. Teknologi ini mempunyai fasilitas untuk mengubah *input* suara kedalam rangkaian kosakata dalam bentuk sinyal-sinyal suara lalu menghasilkan probabilitas model statistik untuk menemukan persamaan terhadap *input* suaranya (Marietha, dkk, 2012).

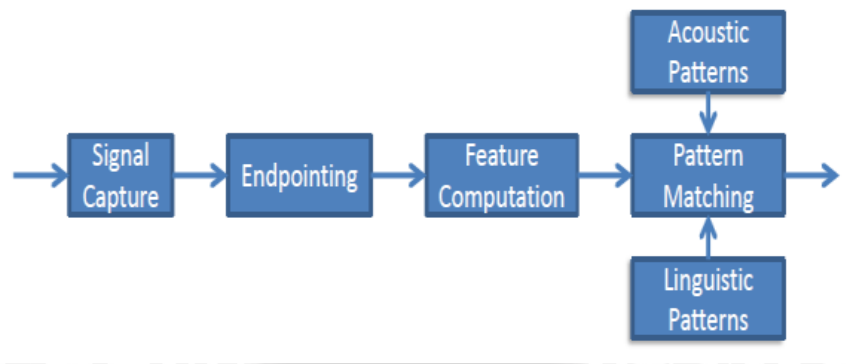
Secara umum pengenalan suara dibagi 2 jenis yaitu pengenalan pembicara dan pengenalan ucapan. Pengenalan pembicara merupakan proses untuk mengenali siapa yang sedang berbicara atau mengucapkan informasi berdasarkan pola suara yang dimasukkan. Teknologi ini sangat ampuh untuk mengidentifikasi suara seseorang dan bermanfaat untuk digunakan pada layanan seperti *control smarthome*, keamanan, layanan informasi dan akses terhadap keadaan tertentu. Jenis yang kedua yaitu, pengenalan ucapan yang didefinisikan sebagai proses pengubahan sinyal suara menjadi dalam bentuk teks. Pengenalan ucapan ini memiliki kemampuan untuk mencocokkan suara terhadap data yang ada seperti rekaman ataupun pembendaharaan kata (Ahsan, K.M.T, 2011).

Pengenalan ucapan bekerja untuk menerjemahkan ucapan seseorang berdasarkan data suara yang sudah dimiliki oleh aplikasi dikarenakan teknologi ini dapat bekerja apabila sudah mengenali pola suara seseorang, makin banyak data suara yang yang dimiliki, makin bagus juga teknologi ini dalam mengenali suara seseorang.

Hasil dari identifikasi kata yang diucapkan dapat ditampilkan dalam bentuk kosakata maupun dapat dibaca oleh perangkat teknologi sebagai sebuah perintah untuk melakukan suatu pekerjaan, misalnya penekanan tombol pada kontrol *smarthome system* yang dilakukan secara otomatis dengan perintah suara (*voice command*).

2.3 Sistematika Pengenalan Ucapan

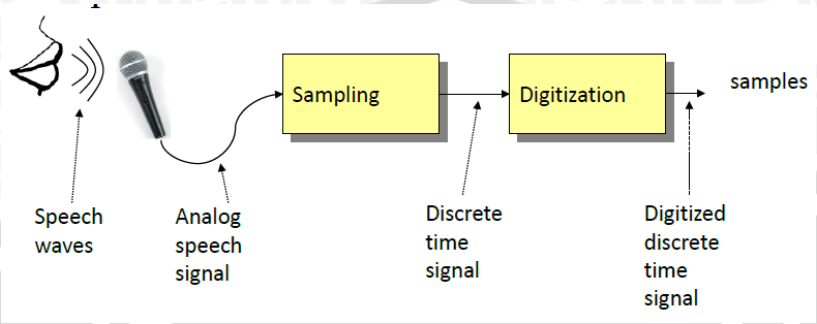
Sistem pengenalan ucapan (*voice recognition*) adalah sebuah rangkaian teknik yang memungkinkan sebuah sistem komputer untuk menerima input berupa suara lisan untuk kemudian diubah menjadi bahasa tulisan (*speech to text*). Selanjutnya sistem akan mengidentifikasikan kata atau kalimat yang diucapkan dan menghasilkan teks yang sesuai dengan apa yang diucapkan. Pada gambar 2.1 merupakan alur system pengenalan ucapan.



Gambar 2.1 Alur Sistem Pengenalan Suara
Sumber: I Kadek Suryadharma (2014)

2.3.1 *Speech Signal Capture* (Penangkapan Sinyal Ucapan)

Tahap ini merupakan langkah pertama dalam *speech recognition*. Suara dihasilkan oleh saluran vokal manusia berupa serangkaian gelombang yang mampu didengar oleh telinga manusia. Secara umum dapat dilihat pada Gambar 2.2 berikut:



Gambar 2.2 Langkah Awal Penangkapan Suara
Sumber: I Kadek Suryadharma (2014)

Sinyal suara yang diterima berupa sinyal analog. *Sampling* dilakukan untuk mencuplik sinyal analog menjadi bit-bit sinyal analog diskrit yang nantinya memudahkan dalam pemrosesan dan hasilnya berupa sampel-sampel bilangan biner (sinyal digital) yang merupakan informasi dari sinyal asli.

2.3.2 *Endpointing*

Pada langkah ini digunakan untuk mengidentifikasi bagaimana hasil sinyal suara yang sudah di *capture* tadi dapat diproses. Misalnya *press and speak* (tekan dan bicara). Ini dilakukan agar dapat menghindari suara-suara yang tidak diingikan masuk ke sistem saat pengenalan suara.

2.3.3 *Feature Extraction*

Feature extraction (ekstraksi ciri) merupakan suatu pengambilan ciri / *feature* dari suatu sinyal informasi yang nantinya nilai yang didapatkan akan dianalisis untuk proses selanjutnya. Setiap informasi memiliki ciri yang berbeda (unik). Prinsip kerja ekstraksi ciri adalah dengan mengkonversi sinyal suara ke dalam beberapa parameter, dimana ada sebagian informasi tidak berguna yang dibuang tanpa menghilangkan arti sesungguhnya dari sinyal suara tersebut. Hasil keluaran dari ekstraksi ciri ini menjadi masukan pada proses pengenalan pola.

2.3.4 *Matching*

Matching atau pencocokan ini merupakan proses akhir pada *speech recognition*. Hasil dari ekstraksi ciri menjadi masukan pada proses pengenalan pola ini. Metode yang digunakan dalam pengenalan pola ialah metode *hidden markov model* (HMM). Pola yang didapat akan dicocokan dengan berbagai macam model. Ada 3 jenis model yang umum digunakan pada speech recognition yakni *Accoustic models*, *Pronunciation models* dan *Language models*.

2.4 *Hidden Markov Model*

Model Markov ditemukan oleh Andrey Markov dan merupakan bagian dari proses stokastik. Model ini memiliki kehandalan untuk memprediksi keadaan yang akan datang artinya kondisi saat ini menangkap semua informasi yang mempengaruhi evolusi dari suatu sistem dimasa depan.

Hidden Markov Model (HMM) adalah sebuah model statistik dari sebuah sistem yang melakukan perhitungan probabilitas dari suatu kejadian yang tidak dapat diamati berdasarkan kejadian yang dapat diamati (Jurafsky, 2000). Perhitungan probabilitas dilakukan dengan melihat kejadian-kejadian lain yang dapat diamati secara langsung.

Hidden Markov Model memiliki 2 macam bagian yaitu observed state dan hidden state. Observed state merupakan bagian yang dapat diamati secara langsong dan hidden state merupakan bagian yang tidak dapat diamati (Wibisono, Y. 2008).

Model ini merupakan dari *finite automation* yaitu beberapa kumpulan *state* yang transisi antar *state*-nya dilakukan berdasarkan masukan observasi. Pada proses *Markov,* setiap busur antar state berisi probabilitas yang mengindikasikan kemungkinan dari jalur tersebut. Jumlah probabilitas semua jalur yang keluar dari sebuah simpul memiliki nilai total satu.

Penerapan HMM pada pengenalan suara dgunakan untuk memprediksi suara yang diucapkan berdasarkan *dictionary file*, *language model*, dan *acoustic model*nya. HMM akan memberikan probabilitas berdasarkan kosakata yang akan diucapkan. Probabilitas kemudian dihitung berdasarkan pembobotannya sehingga mempunyai nilai terbaik yang memungkinkan kata tersebut dikeluarkan berdasarkan persamaan suara. Pencarian probabilitas terbaik dari masing-masing suku kata diselesaikan dengan Algoritma *Viterbi*.

Algoritma viterbi adalah metode *decoding* untuk mengkodekan kembali bit yang telah dikodekan oleh *convolutional code* dengan prinsip mencari kemungkinan bit yang paling mirip atau dapat disebut *maximum likelihood*. Proses *decoding* dapat disamakan dengan membandingkan deretan bit yang diterima dengan semua kemungkinan bit terkode, dari proses perbandingan tersebut akan dipilih bit yang paling mirip antara deretan bit yang diterima dengan kemungkinan deretan bit bit yang ada (Suryadharma, 2014).

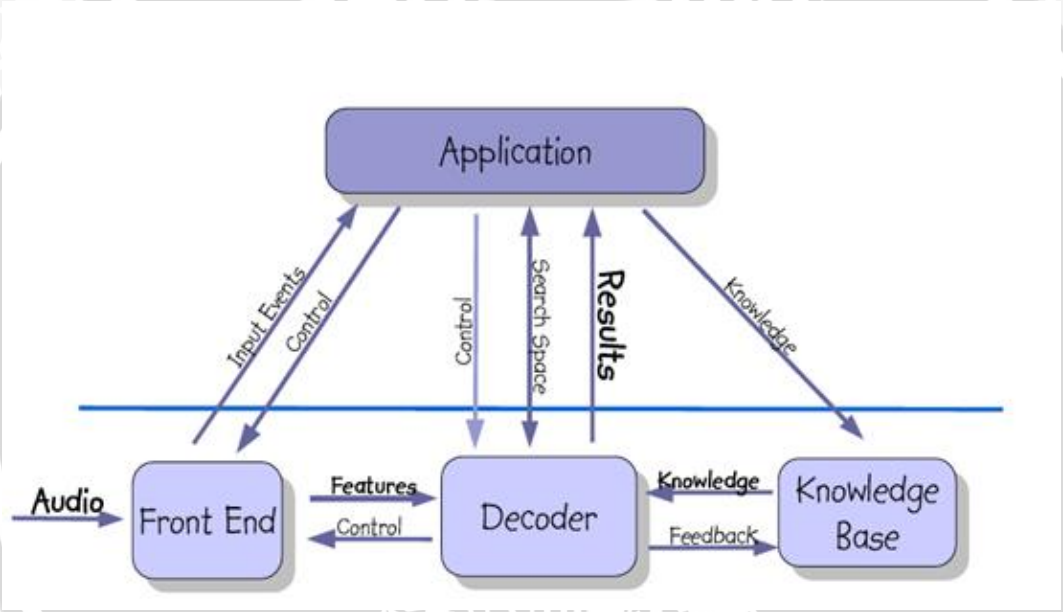
2.5 PocketSphinx

PocketSphinx ialah sistem pengenalan suara tercepat yang dikembangkan oleh Carnegie Mellon University (CMU). PocketSphinx menggunakan bahasa pemrograman C murni dan sangat optimal karena bersifat *real-time* sehingga mesin dapat mengenali suara dengan akurat. Dengan mengandalkan respon yang sangat cepat dan konsumsi sumber daya yang minim sehingga memungkinkan pengembangan pengenalan suara cocok digunakan untuk

aplikasi *desktop*, komando dan kontrol serta pendiktean. Selain itu, *engine* ini dapat berguna bagi perangkat tertanam dengan *fixed-point ariphmetics* yang berhasil digunakan pada iPhone, perangkat Nokia Maemon dan Windows Mobile (CMU, 2014). Komponen utama untuk membangun pocketsphinx ini ada 3 yakni PocketSphinx, SphinxBase, dan SphinxTrain

Library ini menyediakan fasilitas untuk para penggunaanya untuk memodifikasi bahasa yang akan digunakan untuk mengenali suara seseorang. Aplikasi PocketSphinx merupakan sistem pengenalan suara yang biasa digunakan untuk aplikasi *desktop*. Pada saat ini, PocketSphinx biasa digunakan untuk kebutuhan *embedded system* dan robotika karena *library* didukung memiliki kelebihan komputasi yang cepat dan ringan dalam mengenali suara seseorang.

2.5.1 Arsitektur Sphinx



Gambar 2.3 Arsitektur Sphinx
Sumber: http://cmusphinx.sourceforge.net

Gambar 2.3 menunjukan arsitektur umum yang digunakan oleh *Sphinx*. Setiap elemen pada gambar ini dapat diganti sesuai dengan kebutuhan peneliti. Dengan demikian, kita dapat mengubah beberapa fitur tanpa mengubah sisi fungsionalitasnya secara umum.

Arsitektur umum aplikasi ini terdiri komponen umum untuk menunjang hasil yang presisi pada saat pengujian aplikasi. Kompenen tersebut terdiri dari *front end*, *decoder*, basis pengetahuan, dan aplikasi itu sendiri. *Front end* bertanggung jawab untuk mengumpulkan, memberikan keterangan dan pengolahan *input* data. Selain itu, *front end* juga menyediakan kemudahan untuk pengaksesan API *audio* yang merupakan fitur untuk merekam masukan pengguna ucapan dan memutar hasil rekaman tersebut. Pada *knowledge base* memberikan sebuah informasi *decoder* untuk melakukan tugasnya. Informasi tersebut mencakup model akustik dan model bahasa. Basis pengetahuan pula yang memberikan

umpan balik dari decoder sehingga memungkinkan basis pengetahuan dapat dimodifikasi secara dinamis. Sedangkan pada *decoder* sendiri berfungsi sebagai komponen utama pada aplikasi pengenalan suara. *Decoder* berfungsi untuk menentukan urutan yang paling mungkin dari kata-kata yang dapat diwakili oleh serangkaian fitur dan dilakukan secara dinamis selama proses *testing/decoding*. (CMU, 2014)

2.5.2 *Language Model*

Language Model merupakan pembangkit *grammar* kosakata pada aplikasi pengenalan suara. Kompleksitas *grammar* tergantung pada sistem yang akan dikembangkan. Dalam penelitian kali ini, *language model* dibangun berdasarkan statistik yang digunakan oleh *library* PocketSphinx. *Toolkit language model* dibuat berdasarkan pemodelan *uni-gram*, *bi-gram*, dan *tri-gram* dari bahasa yang akan dikenali. Penciptaan *language model* terdiri dari komputasi kata jumlah *uni-gram* yang kemudian diubah menjadi kosakata dengan frekuensinya sehingga nantinya dapat menghasilkan *bi-gram* dan *tri-gram* dari teks pelatihan dan akhirnya mengubah *n-gram* ke dalam format biner *language model* dan format ARPAbet. Pada penelitian kali ini, *language model* dibuat dengan cara menuliskan data artikel yang ingin dikenali oleh aplikasi *voice command* menggunakan notepad lalu mengunggahnya ke server CMU Sphinx.

N-gram adalah potongan N-karakter yang diambilkan dari suatu string. Untuk mendapatkan N-gram yang utuh ditempuh dengan menambahkan *blank* pada awal dan akhir string. Misalnya suatu string "TEXT" setelah ditambah aal dan akhir dengan "_" sebagai pengganti blank akan didapat N-gram sebagai berikut:

Unigram : T,E,X,T

Bigram : _T, TE, EX,XT, dan T

Trigram : _TE,TEX,EXT, XT_ dan T_ _

Quadgram : _TEX, TEXT, EXT_, EX_ _, X_ _ _

Dapat disimpulkan bahwa untuk string berukuran n akan dimiliki n unigram dan n+1 bigram, n+1 trigram, n+1 quadgram dan seterusnya. Penggunaan N-gram untuk matching kata memiliki keuntungan sehingga dapat diterapkan pada recovery pada input karakter ASCII yang terkena noise, interpretasi kode pos, information retrieval dan berbagai aplikasi dalam pemrosesan bahasa alami.

Keuntungan N-gram dalam matching string adalah berdasarkan karakteristik N-gram sebagai bagian dari suatu string, sehingga kesalahan pada sebagian string hanya akan berakibat perbedaan pada sebagian Ngram. Sebagai contoh jika N-gram dari dua string dibandingkan, kemudian kita menghitung cacah N-gram yang sama dari dua string tersebut maka akan didapatkan nilai similaritas atau kemiripan dua string tersebut yang bersifat resistan terhadap kesalahan tekstual.

Kemiripan antara kata JOKO dengan JOKI (ada perbedaan 1 huruf), dapat diukur derajat kesamaan dengan cara menghitung berapa buah N-gram yang diambil dari dua kata tersebut yang bernilai sama, yaitu:

JOKO: _J, JO, OK,KO,O_ , JOKI : _J, JO, OK,KI, I_ kesamaan :3

Sementara antara kata JOKO dengan JONI (ada perbedaan 2 huruf), nilai kesamaan adalah :

JOKO: _J, JO, OK,KO,O_ , JONI : _J, JO, ON,NI,I_ kesamaan : 2

Sehingga dapat disimpulkan bahwa kemiripan atau kesamaan antara JOKO-JOKI dari pada antara JOKO-JONI.

2.5.3 *Acoustic Model*

Acoustic model merupakan fasilitas yang diberikan oleh PocketSphinx untuk mengenali ciri suara pengguna. *Acoustic model* menggunakan *Hidden Markov Model* dan *Gaussian Mixture* sebagai metode dalam menentukan sinyal suara yang disesuaikan dengan fonem seseorang. *Hidden Markov Model* pada *acoustic model* merepresentasikan beberapa parameter berupa *hidden state* yang berupa *input* suara, *observed state* berupa fitur vektor *input* suara, dan probabilitas transisi yang berupa nilai probabilitas terbaik untuk pengucapan suara sehingga dapat membangkitkan ciri suara yang digunakan pada aplikasi.

Pelatihan *acoustic model* dilakukan dengan cara merekam suara yang disesuaikan dengan *transcription file* dari setiap kalimat yang diucapkan dalam rekaman suara. *Input acoustic model* berupa rekaman suara dengan format *.wav*. Setiap file berisikan satu kalimat dari satu pembicara. Selanjutnya rekaman suara tersebut disesuaikan dengan *transcription file.* Pedoman pada *transcription file* merujuk *dictionary file. Filler file* yang berisikan suara *non-speech* yang bertujuan memberikan jeda pada sebelum kalimat mulai dan berakhir diucapkan (CMU, 2014).

2.5.4 *Dictionary File*

Dictionary file berisikan kosakata yang digunakan dalam teks artikel dan cara-cara pengucapan tiap kosakata yang terdiri dari fonem-fonem yang dapat disesuaikan dengan model bahasa yang akan digunakan. Pada keadaan awal, teks artikel diupload ke server CMU Sphinx menggunakan *web browser* dengan alamat http://www.speech.cs.cmu.edu/tools/lmtool-new.html. Setelah file tersebut diunggah, maka akan dihasilkan *dictionary file* yang berisikan kosakata yang dapat diucapkan oleh pengucapan bahasa Inggris. Oleh karena itu, fonem-fonem harus disesuaikan dengan gaya pengucapan orang Indonesia (Noora, 2010).

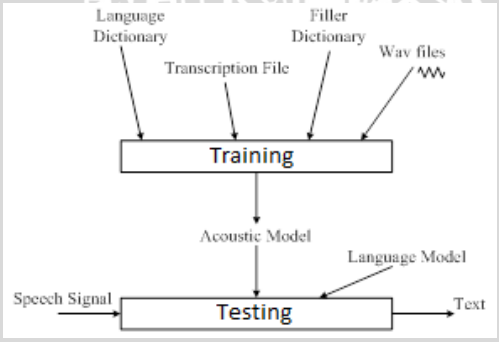
Dictionary file menjadi pedoman utama untuk pengenalan suara yang akan dirancang karena kosakata yang diucapkan terbatas sesuai dengan kosakata yang ada di d*ictionary file*. Penulisan kosakata pada d*ictionary file* menggunakan peraturan Arpabet. Arpabet adalah transkripsi fonetik yang dikembangkan oleh *Advanced Research Projects Agency* (ARPA) sebagai bagian dari *Speech Understanding Project* (1971-1976). Arpabet mewakili setiap fonem dari dalam bahasa Inggris Amerika umum (*General American English*) dengan urutan yang berbeda dari karakter ASCII (ICCCE, 2010).

2.5.5 *Training*

Training merupakan upaya pelatihan suara untuk membangkitkan kosakata yang akan diucapkan oleh seseorang. *Training* berfungsi sebagai pengenalan ciri suara manusia sehingga nantinya suara dapat dikenali dengan baik. *Training* akan mempelajari model unit suara menggunakan kumpulan sampel sinyal suara pada *database training* berdasarkan *transcription file*. Selain *transcription file*, dibutuhkan juga kamus pengucapan (*dictionary file*) yang memetakan setiap kata ke urutan unit suara. Hasil *training* menghasilkan akurasi rekaman suara yang telah dipetakan pada *database training*, hasil akurasi yang terbaik dapat didapatkan apabila nilai parameter-parameter pada saat *training* digunakan secara maksimal. Setelah proses *training* berakhir dengan menghasilkan hasil akurasi yang baik lalu peneliti dapat mengambil file *acoustic model* yang akan dapat digunakan untuk membantu *testing* pengenalan suara (Monika, 2012). Proses *training* suara menggunakan *library* dari CMU Sphinx yaitu SphinxTrain. *Training* membantu untuk membangkitkan probabilitas suatu kosakata, apabila kosakata sering dilatih maka akurasi kosakata yang diucapkan pada saat *testing* akan semakin baik.

2.5.6 *Testing*

Testing merupakan percobaan pengenalan suara dengan memberikan *executable* tunggal yang dapat melakukan tugas pengenalan suara saat diberi *input* suara. *File-file* yang dibutuhkan pada saat melakukan *testing* ialah *acoustic model*, *dictionary file,* dan *language model. Testing* dapat dilakukan dengan baik apabila proses pada saat *training* dilakukan dengan baik untuk mengenali pola suara orang Indonesia. Pada percobaan *testing* menggunakan *library* PocketSphinx (Monika, 2012). Hubungan antar proses *trainingi dan testing* diilustrasikan pada Gambar 2.4.



Gambar 2.4 Hubungan antar Proses *Training* dan *Testing*
Sumber: Chowdhury, 2010

2.6 Mikrokomputer Raspberry Pi 2B+

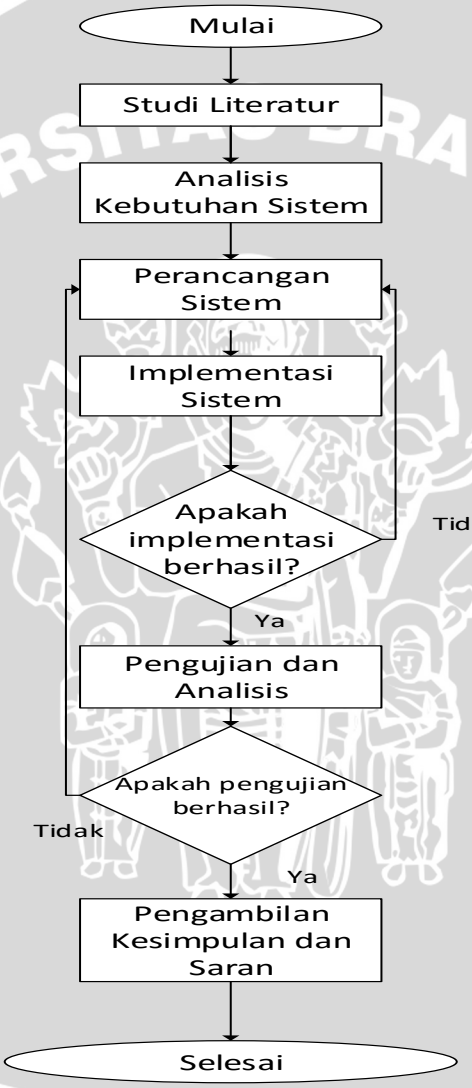
Salah satu mikrokomputer yang paling populer di pasaran adalah Raspberry Pi. Pada penelitian kali ini penulis akan memanfaatkan Raspberry Pi 2 untuk merealisasikan rancangan sistem yang telah dibuat sebelumnya. Memiliki bentuk dan ukuran yang sama dengan Raspberry Pi 1 B+, Raspberry Pi 2

dilengkapi dengan *Random Access Memory* (RAM) sebesar 1 GB disertai peningkatan prosesor dari BCM2835 (*single core* ARMv6) ke BCM2836 (*quad core* ARMv7) yang meningkatkan performa mikrokomputer ini menjadi hampir 2 kali lipat.

BAB 3 METODOLOGI

3.1 Alur Metode Penelitian

Bab ini menjelaskan langkah-langkah yang dilakukan dalam penyusunan skripsi, yaitu studi literatur, analisis kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian dan analisis serta pengambilan kesimpulan dan saran sebagai referensi untuk pengembangan sistem selanjutnya. Gambar 3.1 merupakan diagram alir tahap pengerjaan penelitian ini:



Gambar 3.1 Alur Metodologi Penelitian

Alur penelitian diawali dengan mencari pokok permasalahan yang akan diangkat menjadi sebuah penelitian, kemudian mencari literatur pendukung untuk bahan acuan melakukan penelitian. Proses selanjutnya yaitu menganalisis kebutuhan untuk melakukan perancangan dan implementasi sistem. Setelah sistem jadi, dilakukan penelitian untuk menguji apakah hasil sesuai dengan yang

diinginkan, jika tidak sesuai maka mengulangi proses perancangan sistem, dan jika hasil yang didapatkan sesuai maka didapat beberapa kesimpulan dari penelitian tersebut.

3.2 Studi Literatur

Dalam perancangan dan implementasi penelitian ini, perlu diadakan studi literatur. Studi literatur menjelaskan dasar teori yang digunakan sebagai penunjang dan pendukung penulisan skripsi. Teori penunjang dan pendukung tersebut didapat dari buku, jurnal, paper dan internet. Literatur yang digunakan meliputi:

a. Pengenalan Suara
b. *Hidden Markov Model*
c. PocketSphinx
d. Raspberry Pi

3.3 Analisis Kebutuhan Sistem

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem yang akan dibangun dan diuji. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan *(requirements)* sistem. Analisis juga dilakukan untuk mengetahui kondisi lapangan yang ada sehingga dapat diketahui implementasi perangkat lunak dan perangkat keras yang akan digunakan. Analisis kebutuhan ini didapat berdasarkan berbagai sumber di internet, buku-buku, dan jurnal-jurnal tentang definisi *speech recognition* untuk menunjang pembuatan sistem *voice command* berbahasa Indonesia.

Pada tahap perencanaan dan perancangan, sistem ini nantinya diharapkan mampu memenuhi kebutuhan dan dapat merealisasikan hal-hal berikut:

1. Sistem dapat digunakan dengan cara dipasang pada bagian tubuh pengguna.
2. Bagian sensor mikrofon mampu memiliki sensitivitas tinggi dan dapat mendeteksi sumber suara hingga jarak maksimal 100 meter dari pengguna.
3. Proses penerimaan data oleh sensor dan pemrosesan data hingga menghasilkan *output* dapat dilaksanakan kurang dari 1 detik.
4. Sistem dapat dirancang dengan bahasa pemrograman dan *library* yang kompatibel dengan *hardware*.

Berdasarkan analisis kebutuhan tersebut, penulis berencana merancang sistem ini dengan menggunakan perangkat-perangkat sebagai berikut:

1. *Powerbank* sebagai sumber daya listrik sistem akan digunakan dalam mobilitas tinggi dan tidak selalu dekat dengan sumber listrik saat digunakan.
2. Sensor mikrofon *omnidirectional* merupakan mikrofon yang mempunyai sensitivitas ke segala arah.
3. Mini PC berjenis Raspberry Pi 2 karena Raspberry Pi 2 dilengkapi dengan *Random Access Memory* (RAM) sebesar 1 GB disertai peningkatan prosesor dari BCM2835 (*single core* ARMv6) ke BCM2836 (*quad core* ARMv7) yang meningkatkan performa mini PC ini menjadi hampir 2 kali lipat dibandingkan Raspberry Pi 1 untuk mempercepat kinerja sistem yang akan dibangun.

4. Bahasa pemrograman C dan *library* PocketSphinx karena bahasa pemrograman ini cocok untuk perancangan sistem yang terkait dengan aplikasi yang berhubungan dengan mesin dan *library* ini sangat memudahkan peneliti untuk merancang aplikasi ini untuk pengenalan suara dengan berbagai bahasa secara fleksibel.

3.4 Perancangan

Sistem ini dirancang dengan menggabungkan rangkaian sensor mikrofon dan rangkaian *vibrator* dengan Mikrokomputer Raspberry Pi. Sistem menggunakan sumber daya tunggal yang berasal dari *Powerbank*. Pada mode *training*, sistem akan diaktifkan dalam mode *semi continuous* untuk mencuplik rekaman suara yang nantinya akan digunakan sebagai berkas *konowledge base*. Pada mode *testing*, sistem akan aktif dalam mode *continuous* untuk selalu aktif mendeteksi apabila ada suara manusia yang memanggil nama pengguna sistemnya.

3.5 Implementasi

Implementasi metode HMM pada sistem mengacu kepada perancangan analisis kebutuhan sistem. Implementasi perangkat lunak diawali dengan penjabaran spesifikasi lingkungan perancangan sistem. Implementasi metode HMM dilakukan dengan menggunakan bahasa pemrograman C dengan menggunakan *library* PocketSphinx. Pada tahap *training* hingga *testing* dilakukan pada sistem operasi Raspbian. Proses ini mungkin tidak berjalan satu kali, Jika dirasa perangkat keras dan perangkat lunak belum memenuhi kebutuhan sistem, maka sistem akan didesain ulang dalam dengan memberikan manipulasi terhadap metode HMM.

Implementasi di penelitian ini dibagi 2 tahap yaitu, proses *training* yang bertujuan untuk melatih suara orang Indonesia sehingga dapat mengenali pola suaranya dan proses *testing* yang bertujuan untuk mengetahui keakuratan sistem dalam mengenali input linguistik berbahasa Indonesia. Proses *training* memiliki persyaratan tingkat keakuratan sistem sehingga dapat mengenali pola suara pengucapan orang Indonesia dengan baik. Proses *training* dilakukan dengan cara merekam suara orang Indonesia yang disesuaikan dengan *transcription file* yang berupa kumpulan kalimat dari Bahasa Indonesia. *Training* dilakukan dengan menggunakan *library* SphinxTrain*. Training* dilakukan hingga memunculkan hasil terbaik dan dilakukan analisis sehingga dapat diukur dan dapat dituangkan ke dalam laporan skripsi ini. Setelah hasil pengujian *training* sesuai dengan persyaratan tingkat keakuratan sistem, dapat dilakukan analisis sehingga nantinya dapat digunakan pada saat proses *testing*. Pada saat proses *testing*, hasil proses *training* yang berupa *acoustic model* dijadikan file pendukung selain *dictionary file* dan *language model.* Implementasi terhadap *testing* dan *training* merupakan perintisan untuk melakukan pengujian sehingga diketahui hasil akurasi dari kedua proses tersebut dalam mengenali suara bahasa Indonesia.

Hardware yang digunakan selama pembuatan aplikasi pembelajaran bahasa Indonesia hanya digunakan 1 buah Raspberry Pi 2, sensor mikrofon, baterai dan beberapa komponen penunjang rangkaian. Selain itu juga diperlukan sistem operasi Rasbian pada Raspberry untuk mengoperasikan PocketSphinx.

3.6 Rencana Pengujian dan Analisis

Pengujian sistem akan dilakukan dengan mengubah nilai dari faktor-faktor yang mempengaruhi tingkat akurasi penangkapan sinyal suara oleh sensor mikrofon. Faktor-faktor tersebut antara lain jenis mikrofon, jarak antara koresponden dengan sensor mikrofon dan tingkat kerumitan nama. Tiga faktor tersebutlah yang diprediksi memiliki pengaruh paling besar terhadap daya tangkap sensor mikrofon.

3.7 Rencana Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan sistem, implementasi sistem, dan pengujian perangkat sistem telah diselesaikan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibangun. Kesimpulan diambil untuk mengetahui kesesuaian penerapan perangkat keras dan perangkat lunak penyusun sistem terhadap perancangan yang telah dibuat. Selanjutnya, peneliti melakukan perbaikan pada kesalahan-kesalahan yang terjadi pada pembuatan sistem ini. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan dan kekurangan yang terjadi dan menyempurnakan penulisan serta sebagai acuan untuk pengembangan sistem selanjutnya.

BAB 4 PERANCANGAN

4.1 Deskripsi Umum

Dalam bab ini menjelaskan persyaratan minimal yang harus dipenuhi untuk perancangan hingga implementasi. Dengan harapan perancangan dan implementasi bisa berjalan dengan baik.

4.1.1 Perspektif Sistem

Sistem ini dapat berjalan sesuai tujuan apabila sistem mampu melakukan deteksi nama pengguna sebagai hasil dari proses *training* dan *testing*. Sistem juga mampu membedakan antara suara orang lain yang memanggil pengguna sistem atau suara lain yang tidak memanggil nama *user*. Serta sistem dapat bekerja dengan input *switch* sebagai media interaksi manusia dengan sistem dan sistem dapat dijalankan secara *realtime*.

4.1.2 Kegunaan

Sistem ini berguna untuk membantu tunarungu agar dapat mengetahui ada seseorang yang sedang memanggilnya. Suara panggilan tersebut akan diubah menjadi *output* lain berupa getaran yang dapat diterima pengguna sistem.

4.1.3 Karateristik Pengguna

Karateristik pengguna pada sistem ini merespon *input* berupa suara panggilan nama pengguna sistem untuk diubah menjadi getaran yang dapat dirasakan oleh tunarungu.

4.1.4 Lingkungan Operasi

Sistem ini membutuhkan lingkungan yang dapat mendukung agar sistem dapat bekerja secara optimal. Hal ini dapat diindikasikan oleh hal-hal berikut:

1. Kondisi lingkungan yang sunyi adalah keadaan dimana sistem dapat bekerja dengan kemampuan maksimal. Semakin bising suatu lingkungan, semakin susah sistem mendeteksi adanya panggilan nama pengguna sistem.
2. Pada bagian tepi sensor mikrofon ditambahi corong agar sensor semakin akurat dalam menangkap suara panggilan nama pengguna sistem.

4.1.5 Batasan Perancangan dan Implementasi

Adapun batasan sistem sebagai berikut:

1. Sistem hanya dapat mendeteksi panggilan seseorang secara optimal dalam radius maksimal 2 meter.
2. Sistem hanya mendeteksi nama yang sudah diatur saat proses *training* dilakukan.
3. Tipe HMM yang diimplementasikan dalam sistem ini adalah Gaussian HMM (HMM standar).
4. Sistem hanya dapat diaktifkan oleh sumber listrik bertegangan 5 volt dan aliran listrik 2 ampere.

5. Sistem hanya dapat bekerja selama sumber daya listrik dari *powerbank* masih tersedia.
6. Proses *training* hanya dapat dilakukan oleh manusia non-disabilitas.

4.1.6 Asumsi dan Ketergatungan

Beberapa asumsi dan ketergantungan pada sistem ini yaitu:

1. Sistem hanya akan memberikan respon *output* yang benar hanya apabila sensor tepat mendeteksi nama pengguna saat proses *training*.
2. Semakin rumit ejaan nama seseorang, semakin kecil pula akurasi pembacaan sensor.
3. Sistem akan lebih mudah mendeteksi nama orang yang sudah sering melalui tahap *training* pada pengujian sebelumnya akibat sifat dari metode HMM.

4.2 Kebutuhan Antarmuka *Eksternal*

Bab ini menjelaskan semua kebutuhan dari sistem supaya sistem dapat bekerja sesuai dengan tujuan mulai dari kebutuhan media interaksi *user* dan antarmuka sistem, kebutuhan fungsional sistem, kebutuhan *eksternal* sistem, dan kebutuhan lainnya. Pada kebutuhan sistem tersebut meliputi aspek input dan output sistem, serta fungsi respon sistem terhadap *input* dan *output* dalam proses berjalannya sistem.

4.2.1 Kebutuhan Antarmuka Pengguna

Proses instalasi perangkat lunak yang diperlukan oleh sistem beserta konfigurasinya hanya dapat dilihat langsung oleh pengguna melalui monitor yang dihubungkan ke Raspberry dengan menggunakan kabel *Unshielded Twisted Pair* (UTP) bertipe *cross over*. Hasil instalasi dan konfigurasi pada akhirnya hanya akan membuat sistem dapat menjalankan instruksi dengan media komunikasi berupa *switch* untuk mengaktifkan mode *training* dan *testing* agar pengguna dapat mengoperasikan sistem dengan lebih mudah.

4.2.2 Kebutuhan Antarmuka Perangkat Keras

Beberapa kebutuhan perangkat keras pada sistem ini antara lain:

1. Raspberry Pi 2B+ sebagai mini PC agar sistem dapat digunakan secara *mobile*
2. Mikrofon mini yang kompatibel dengan Raspberry Pi 2 B+ untuk mendeteksi suara manusia.
3. *Soundcard* USB agar *audio jack* dapat terhubung ke *port* USB Raspberry.
4. *Powerbank* sebagai sumber listrik untuk mengaktifkan sistem.
5. Rangkaian kontrol untuk mengaktifkan sistem melalui *switch* dan *Light Emitting Diode* (LED) sebagai indikator diterimanya input suara maupun command yang terhubung ke *General-Purpose Input/Output* (GPIO) Raspberry.
6. *Casing* untuk Raspberry agar sistem lebih mudah dipasang di tubuh pengguna.
7. *Vibrator* untuk memberikan getaran sebagai respon *output* dari sistem.

4.2.3 Kebutuhan Antarmuka Perangkat Lunak

Ada beberapa kebutuhan perangkat lunak sistem ini, yaitu:

1. Putty, digunakan sebagai *output* visual proses yang dilakukan oleh Raspberry.
2. *Library* PocketSphinx untuk melakukan konversi ucapan ke tulisan (*speech to text*).
3. SphinxBase sebagai *Base Knowledge* untuk menyimpan hasil *training* yang pernah dilakukan sebelumnya.
4. SphinxTrain untuk melakukan proses *training* dan menghaslkan *transciption file*.

4.2.4 Kebutuhan Antarmuka Komunikasi

Pada sistem ini kebutuhan komunikasi hanya berlangsung pada saat ada suatu *input* berupa suara panggilan nama pengguna sistem yang sebelumnya sudah dideteksi pada sesi *training*. Suara panggilan nama user akan diproses oleh mikrofon dan diubah menjadi sinyal digital untuk kemudian dikomputasikan dan dilakukan pengecekan apakah *input* tersebut adalah input yang dapat diproses dan menghasilkan *output* berupa getaran dari *vibrator.*

4.3 Kebutuhan Fungsional

Kebutuhan fungsional merupakan jenis kebutuhan tentang proses yang nantinya dioperasikan pada sistem dan informasi yang dihasilkan oleh sistem.

4.3.1 Fungsi membedakan suara

Sistem dapat membedakan antara suara biasa dan suara manusia yang memanggil nama pengguna sistem.

4.4 Kebutuhan Performasi Sistem

Sistem ini mampu bekerja dengan performa maksimal hanya jika semua kebutuhan dipenuhi, seperti kebutuhan lingkungan sistem yang lebih optimal dalam kondisi yang lebih sunyi. Sistem juga hanya dapat membantu tunarungu secara efektif apabila proses eksekusi sistem dilakukan kurang dari 2 detik.

4.5 Spesifikasi Sistem

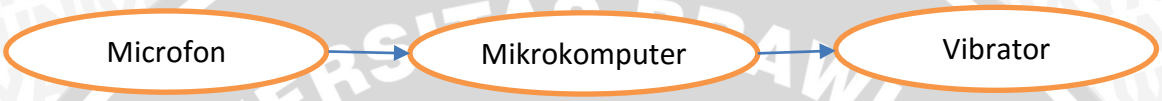
Spesifikasi alat secara global ditetapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya. Spesifikasi sistem yang direncanakan adalah sebagai berikut:

1. Mikrofon yang terhubung ke Raspberry melalui *souncard* USB agar *input* suara dapat segera diproses.
2. Untuk mengatur aktivasi sistem menggunakan tombol power yang ada pada *powerbank*.
3. *Software* dalam mengatur *input* sistem menggunakan bahasa C dan Python pada Raspberry.

4. Tombol *switch* yang dihubungkan ke GPIO untuk mengaktifkan mode *testing* dan *training*.
5. *Vibrator* yang akan memberikan *output* berupa getaran yang hanya memberikan respon tersebut apabila *input* nama sesuai dengan nama pengguna sistem.

4.6 Perancangan Sistem

Pada perancangan sistem dapat disimpulkan pada blok diagram ini, pada blok diagram ini akan menjelaskan tentang sistem kerja alat yang akan dibuat secara keseluruhan dan digunakan untuk mempermudah pembaca dalam memahami alur kerja sistem.

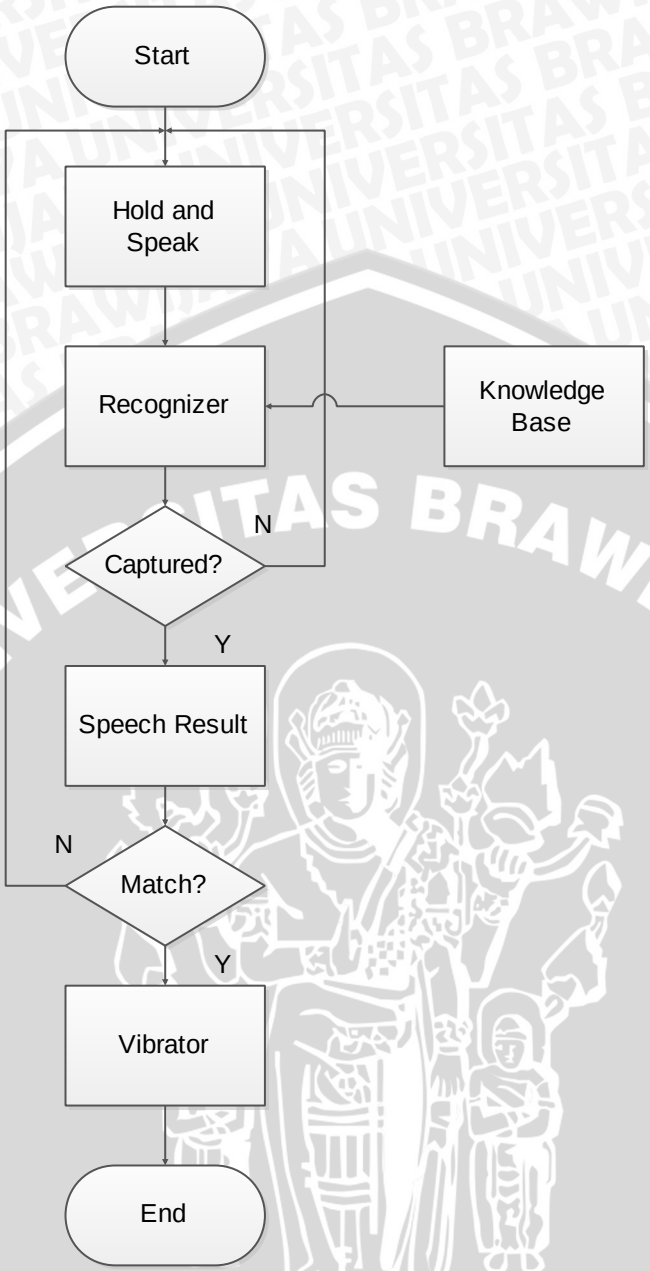


Gambar 4.1 Blok Diagram Sistem

Berdasarkan Gambar 4.1 perancangan sistem penelitian dapat dijelaskan sebagai berikut :

1. Sensor mikrofon digunakan untuk membaca suara yang diucapkan oleh manusia pada saat proses *training* dan *testing* dilakukan.
2. *Soundcard* USB digunakan agar mikrofon dapat dihubungkan ke USB sebagai input tanpa mengubah konfigurasi mikrofon.
3. Mini PC Raspberry merupakan bagian pemrosesan yang mengolah data dari sensor mikrofon apakah ucapan seseorang dapat dibaca sensor dengan baik yang nantinya menentukan data akan dilanjutkan ke proses berikutnya atau tidak.
4. *Vibrator* digunakan untuk memberi *output* pada pengguna sistem apabila *input* yang dibaca oleh sensor sesuai kriteria sistem.
5. *Light Emitting Diode* digunakan sebagai indikator keberhasilan pembacaan *input* atau keberlangsungan proses yang dijalankan oleh sistem.

Secara umum, gambaran diagram alir sistem dapat digambarkan pada Gambar 4.2.

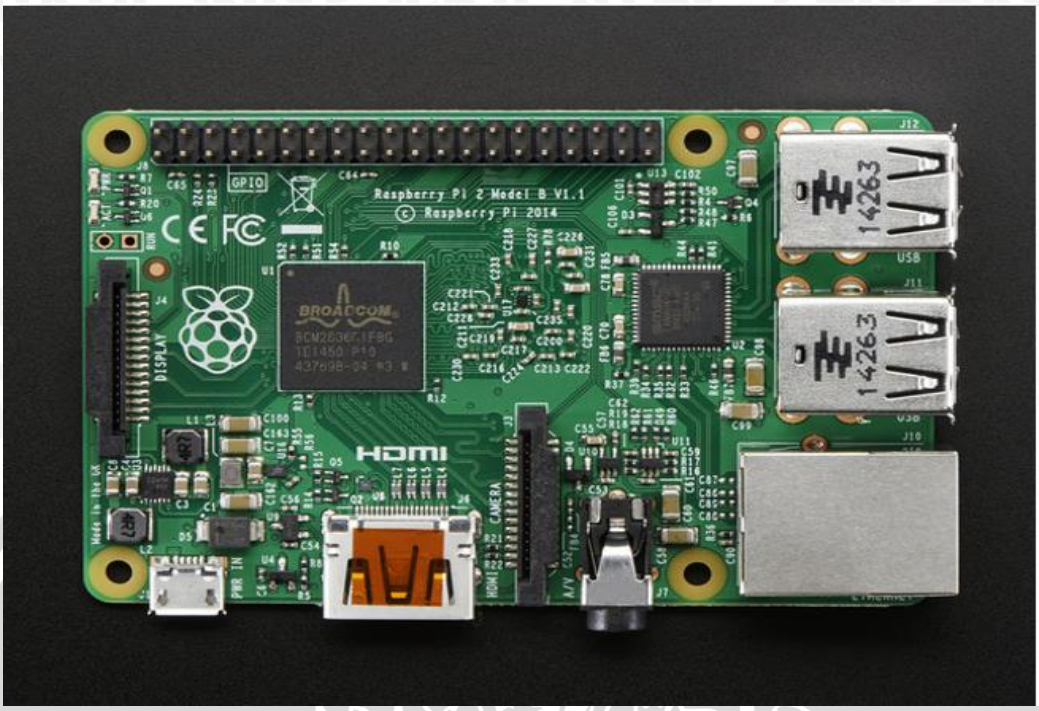


Gambar 4.2 Diagram Alir Sistem

4.6.2 Perancangan Perangkat Keras

Ada beberapa perancangan pada perangkat keras, yaitu perancangan *hardware*. Perancangan *hardware* akan diimplementasikan berdasarkan analisis kebutuhan sistem, antara lain rangkaian sensor mikrofon dan rangkaian *switch* sebagai input dari sistem kemudian mini PC Raspberry sebagai pengontrol sistem yang berfungsi untuk pemrosesan data, dan rangkaian *vibrator* sebagai *output* berdasarkan keberhasilan sistem memroses *input*.

4.6.2.1 Mikrokomputer Raspberry Pi 2B+



Gambar 4.3 Raspberry Pi 2

Sumber: Adafruit Industries. 2015. *Introducing the Raspberry Pi 2 – B Model*.

Raspberry menjadi mikrokomputer yang menjadi bagian inti dari sistem yang dirancang. Dengan menggunanakan sistem operasi Raspbian, segala proses pengolahan data sistem melalui *library* dilakukan di sini.

Tabel 4.1 Spesifikasi Mikrokomputer *Raspberry Pi 2*

Mikrokomputer *Raspberry Pi 2*	
Nama Produk	*Raspberry Pi 2, Model B*
Chip	*Broadcom BCM2836 SoC*
Core Architecture	*Quad-core ARM Cortex-A7*
Central Processing Unit	*900 MegaHertz*
GPU	*Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30* *H.264 high-profile decode* Mampu hingga *1Gpixel/s, 1.5Gtexel/s or 24GFLOPs* dengan *texture filtering* dan *DMA infrastructure*
Memory	*1GB LPDDR2*
Sistem Operasi	*Rasbian*

Dimensi	*85 x 56 x 17mm*
Catu Daya	*Micro USB socket 5V, 2A*

4.6.2.2 Perancangan Rangkaian Sensor Mikrofon

Pada perancangan rangkaian sensor mikrofon terdiri dari sensor mikrofon yang dihubungkan ke Raspberry sebagai pemroses data. Data sensor mikrofon dihubungkan ke input GPIO Raspberry sebagai jalur komunikasi.

Pada penelitian kali ini penulis akan menggunakan 2 buah mikrofon untuk meneliti seberapa besar pengaruh hasil pembacaan sensor terhadap kinerja sistem.

Gambar 4.4 Sony Mini Microphone

Mikrofon pertama adalah Sony *mini microphone* dengan spesifikasi sebagai berikut:

Tabel 4.2 Sony *Mini Microphone*

Sony *Mini Microphone*	
Nama Produk	Sony *Mini Microphone*
Tipe	MD150
Sensitivitas	-58 dB
Batas Frekuensi	8 Hz – 20 KHZ

Gambar 4.5 TOA Mikrofon Jepit

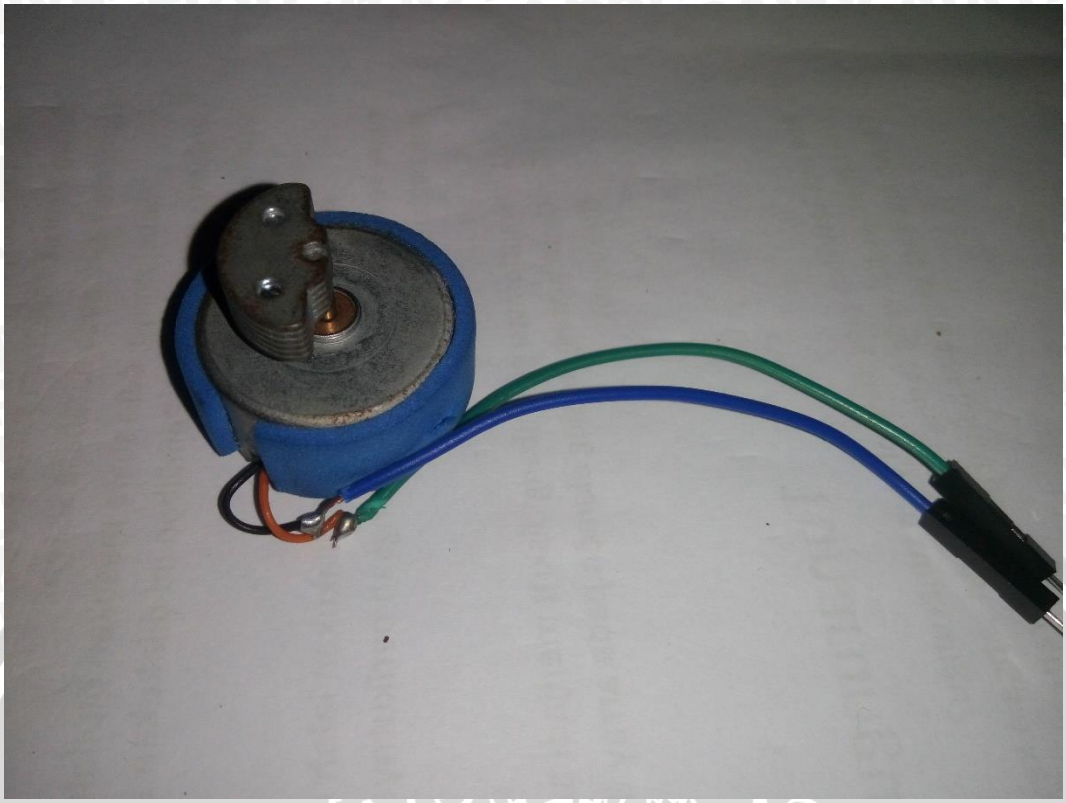
Mikrofon kedua adalah TOA Mikrofon Jepit dengan spesifikasi sebagai berikut:

Tabel 4.3 Mikrofon Jepit

TOA Mikrofon Jepit	
Nama Produk	TOA Mikrofon Jepit
Model	ZM-360
Tipe	Mikrofon Kondensor
Tanggapan Frekuensi	70 Hz – 20 KHZ (sensistivitas -20 dB di bawah 1 KHz)
Tegangan Keluaran pada 1 KHz	-61,5 dB (0,82 mV)
Baterai	1 pc of AA (R 65 atau UM-3U)

4.6.2.3 *Vibrator*

Sebagai *output* utama dalam sistem, *vibrator* memiliki peranan yang sangat penting. *Vibrator* yang digunakan adalah *vibrator* yang diambil dari *joystick*. *Vibrator* ini dapat menyala jika diberi tegangan 5 volt. Wujud *vibrator* ini ditunjukkan oleh Gambar 4.6.



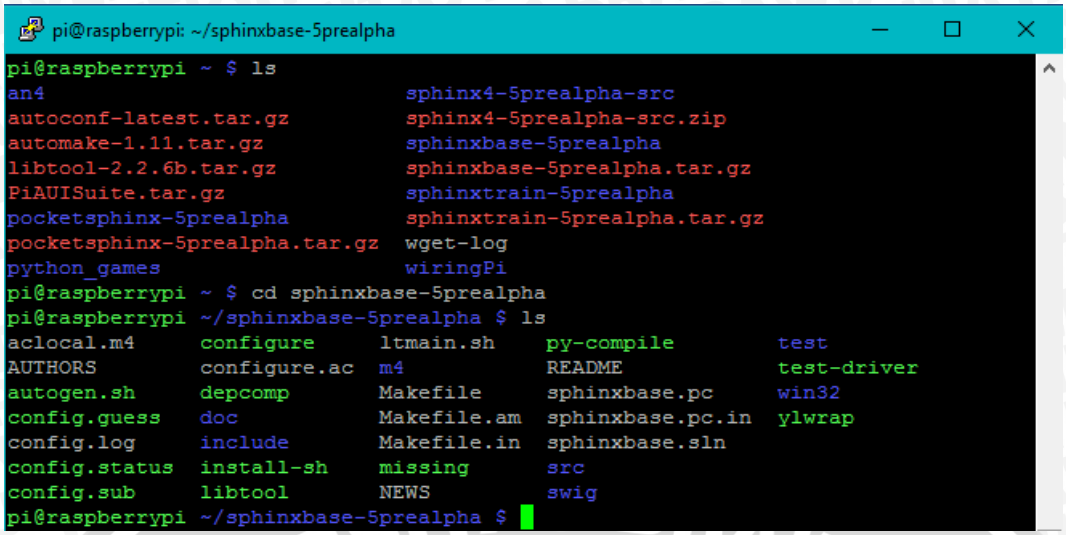
Gambar 4.6 *Vibrator*

4.6.3 Perancangan Perangkat Lunak

Pada perancangan perangkat lunak sistem akan dijelaskan perangkat lunak apa saja yang akan digunakan pada sistem ini. Implementasi HMM untuk mengenali ciri ucapan dan beberapa warna ucapan manusia memerlukan beberapa *software*, yaitu:

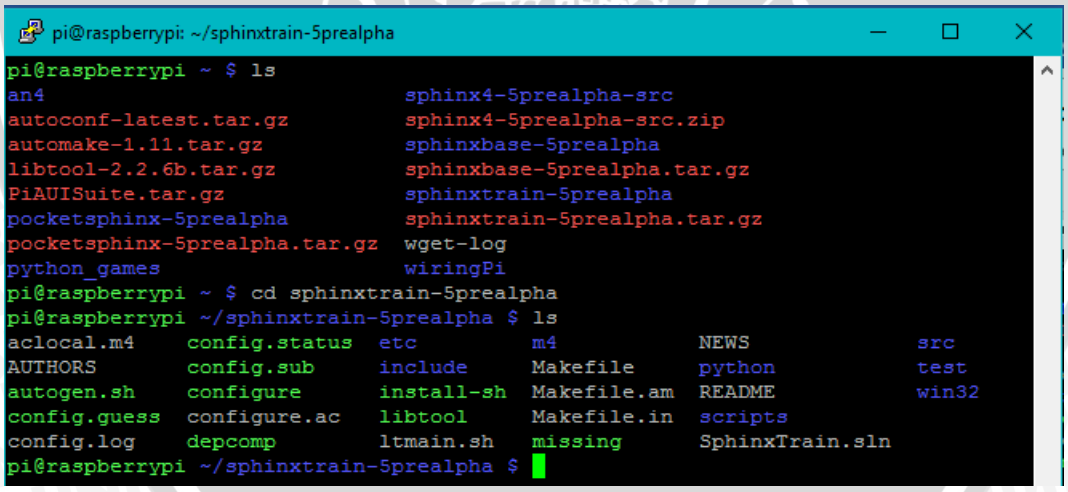
1. SphinxBase
2. SphinxTrain
3. PocketSphinx

Setelah dilakukan proses *download* dan *building*, maka dapat komponen dari *sphinxbase* ditunjukkan oleh Gambar 4.6.



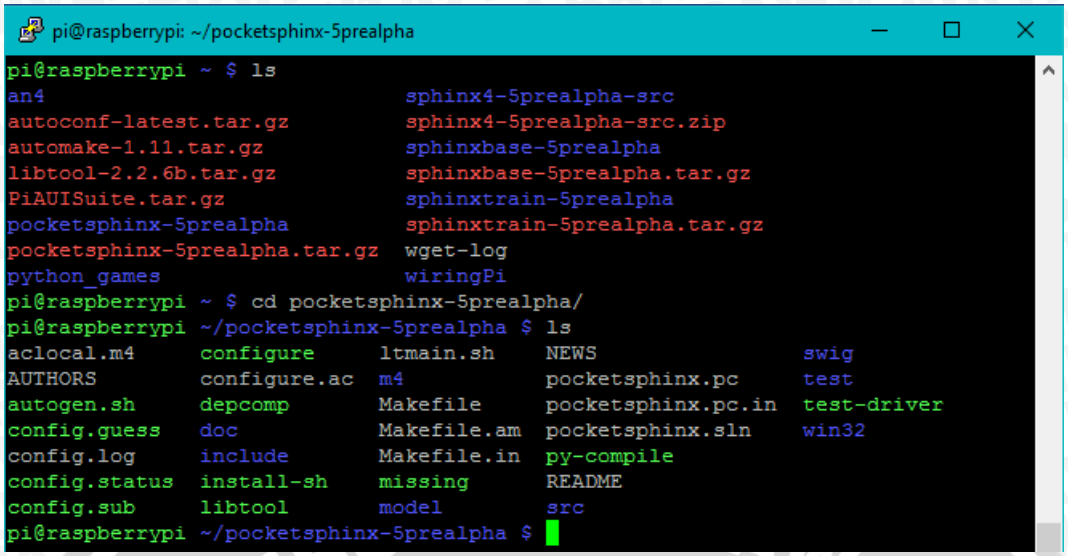
Gambar 4.7 Isi *file* sphinxbase

Secara umum, sphinxbase memiliki fungsi penting sebagai pusat *knowledge base* yang menyimpan segala kosakata yang ada dalam *dictionary*. Semakin sering suatu *state* dalam suatu kosakata diakses pada saat proses *training*, semakin mudah pula sistem dalam mendeteksi kosakata tersebut apabila muncul kembali pada kesempatan berikutnya. Dengan langkah yang sama, lakukan proses *download* dan *building* SphinxTrain untuk melakukan serangkaian aktivitas *training*. Isi dari file SphinxTrain dapat dilihat pada Gambar 4.7.



Gambar 4.8 Isi *file* sphinxtrain

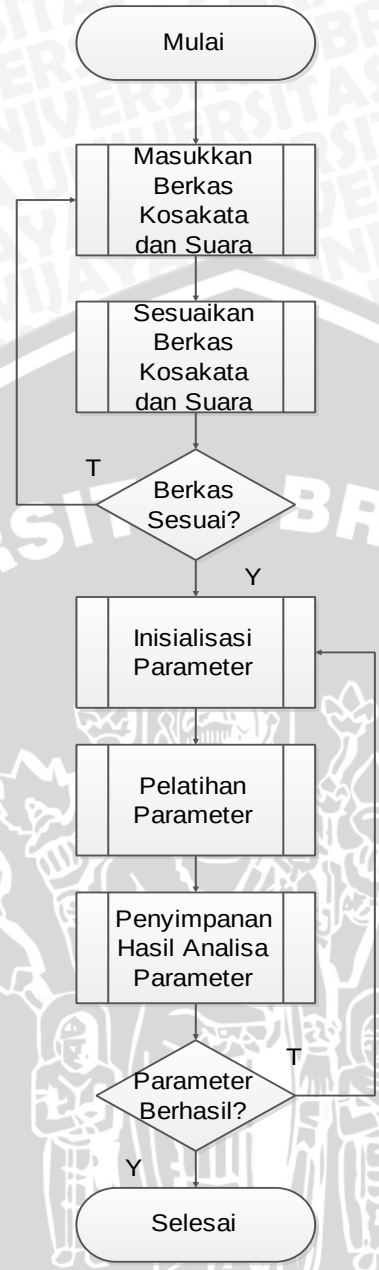
SphinxTrain memiliki peranan penting dalam mengolah suara yang dijadikan pedoman dalam proses pengenalan suara untuk diuji pada proses *testing*. Semakin sering proses *training* dilakukan, semakin mudah sistem dalam mengenali suara yang sering dilatih. Berikutnya, lakukan pula *download* dan *building* untuk PocketSphinx. Setelah selesai, isi dari file PocketSphinx dapat dilihat pada Gambar 4.8.



Gambar 4.9 Isi *file pocketsphinx*

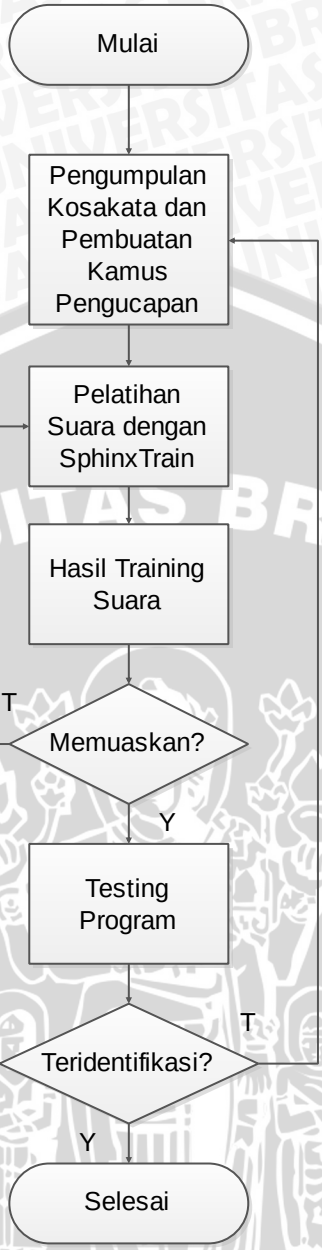
PocketSphinx adalah *library* utama dalam sistem ini yang mengimplementasikan metode *Hidden Markov Model* (HMM). *Library* ini juga bertugas untuk mencocokkan berkas berupa teks dan suara yang nantnya akan dicocokkan untuk kemudian diolah sebagai berkas utama pengenal *input* suara.

Secara garis besar, ada 2 proses utama yang akan merealisasikan implementasi sistem ini. Dua proses tersebut adalah *training* dan *testing*. Sebelum melakukan implementasi, gambaran umum proses *training* ada pada Gambar 4.9.



Gambar 4.10 Diagram Alir Proses *Training*

Setelah proses *training* selesai dilaksanakan, maka berikutnya masuk ke proses pengujian atau *testing*. Pada proses *testing*, semua berkas yang diperlukan akan digunakan. *File dictionary, language model* dan *acoustic model* dikolaborasikan untuk mengenali semua *state* yang ada dalam fonem yang diucapkan seseorang pada saat *testing* dilaksanakan. Setelah proses *training* diselesaikan, kita berlanjut ke proses berikutnya yaitu proses *testing.* Gambaran secara umum proses *testing* ada pada Gambar 4.10.



Gambar 4.11 Diagram Alir Proses *Testing*

BAB 5 IMPLEMENTASI

Bab ini menjelaskan implementasi sistem penelitian yang terdiri dari proses *training* dan *testing* agar sistem dapat bekerja secara optimal.

5.1 Implementasi Sistem

Implementasi sistem bisa dilakukan apabila proses perancangan sistem telah terpenuhi karena implementasi sistem ini secara keseluruhan mengacu pada perancangan sistem yang telah ditentukan sebelumnya. Pada bagian implementasi sistem memaparkan secara rinci spesifikasi perangkat lunak dan perangkat keras yang digunakan oleh sistem.

5.1.1 Implementasi Perangkat Keras

Pada tahap implementasi perangkat keras pada sistem ini disesuaikan dengan perancangan yang telah dilakukan sebelumnya. Sesuai dari hasil perancangan pada sistem ini terdiri dari rangkaian sensor mikrofon, rangkaian *input* dari *switch* dan rangkaian *vibrator*. Sistem ini menggunakan mini PC yang memerlukan tegangan 5 volt dan arus 2 ampere. Implementasi perangkat keras pada sistem ini dapat dilihat dari Gambar 5.1.

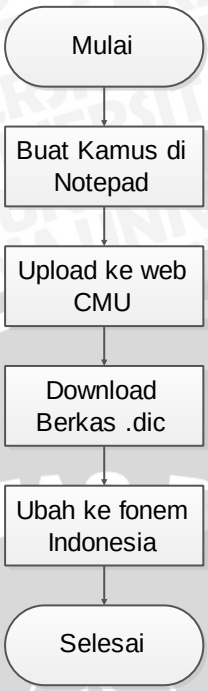
Gambar 5.1 Rangkaian Sistem Keseluruhan

5.1.2 Implementasi Perangkat Lunak

Implementasi perangkat lunak dalam penelitian ini menjelaskan proses *download file* kelengkapan, instalasi *library*, dan *tools* hingga dapat mengolah *input*.

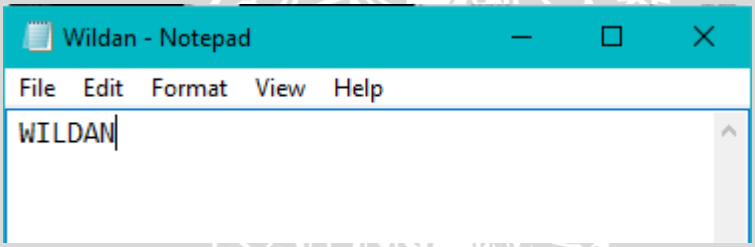
5.1.2.1 Pembuatan *Dictionary File*

Dictionary file memiliki peranan penting dalam pengenalan ucapan. Semua kata yang nantinya akan dideteksi oleh sistem harus terlebih dahulu disertakan dalam *dictionary file*. Diagram alir proses pembuatan *dictionary file* dijelaskan pada Gambar 5.2.



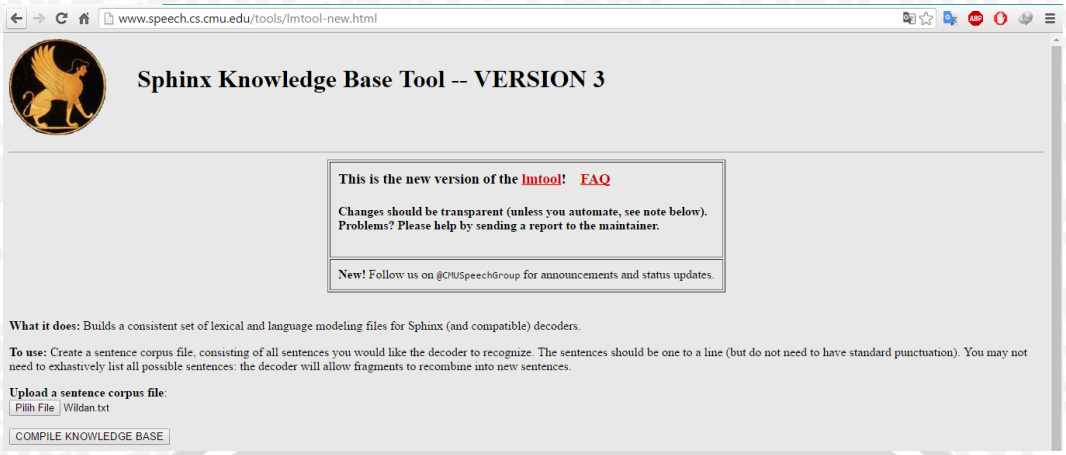
Gambar 5.2 Diagram alir pembuatan *dictionary file*

Pembuatan *file* perbendaharaan kata di *notepad* dapat dilihat pada Gambar 5.3 berikut:



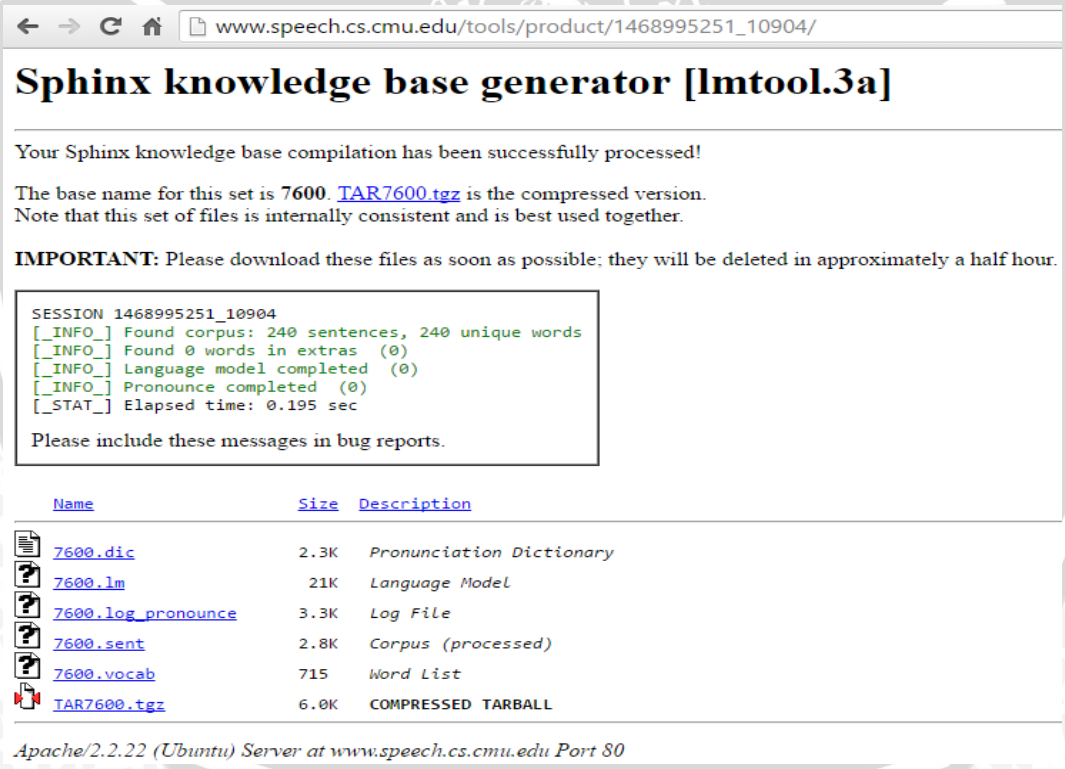
Gambar 5.3 Pengenalan nama Wildan

Setelah selesai, kemudian *upload* ke http://www.speech.cs.cmu.edu/tools/lmtool-new.html.



Gambar 5.4 SphinX *Knowledge Base Tools*

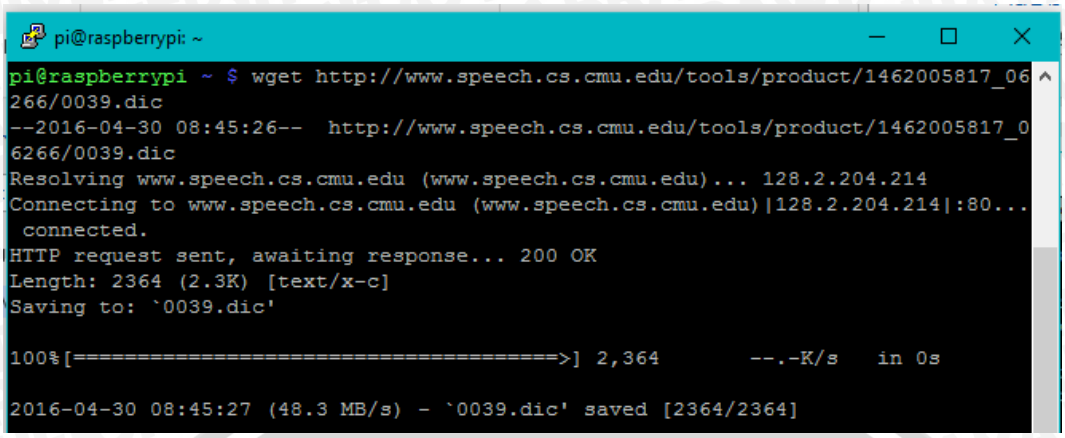
Pilih "Compile Knowledge Base" untuk melakukan kompilasi file "Wildan" agar memperoleh file *language model* dan *dictionary file* sebagai *file* kelengkapan yang diperlukan oleh PocketSphinx. Hasil kompilasi dapat dilihat pada Gambar 5.5.



Gambar 5.5 Hasil Kompilasi *file* Wildan

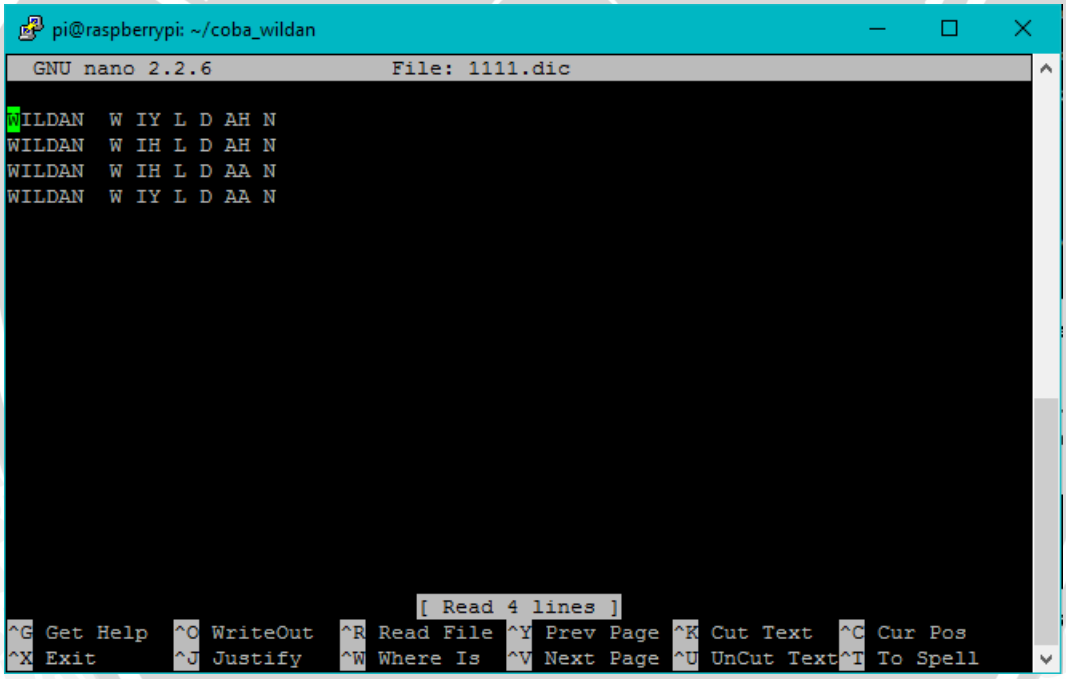
Dictinary file bernama 7600.dic dan *language model* bernama 7600.lm. Empat angka yang enjadi nama file tersebut dipilih secraa acak oleh sistem SphinX *Knowledge Base Tools*.

Language model berisi file nilai n-gram. Proses *download dictionary file* dapat dilihat pada gambar 5.6 berikut:



Gambar 5.6 Proses *download dictionary file*

Fonem dalam *dictionary file* masih dalam pelafalan bahasa Inggris, untuk melakukan pengubahan menjadi fonem bahasa Indonesia, lakukan penyesuaian dengan mengubah konten *dictionary file* hasil kompilasi SphinX *Knowledge Base Tools*. Prosesnya ditunjukkan melalui Gambar 5.7.



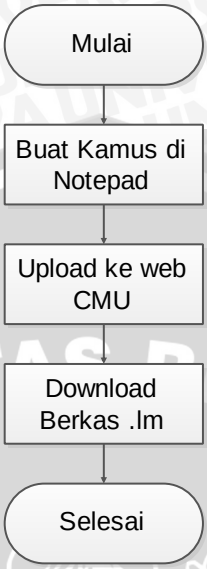
Gambar 5.7 Penyesuaian fonem Wildan menjadi bahasa Indonesia

Penyesuaian dilakukan agar nama “Wildan” dapat dipanggil dengan 4 kombinasi pelafalan, yaitu “Wildan” dengan pelafalan huruf i panjang dan a pendek, “Wildan” dengan pelafalan huruf i pendek dan a pendek, “Wildan” dengan pelafalan huruf i pendek dan a panjang, dan terakhir “Wildan” dengan pelafalan huruf i panjang dan a panjang. Semuanya dalam bahasa Indonesia.

5.1.2.2 *Pembuatan Language Models*

Language models digunakan untuk mengetahui karakteristik pengucapan suara tertentu oleh sesorang. Hal ini bertujuan untuk mengetahui variasi cara

pengucapan kata tertentu oleh orang-orang yang nantinya akan bertindak sebagai koresponden dalam proses *training*. Proses pembuatan *lnguage models* dapat dilihat pada Gambar 5.8.



Gambar 5.8 Diagram alir pembuatan *language models*

Dengan menggunakan *file* yang sama dengan *file* yang di*upload* untuk membuat *dictionary file*, kita dapat memperoleh *file language model* dari web CMU. Proses *download language model* ditunjukkan pada Gambar 5.9.

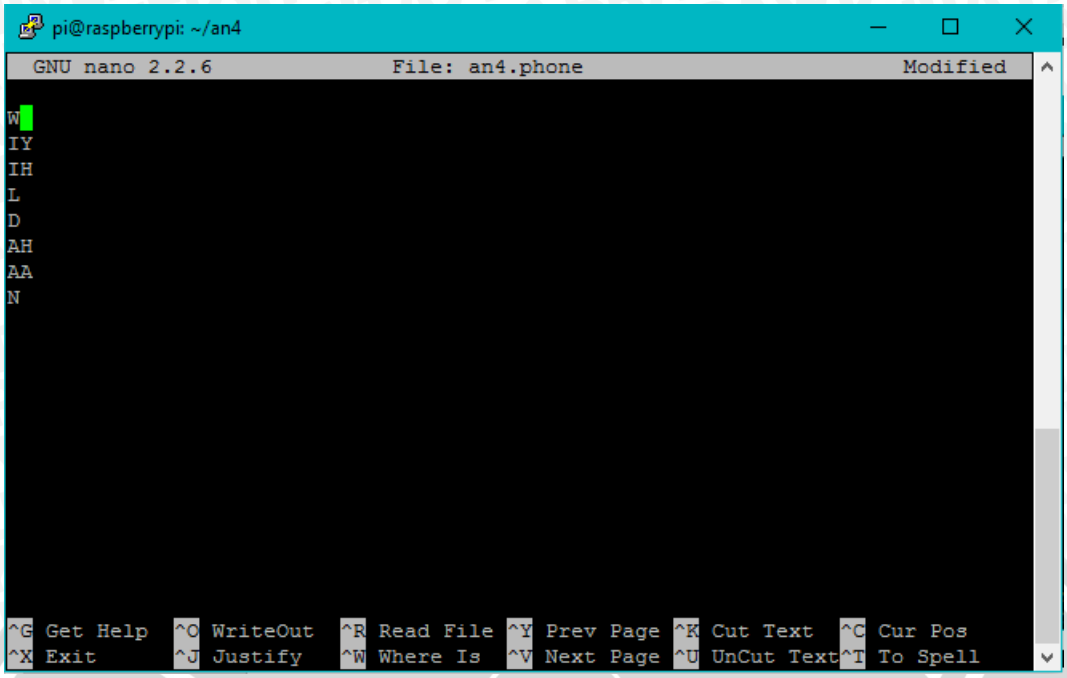
```
pi@raspberrypi ~ $ wget http://www.speech.cs.cmu.edu/tools/product/1462005817_06
266/0039.lm
--2016-04-30 08:45:31--  http://www.speech.cs.cmu.edu/tools/product/1462005817_0
6266/0039.lm
Resolving www.speech.cs.cmu.edu (www.speech.cs.cmu.edu)... 128.2.204.214
Connecting to www.speech.cs.cmu.edu (www.speech.cs.cmu.edu)|128.2.204.214|:80...
 connected.
HTTP request sent, awaiting response... 200 OK
Length: 21031 (21K) [text/plain]
Saving to: `0039.lm'

100%[======================================>] 21,031      55.4K/s   in 0.4s

2016-04-30 08:45:32 (55.4 KB/s) - `0039.lm' saved [21031/21031]
```

Gambar 5.9 Proses *download language model* dari *website* CMU

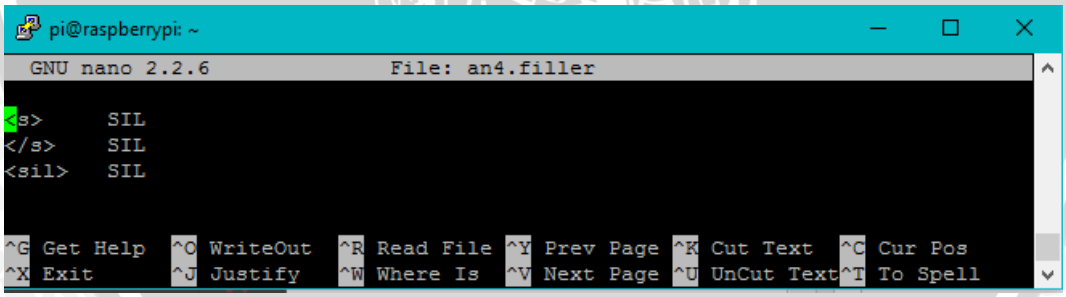
Berdasarkan isi *dictionary file*, implementasi yang dilakukan selanjutnya ialah membentuk file *phone* yang merupakan file yang berisi daftar fonetik yang digunakan dalam format ARPAbet. Pembuatan daftar fonetik menyesuaikan dengan pengucapan orang Indonesia terhadap kata yang akan diuji. Proses pembuatan *phone file* yakni data file *<nama_file>.phone* dimasukkan ke dalam *software* LibreOffice Calc yang dipisah dengan spasi. Penggambaran pemisahan file tersebut dengan *software* ditampilkan dalan Gambar 5.10.



Gambar 5.10 Isi *file .phone*

5.1.2.3 *Pembuatan File Filler*

Setelah membuat file kamus dan phone, peneliti membuat *file fillers*. *File filler* dapat dibuat secara manual berdasarkan rekaman suara pada keadaan sepi, sehingga penulisan "SIL" diartikan sebagai *silent* atau sepi. *File filler* juga memberikan tanda mulai dan berhenti kalimat pada *transcription file*. Adapun format penulisan secara default dari basis data suara di folder AN4. Bentuk *file filler* diilustrasikan dalam Gambar 5.11.

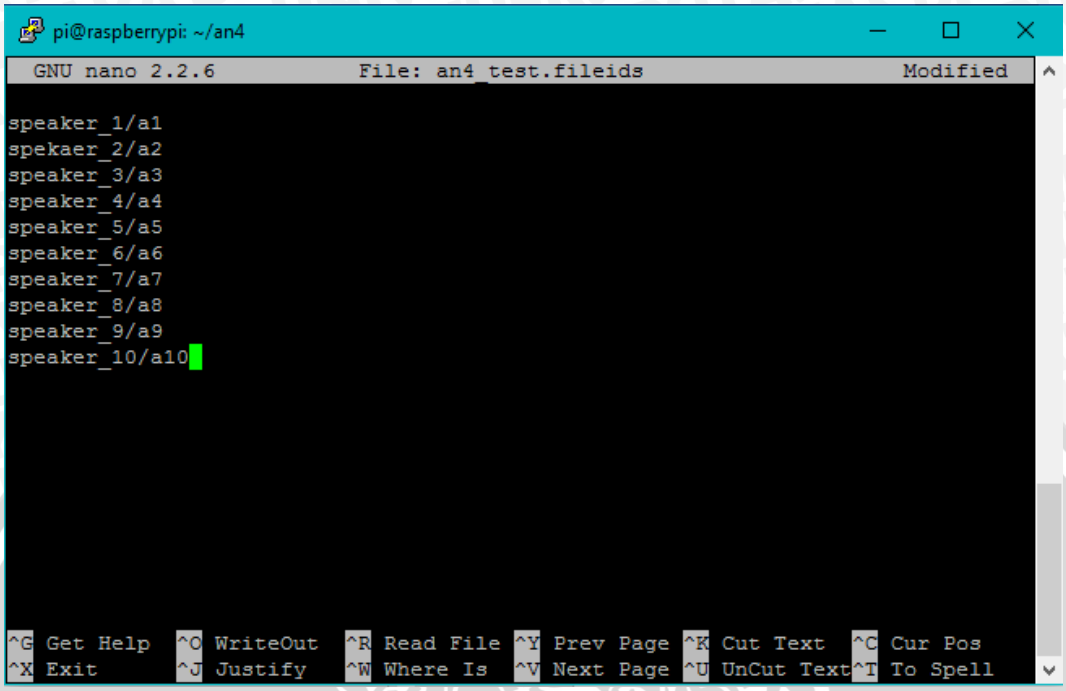


Gambar 5.11 Isi *file .filler*

5.1.2.4 Perekaman *Suara Koresponden Training*

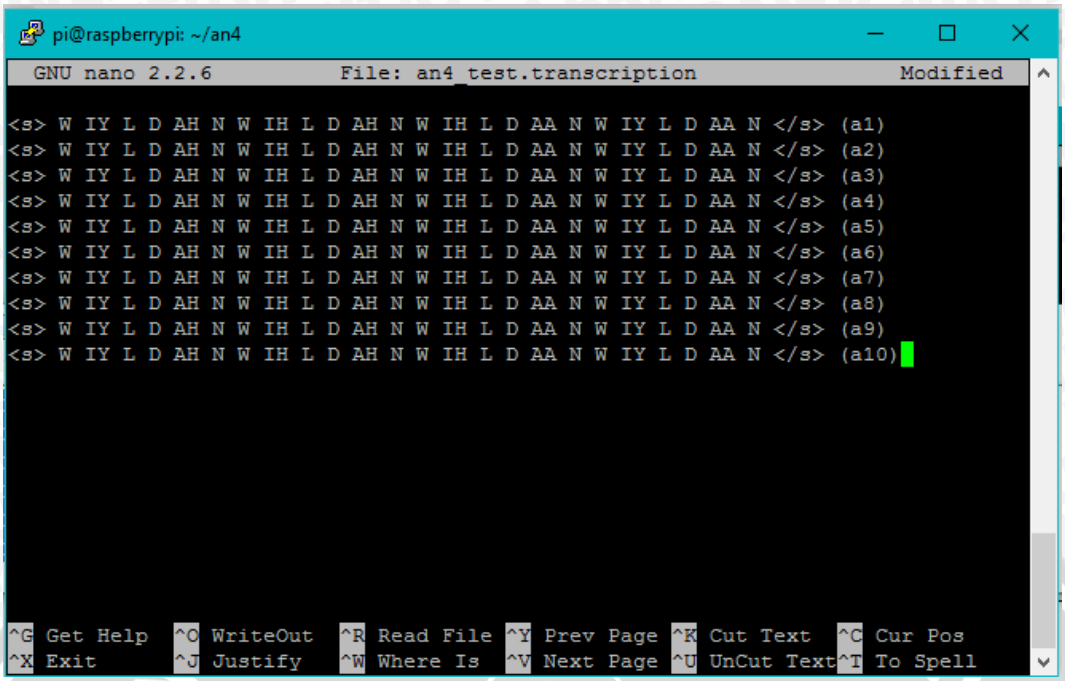
Selanjutnya, ketika semua proses sebelumnya telah dilakukan, pengolahan manual yang dilakukan saat ini ialah membuat *file transcription*. *File transcription* berfungsi sebagai transkrip yang terdiri dari transkrip *training* berdasarkan kesesuaian file rekaman suaranya. Untuk membuat *file transcription* sesuai dengan peletakan file rekaman suara dibuatlah *file fileids* yang merupakan path/penunjuk file suara yang akan di*training*. Pada *database* peletakan *file*

tersebut terdapat dua jenis *file transcription* dan *fileids* yaitu *file an4_train.fileids*, *an4_test.fileids* yang isi *file*nya disesuaikan satu sama lain, Gambar 5.13 menjabarkan kedua isi *file* tersebut.



Gambar 5.12 Isi File an4_test.fields

Sama halnya dengan *an4_train.transcription* dan *an4_test.transcription* yang berisi daftar semua kalimat yang telah dipasangkan dengan *file* suara untuk diuji coba, isi dari kedua *file* tersebut disesuaikan satu sama lain. Penjabaran isi kedua file tersebut dapat dilihat pada Gambar 5.14.

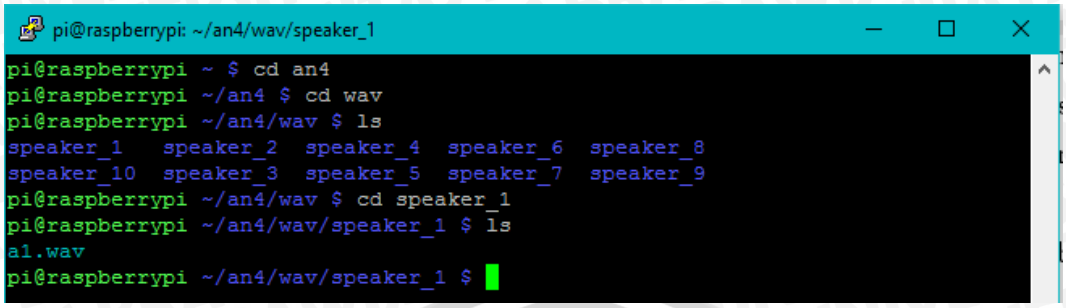


Gambar 5.13 Isi File an4_test.transcription

Setelah menyesuaikan *file transcription* dan *fileids* dibuatlah beberapa rekaman yang isi rekaman tersebut berdasarkan hasil *file transcription* yang sudah dibuat. Rekaman ini menggunakan 10 koresponden suara dengan perincian (5 laki-laki dan 5 perempuan). Hasil rekaman diletakkan pada *folder wav* yang ada pada AN4 dengan peletakan file sebagai berikut.

- *speaker*_1
  - o a1.wav
- *speaker*_2
  - o a2.wav
- *speaker*_3
  - o a3.wav

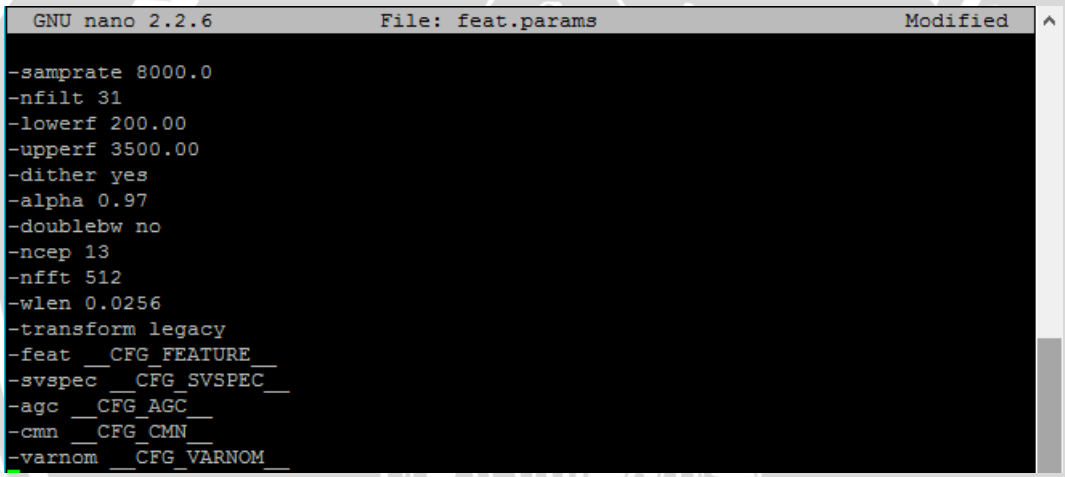
dan seterusnya. Peletakan file suara *.wav* dapat dilihat pada Gambar 5.15.



Gambar 5.14 Peletakkan *file* suara.wav

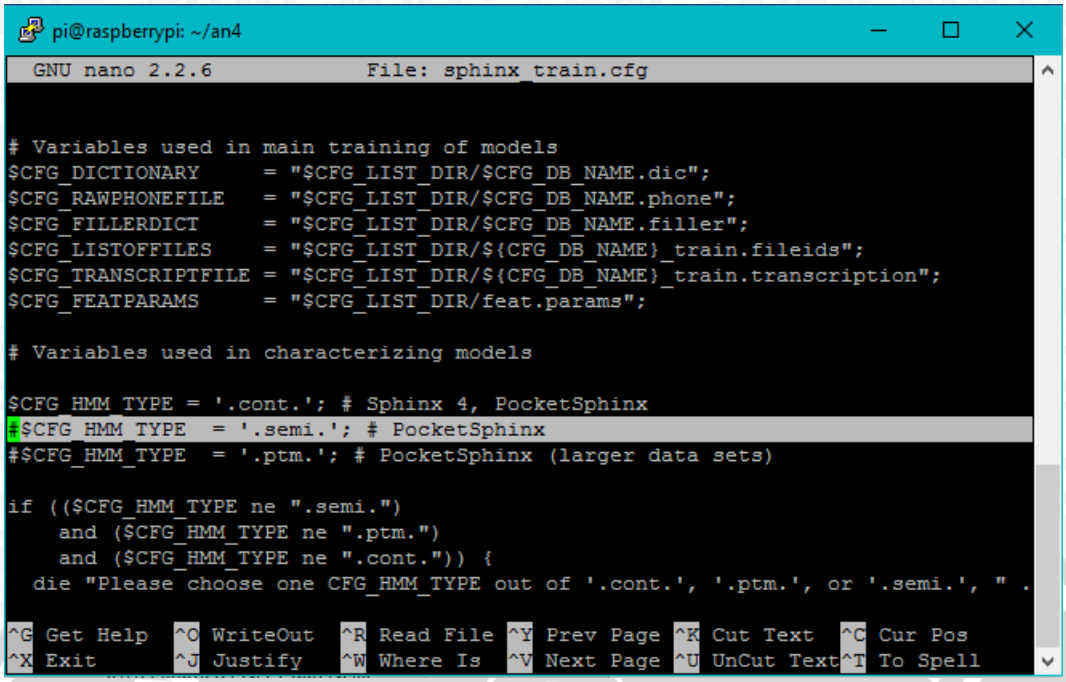
5.1.2.5 *Pengubahan Parameter*

Setelah folder rekaman diisi, selanjutnya merubah beberapa parameter yang digunakan untuk memaksimalkan hasil *training*. Pada saat training yang peneliti lakukan berdasarkan referensi yang peneliti dapat dari website CMU Sphinx, samprate menggunakan 8000 Hz, NFFT menggunakan 512, NFILT menggunakan 31 dan lowerf/upperf menggunakan 200Hz & 3500Hz, penjabaran perubahan parameter dapat dilihat pada Gambar 5.15.



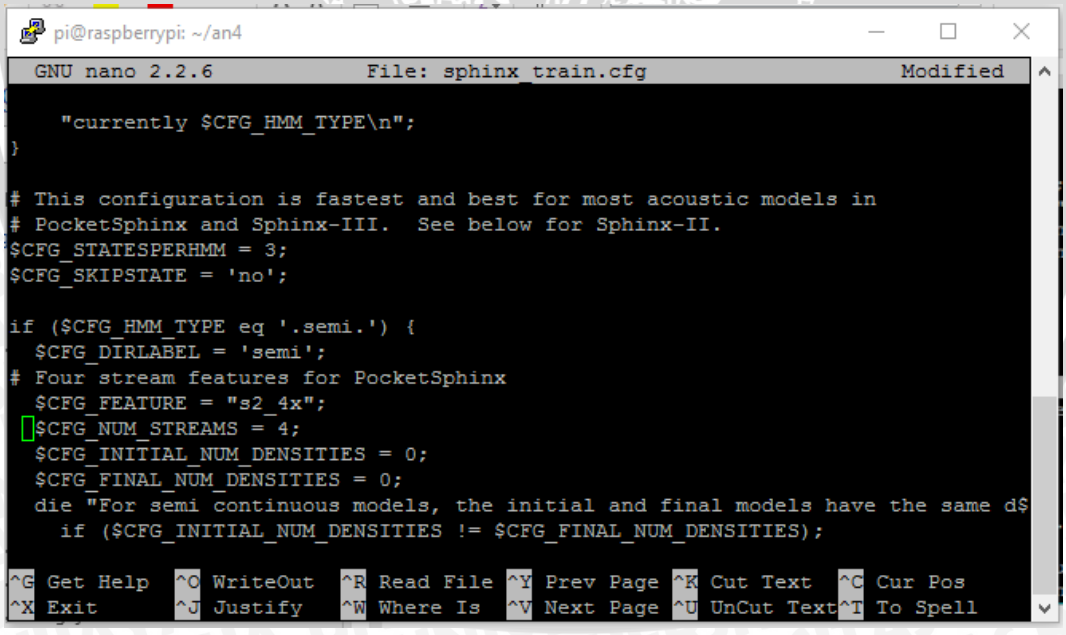
Gambar 5.15 Isi file feat.params

Berdasarkan tutorial yang diberikan oleh website CMU Sphinx untuk pembuatan *training acoustic model*, terdapat 3 tipe model jenis perekaman, yakni *continuous* (*cont*), *semi-continuous* (*semi*), dan *phonetically tied mixtures* (*ptm*). Model *ptm* digunakan jika jumlah data perekaman yang cukup banyak. Model *semi* digunakan jika jumlah data perekaman sedikit dan membutuhkan kecepatan akses saat proses *testing*. Model *cont* digunakan jika jumlah data perekaman berada di antara model *ptm* dan *semi*. Pada implementasi saat ini digunakan model *semi-continuous* model seperti dalam Gambar 5.16.



Gambar 5.16 Implementasi HMM *semi-continous*

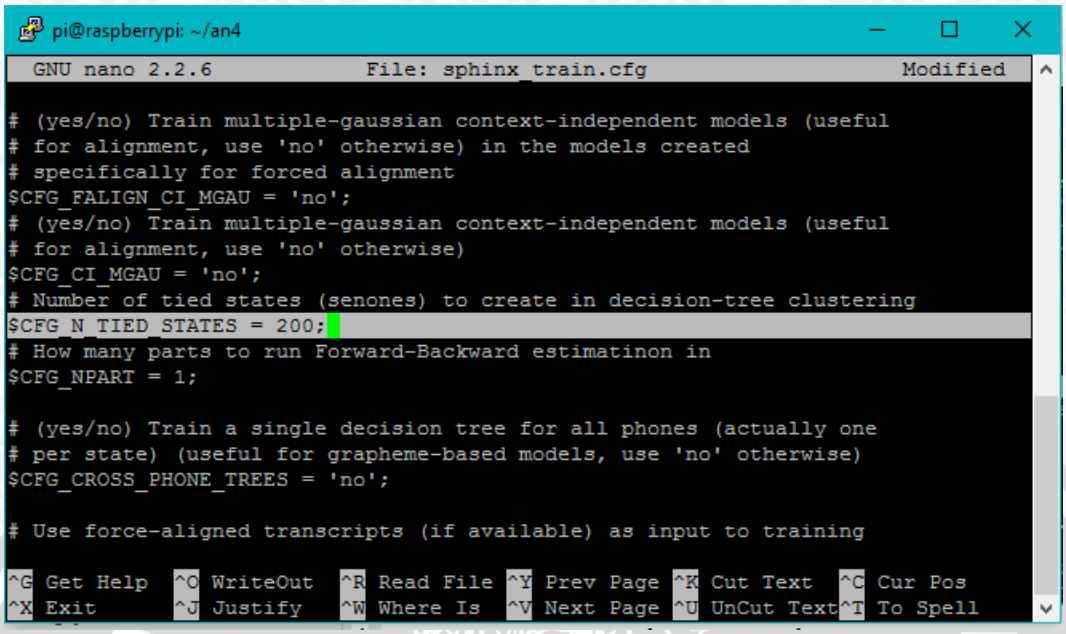
Parameter yang digunakan terdiri dari *density* dan *senone*. Jumlah *density* tergantung pada jumlah kosakata yang dimiliki basis data. Penulisan *density* berkelipatan 2, seperti 1, 4, 8, 16, 32, 64. Penjabaran penulisan *density* dalam Gambar 5.17.



Gambar 5.17 Besarnya *Density*

Penulisan *senone* juga berdasarkan jumlah kosakata, dan waktu *training* kosakata sehingga diperlukan nilai yang optimal. Pada basis data Bahasa Indonesia digunakan jumlah kosakata, waktu *training* dan *density* yang minimal.

Jika penulisan *senone* terlalu banyak, muncul pesan kesalahan sewaktu proses *training* dalam *folder log*. Penulisan *senone* diilustrasikan dalam Gambar 5.18.



Gambar 5.18 Banyaknya *Tiedstate*

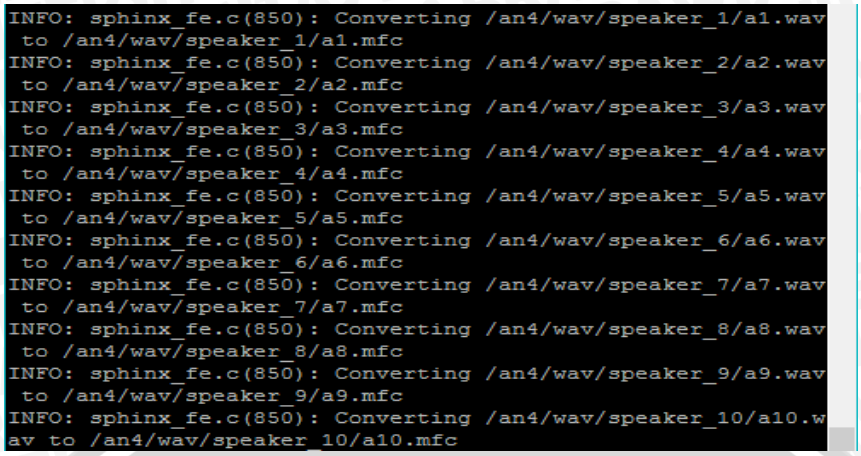
Berikut ini perkiraan perbandingan penggunaan *senones* dan *densities* untuk pelatihan *training* suara. Pada penelitian kali ini saya menggunakan senones "200" dan density "8". Penggunaan nilai *senones* dan *density* tersebut berdasarkan banyaknya basis data rekaman yang sudah saya lakukan, apabila menggunakan vocabulary minimal maka digunakan nilai yang minimal juga. Peraturan penggunakan *senones* dan *density* dilihat pada Tabel 5.1

Tabel 5.1 Perbandingan nilai *senones* dan *densities*

(Sumber: *Training Acoustic Model CMU Sphinx Documentation*)

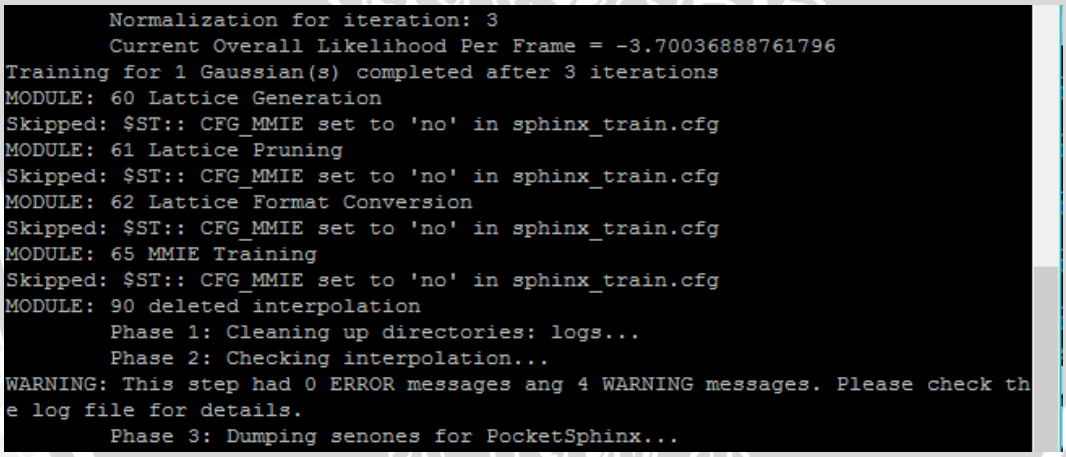
Vocabulary	Hours in db	Senones	Densities	Example
20	5	200	8	Tidigits Digits Recognition
100	20	2000	8	RM1 Command and Control
5000	30	4000	16	WSJ1 5k Small Dictation
20000	80	4000	32	WSJ1 20k Big Dictation
60000	200	6000	16	HUB4 Broadcast News
60000	2000	12000	64	Fisher Rich Telephone Transcription

Setelah melakukan konfigurasi terhadap nilai-nilai untuk persiapan *training*, ketikkan "./scripts_pl/make_feats.pl –ctl etc/an4_train.fileids" untuk melakukan konversi *file* suara wav (*Waveform Audio FIle* Format) menjadi mfc dalam bentuk *binary*. Hasil konversi file dapat dilihat pada Gambar 5.19.



Gambar 5.19 Konversi *file* wav menjadi mfc

Pada sintaks yang dilakukan pada Gambar 5.17 didapatkan pembentukan *file* baru yang digunakan untuk melakukan proses *training* suara yang dilakukan oleh *library* SphinxTrain Setelah dilakukan tahap konfigurasi basis data suara, dilanjutkan dengan penggunaan *library* SphinxTrain pada tahap *training* file suara suara telah dipetakan pada *transcription file* dan telah disesuaikan dengan *file* rekaman pada *folder wav* dengan perintah *./scripts.pl/RunAll.pl* seperti pada Gambar 5.20.



Gambar 5.20 *Modul* 90, deleted_interpolation

Teknik yang dilakukan pada Gambar 5.20 adalah *N-gram smoothing*. Penjelasan selengkapnya tentang modul training selengkapnya ada pada tabel 5.2.

Tabel 5.2 Penjelasan Modul Training

Nama Modul	Fungsi
000.*comp_feat*	Membuat direktori baru yang terdiri dari beberapa *file* kosong, dan sinyal suara diubah menjadi vektor fitur MFCC yang diubah ke dalam direktori *feat*.

00.*verify*	Pengecekan dan penyesuaian *file* yang dibutuhkan selama proses *training*. Proses *training* tidak dapat dijalankan ketika kelengkapan dari file suara, kamus, transkrip dan file path tidak sesuai.
20.*ci_hmm*	Melakukan proses *training* model *Context-Independent* (CI) pada setiap kata dalam *file dictionary*
30.*cd_hmm_untied*	Melakukan pelatihan model *Context-Dependent* dengan *untied-state* (CD-untied) berdasarkan model *Context-Independent* (CI) pada setiap kata dalam kamus.
40.*buildtrees*	Pembangunan *decision tree* untuk setiap *state* dari tiap kosakata.
45.*prunetree*	Membuat pemangkasan *decision tree* dan *tied-state*
50.*cd_hmm_tied*	Pelatihan model *CD-tied* menggunakan HMM
90.*deleted_interpolation*	Merupakan teknik N-gram *smoothing*
Decode	Pengambilan model, dilakukan pengujian berdasarkan rekaman suara dengan *file transcription* untuk estimasi kualitas model berdasarkan *Word Error Rate* (WER) dan *Sentence Error*.

5.1.2.6 *Decoding*

Berikutnya kita lakukan proses *decoding* menggunakan modul *decode*. Hasil *output* perintah *./scripts.pl/RunAll.pl*. dapat dilihat pada Gambar 5.21.

```
MODULE: DECODE Decoding using models previously trained
        Decoding 20 segments starting at 0 (part 1 of 1)
        0%
        Aligning results to find error rate
        SENTENCE ERROR: 0.0% (0/10) WORD ERROR RATE: 0.0% (0/80)
```

Gambar 5.21 Hasil Akhir Proses *Decoding*

Hasil proses *training* ini menghasilkan *Acoustic Model* yang dapat memaksimalkan pada saat melakukan *testing* pengenalan suara. *Acoustic Model* dapat dilihat dalam Gambar 5.22.

```
pi@raspberrypi: ~/an4/an4.cd_semi_200_delinterp
pi@raspberrypi ~ $ ls
an4                          sphinx4-5prealpha-src
autoconf-latest.tar.gz       sphinx4-5prealpha-src.zip
automake-1.11.tar.gz         sphinxbase-5prealpha
libtool-2.2.6b.tar.gz        sphinxbase-5prealpha.tar.gz
PiAUISuite.tar.gz            sphinxtrain-5prealpha
pocketsphinx-5prealpha       sphinxtrain-5prealpha.tar.gz
pocketsphinx-5prealpha.tar.gz wget-log
python_games                 wiringPi
pi@raspberrypi ~ $ cd an4
pi@raspberrypi ~/an4 $ ls
an4.cd_semi_200_delinterp  an4_test.fileids        an4_train.transcription  wav
an4.filler                 an4_test.transcription  feat.params
an4.phone                  an4_train.fileids       sphinx_train.cfg
pi@raspberrypi ~/an4 $ cd an4.cd_semi_200_delinterp
pi@raspberrypi ~/an4/an4.cd_semi_200_delinterp $ ls
feat.params  means            noisedict  transition_matrices
mdef         mixture_weights  sendump    variances
pi@raspberrypi ~/an4/an4.cd_semi_200_delinterp $
```

Gambar 5.22 Komponen *acoustic model*

5.1.2.7 *Testing*

Dalam pengujian testing suara, adanya kebutuhan beberapa *file* seperti *language model*, *acoustic model*. Selanjutnya dari beberapa kebutuhan *file* tersebut maka dapat dipanggil seperti *listing code* dalam program pocket.c yang ditunjukkan pada Gambar 5.23.

```
pi@raspberrypi: ~/an4
  GNU nano 2.2.6              File: pocket.c                    Modified

int main (int argc, char *argv[])
{

        if (initElements (
                        MODELDIR "/2799.lm",
                        MODELDIR "/2799.dic",
                        NATIVE_MODELDIR "/hmm/en_US/hub4wsj_sc_8k" ))
                play ();
        else
        {
                printf("Init failed...");
                return -1;
        }

        return 0;
}

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

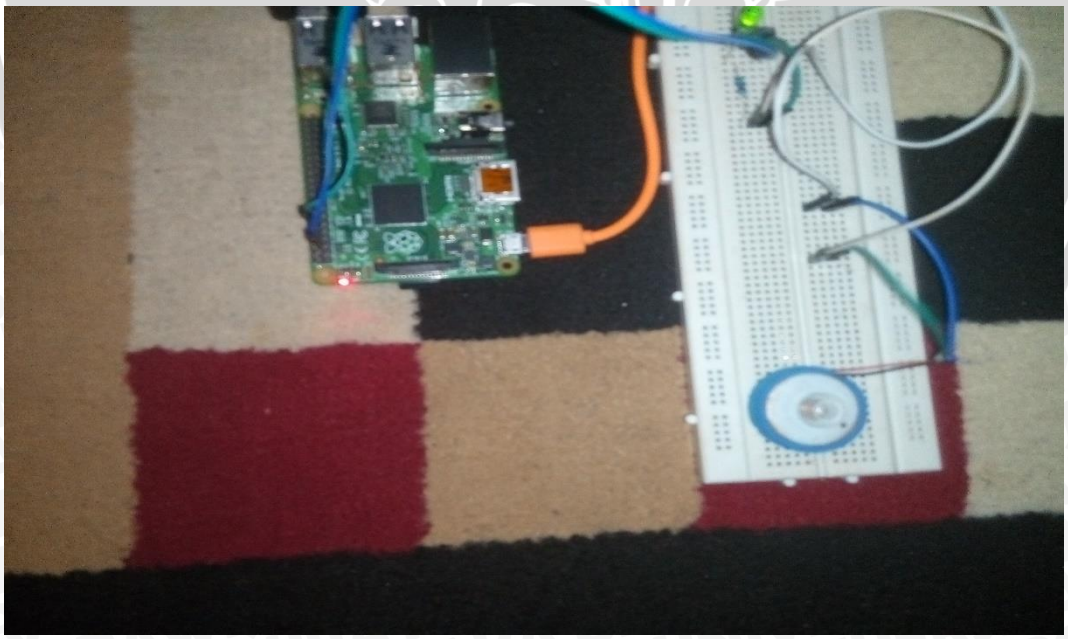
Gambar 5.23 Potongan Program pocket.c bagian Pemanggilan Data

Selanjutnya pengucapan kata dengan mikrofon. Sinyal suara diekstraksi fiturnya dan dilakukan proses dekoding secara otomatis dengan *library* PocketSphinx sehingga kata yang diucapkan ialah "WILDAN" dengan pelafalan fonem "WILDAN" seperti dalam Gambar 5.24.

```
INFO: ngram_search_fwdtree.c(147): 0 root, 0 non-root channels, 12 single-p
hone words
INFO: ngram_search_fwdtree.c(186): Creating search tree
INFO: ngram_search_fwdtree.c(191): before: 0 root, 0 non-root channels, 12
single-phone words
INFO: ngram_search_fwdtree.c(326): after: max nonroot chan increased to 346
INFO: ngram_search_fwdtree.c(338): after: 49 root, 218 non-root channels, 1
1 single-phone words
INFO: cmn_prior.c(121); cmn_prior_update: from < 56.00 -3.00 1.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 >
INFO: cmn_prior.c(139); cmn_prior_update: to   < 52.52 1.55 -2.35 1.77 1.2$
2.08 0.18 0.70 -0.52 0.10 -0.50 -0.19 -0.14 >
INFO: ngram_search_fwdtree.c(1549): 280 words recognized (11/fr)
INFO: ngram_search_fwdtree.c(1551): 11444 senones evaluated (499/fr)
INFO: ngram_search_fwdtree.c(1553): 557 channels searched (211/fr), 6321 ls
t, 5376 last
INFO: ngram_search_fwdtree.c(1557): 294 words for which last channels evalu
ated (12/fr)
INFO: ngram_search_fwdtree.c(1560): 256 candidate words for entering last p
hone (7/fr)
INFO: ngram_search_fwdtree.c(1562): fwdtree 0.08 CPU 0.057 xRT
INFO: ngram_search_fwdtree.c(1565): fwdtree 0.83 CPU 0.622 xRT
Nama Dipanggil: WI IL DA AN
```

Gambar 5.24 Hasil *Testing* Pemanggilan Nama Wildan

Secara keseluruhan rangkaian sistem, hasil *output* sistem ini berupa getaran dari *vibrator* dan nyala indikator lampu LED yang ditunjukkan pada Gambar 5.25.



Gambar 5.25 Hasil *output* sistem

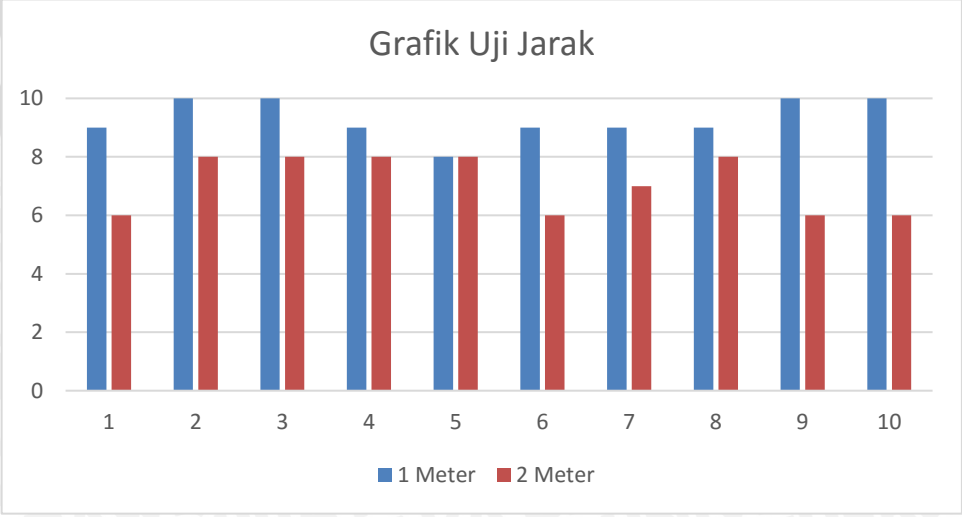
BAB 6 PENGUJIAN DAN ANALISIS

Pengujian dilakukan untuk memperoleh hasil yang kemudian dianalisis apakah hasil dari pengujian sistem ini bekerja sesuai perancangan dan bisa menjawab rumusan masalah. Pengujian dan analisis sistem ini melalui pengujian perangkat keras dan pengujian perangkat lunak. Dari hasil pengujian terdapat pembahasan dari setiap tahap pengujian yang dilakukan. Sedangkan tahap analisis bertujuan untuk memperoleh kesimpulan dari hasil pengujian dan hasil implementasi sistem yang mengacu pada dasat teori. Pengujian dan analisis dari sistem ini meliputi pengujian mikrofon, pengujian jarak dan pengujian nama.

6.1 Pengujian Jarak

Pengujian jarak dilakukan untuk membuktikan adanya pengaruh jarak antara sumber suara dengan sensor mikrofon. Secara teori, semakin jauh jarak antara sumber suara dengan sensor mikrofon maka semakin kecil pula akurasi sensor mikrofon dalam menangkap sinyal suara. Pada pengujian kali ini penulis menguji sistem pada jarak 1 meter dan 2 meter menggunakan nama uji yang sama yaitu "Wildan" dengan pelafalan fonem "WI IL DA AN".

Pengujian ini memerlukan 10 orang koresponden yang terdiri dari 5 orang laki-laki dan 5 orang perempuan agar memberikan variasi karakter suara yang nantinya akan digunakan pada saat *training*. Setiap orang melakukan perekaman suara mereka masing-masing saat menyebutkan nama Wildan sebanyak 10 kali. Jumlah perekaman dilakukan 10 kali atas pertimbangan semakin sering *training* dilakukan dan semakin banyak suara manusia berbeda yang pernah dikenali oleh sistem, maka semakin mudah sistem untuk mendeteksi panggilan nama tersebut ketika ada panggilan pada saat *testing*. Data hasil pengujian jarak selengkapnya ditampilkan pada Grafik 6.1.



Grafik 6.1 Grafik Hasil Pengujian Panggilan Nama Wildan Menggunakan Mikrofon TOA Pada Jarak 1 dan 2 Meter

Pada Grafik 6.1, garis Akurasi 1 menunjukkan tingkat akurasi pendeteksian panggilan nama Wildan pada jarak 1 meter dari sensor mikrofon dengan rata-rata tingkat akurasinya ditunjukkan oleh garis rata-rata 1. Hasil dari pengujian yang dilakukan oleh 10 orang, rata-rata akurasi menunjukkan angka 93% input suara panggilan berhasil dideteksi dari total 100 kali percobaan. Angka 1 sampai 10 pada Grafik 6.1 menunjukkan koresponden dengan rincian koresponden 1 sampai 5 adalah perempuan dan koresponden 6 sampai 10 adalah laki-laki.

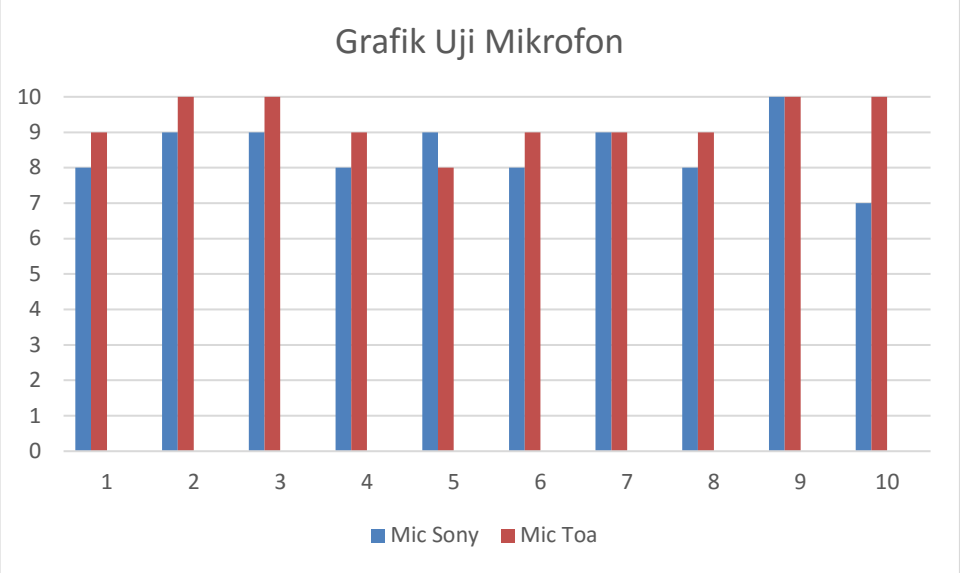
Sedangkan garis Akurasi 2 pada Grafik 6.1 menunjukkan tingkat akurasi pendeteksian panggilan nama Wildan pada jarak 2 meter dari sensor mikrofon dengan rata-rata tingkat akurasinya ditunjukkan oleh garis rata-rata 2. Hasil dari pengujian yang dilakukan oleh 10 orang, rata-rata akurasi menunjukkan angka 71% input suara panggilan berhasil dideteksi dari total 100 kali percobaan.

Hasil pengujian jarak menunjukkan bahwa dengan menggunakan mikrofon yang sama, sistem dapat mendeteksi suara panggilan nama pengguna sistem dengan lebih baik apabila sumber suara semakin dekat dengan sensor mikrofon. Hal ini berkaitan dengan tingkat kejelasan suara yang mampu dideteksi oleh sensor yang memiliki batasan sensitivitas tertentu. Dengan menggunakan mikrofon yang sama yaitu TOA Mikrofon Jepit, sistem menghasilkan daya tangkap suara yang berbeda pula akibat pemasangan sensor mikrofon yang berbeda. Detail hasil pengujian jarak ada pada Lampiran 3.

6.2 Pengujian Mikrofon

Pengujian mikrofon dilakukan untuk menganalisa tingkat pengaruh spesifikasi mikrofon yang digunakan dalam sistem sebagai penangkap sinyal suara. Secara hipotesis, mikrofon yang memiliki sensitivitas yang lebih baiklah yang akan memberikan hasil penangkapan suara yang lebih baik. Melalui pengujian ini kita dapat mengetahui kebenarannya.

Pengujian mikrofon pertama mengaplikasikan sistem dengan sensor mikrofon Sony *Mini Micrphone* MD150. Nama yang akan diuji adalah Wildan, dengan pelafalan fonem “WI IL DA AN” pada jarak 1 meter antara sumber suara dengan sensor mikrofon. Jarak 1 meter dipilih berdasarkan hasil analisa tingkat akurasi penangkapan sinyal suara oleh sensor mikrofon pada pengujian jarak yang telah dilakukan sebelumnya. Data hasil pengujian mikrofon ditampilkan pada Grafik 6.2.



Grafik 6.2 Grafik Hasil Pengujian Panggilan Nama Wildan Menggunakan Sony *Mini Mcrophone* dan Mikrofon Mini TOA Pada Jarak 1 Meter

Pada Grafik 6.2, garis Akurasi 1 adalah hasil pengujian sistem menggunakan Sony *Mini Microphone* yang dilakukan sebanyak 100 kali oleh 10 orang, setiap orang melakukan pengujian sebanyak 10 kali. Rata-rata akurasinya ditunjukkan oleh garis rata-rata 1 yang menunjukkan angka 85%. Hal ini berarti dari 100 kali percobaan, ada 15 kali percobaan panggilan suara tidak dapat dideteksi oleh sistem. Sedangkan garis Akurasi 2 pada Grafik 6.2 menunjukkan hasil pengujian sistem menggunakan TOA Mikrofon Jepit. Dari pengujian yang juga dilakukan sebanyak 100 kali, keberhasilan sistem mendeteksi *input* panggilan nama Wildan mencapai 93% yang artinya hanya 7 dari total 100 kali pengujian mengalami kegagalan pendeteksian. Angka 1 sampai 10 pada Grafik 6.2 menunjukkan koresponden dengan rincian koresponden 1 sampai 5 adalah perempuan dan koresponden 6 sampai 10 adalah laki-laki.

Analisa hasil pengujian menunjukkan bahwa dengan nama yang sama dan jarak antara sensor dengan sumber suara yang sama pula menghasilkan akurasi penangkapan yang berberda akibat adanya penggunaan sesnsor mikrofon yang berbeda. Dengan spesifikasi yang berbeda, dua buah sensor mikrofon menghasilkan dua hasil penangkapan suara yang berbeda pula pada sistem yang sama. Dengan nama yang sama yaitu Wildan, sensor Sony *Mini Micrphone* MD150 dapat mendeteksi dengan rata-rata akurasi 85%, sedangkan sensor mikrofon TOA Mikrofon Jepit ZM-360 dapat mendeteksi dengan rata-rata akurasi mencapai 93%.

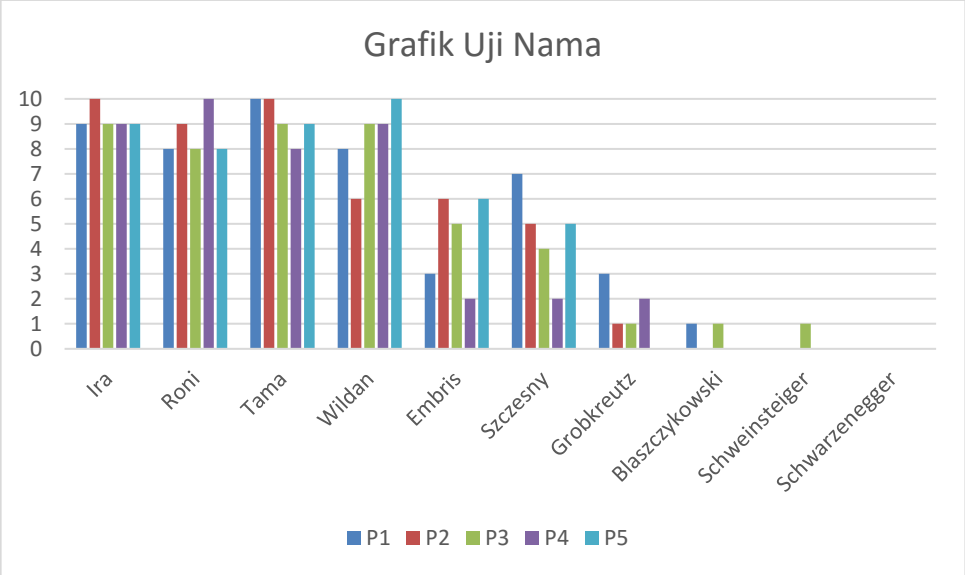
6.3 Pengujian Nama

Pengujian nama dilakukan untuk mengetahui pengaruh tingkat kerumitan pelafalan nama yang akan diuji terhadap hasil pengujian. Secara teori, nama yang sulit dilafalkan oleh orang Indonesia akan mempengaruhi tingkat keberhasilan sistem dalam mendeteksi *input* suara panggilan nama pengguna

sistem. Hal ini diakibatkan oleh perbedaan cara pelafalan nama yang sama oleh orang yang berasal dari daerah yang jauh berbeda, misalnya pelafalan nama "Schweinsteiger" oleh orang Indonesia dan orang Inggris. Perbedaan pelafalan dengan tingkat kerumitan tertentu dapat mempengaruhi kinerja sistem.

Pengujian kali ini tetap dilakukan oleh 10 orang dengan menggunakan TOA Mikrofon Jepit dalam jarak 1 meter antara koresponden dengan sistem, namun jumlah nama yang akan diuji ada 10 buah nama yang terdiri dari 5 nama yang mudah dilafalkan dan 5 nama yang sulit dilafalkan oleh orang Indonesia. Pemilihan nama-nama tersebut dilakukan secara acak oleh penulis dengan mempertimbangkan faktor klaster. Klaster adalah istilah yang menunjukkan keberadaan dua buah huruf konsonan yang beriringan dalam sebuah kata. Pada kasus ini kata yang dimaksud adalah nama orang. Semakin banyak klaster dalam suatu nama, secara teori semakin rumit pula nama tersebut dilafalkan. Semakin susah suatu nama dilafalkan, maka semakin susah pula sistem mendeteksinya.

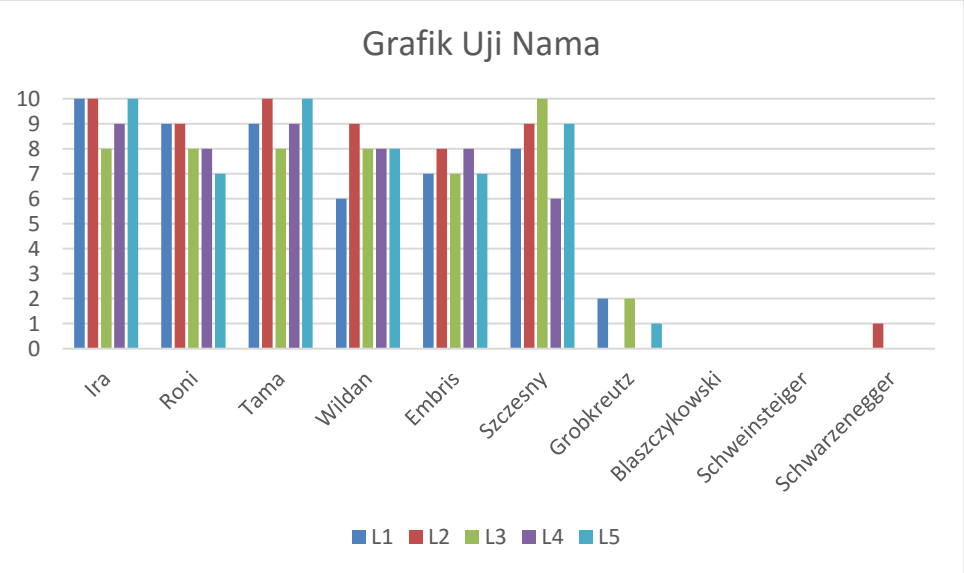
Nama yang dipilih untuk kategori nama mudah adalah Ira, Roni, Tama, Wildan dan Embris. Sedangkan nama yang dipilih untuk kategori nama rumit adalah Szczesny, Grobkreutz, Blaszczykowski, Schweinsteiger, dan Schwarzenegger. Pertimbangan yang digunakan untuk pemilihan nama mudah adalah nama orang Indonesia karena nama-nama tersebut adalah nama yang paling familiar untuk didengar dan diucapkan dalam kehidupan sehari-hari. Sedangkan nama rumit dipilih berdasarkan jumlah klaster yang ada pada nama tersebut. Nama-nama rumit tersebut juga merupakan nama orang luar Indonesia yang jarang disebutkan, terutama bagi koresponden yang belum pernah mendengar pelafalan nama tersebut dengan benar berdasarkan pelafalan dalam bahasa Indonesia. Hasil pengujian 10 nama tersebut dapat dilihat pada Grafik 6.3 untuk koresponden laki-laki.



Grafik 6.3 Hasil Pengujian Nama Rumit Oleh Koresponden Laki-laki

Grafik hasil pengujian nama rumit oleh koresponden laki-laki menunjukkan bahwa memang ada pengaruh yang signifikan dalam pemilihan

nama yang diuji menggunakan sistem ini. Sebenarnya seluruh koresponden laki-laki sudah mengetahui pelafalan nama-nama rumit yang diuji, namun dalam proses pengujiannya tidak demikian. Meskipun koresponden menganggap bahwa pelafalan nama uji yang dia lakukan sudah tepat dan volume suara saat pelafalan cukup jelas, namun sistem tidak dapat mendeteksinya dengan baik. Atas beberapa analisa tersebut, penulis menyimpulkan bahwa tingkat kerumitan nama mempengaruhi akurasi penangkapan sinyal suara oleh sensor. Semakin rumit suatu nama, semakin sulit pula sistem untuk mendeteksinya. Berikutnya Grafik 6.4 menunjukkan hasil pengujian nama yang dilakukan oleh koresponden perempuan.



Grafik 6.4 Hasil Pengujian Nama Rumit Oleh Kresponden Perempuan

Grafik hasil pengujian nama rumit oleh koresponden perempuan tidak menunjukkan hasil yang terlalu berbeda. Sebenarnya para kresponden wanita ada yang tidak mengetahui pelafalan yang benar untuk beberapa nama rumit, namun masih ada panggilan dari mereka yang mampu dideteksi oleh sistem. Pada jarak yang sama, menggunakan sistem dan mikrofon yang sama, data hasil pengujian nama oleh koresponden perempuan ini memang membuktikan bahwa ada pengaruh yang signifikan dalam penangkapan suara oleh sistem akibat adanya tingkat kerumitan yang berbeda dalam nama yang diuji oleh sistem.

Setelah melihat secara keseluruhan hasil pengujian pada pengenalan ucapan panggilan nama pengguna sistem menggunakan *library* PocketSphinx yang hanya akurasi tidak pernah mencapai 100%, penulis menganalisa bahwa terdapat beberapa faktor yang dapat mempengaruhi *library* PocketSphinx dalam menangkap sinyal suara. Hal tersebut disebabkan oleh beberapa faktor yaitu:

1. *Library* PocketSphinx tidak mendukung pengenalan ucapan dalam bahasa Indonesia, sehingga fonem pada kosakata harus mengikuti pola fonem bahasa inggris.

2. Perekaman suara tidak semua dilakukan di ruang rekaman khusus sehingga ada *noise* yang ikut terekam.
3. Terdapat beberapa kosakata yang pengucapannya hampir mirip berdasarkan fonemnya, sehingga sulit dikenali dengan baik oleh *library*.

Dengan hasil pengujian tersebut, penulis yakin bahwa Pocketsphinx yang sebelumnya hanya mampu mengenali ucapan, dapat dikembangkan lebih baik lagi untuk mengenali maksud ucapan, sehingga dapat melakukan interaksi dengan manusia di topik yang lebih luas, bukan hanya mendeteksi nama seseorang.

BAB 7 PENUTUP

Berdasarkan rumusan masalah yang diangkat dan sistem telah melalui tahap perancangan, implementasi dan dilakukan pengujian dapat ditarik kesimpulan bahwa sistem ini telah berhasil memenuhi kebutuhan *user* beserta kebutuhan fungsionalnya. Berikut adalah hasil penelitian yang didapat:

7.1 Kesimpulan

1. Pengenalan ucapan pada sistem ini menggunakan *library* PocketSphinx, PocketSphinx merupakan salah satu *library* pengenalan ucapan dari CMUSphinx yang dapat diimplementasikan pada perangkat *mobile* dan sistem *embedded*. Proses pencuplikan suara, pengolahan suara dan pendeteksian suara dilakukan untuk mengubah input suara menjadi getaran agar dapat dirasakan oleh tunarungu selaku pengguna sistem.
2. Sistem ini dapat bekerja dengan performa terbaik apabila sistem menggunakan mikrofon bertipe *omnidirectional* yang memiliki sensitivitas yang lebih baik didahului dengan proses *training* di ruangan khusus rekaman agar tidak ada *noise* yang ikut terekam. Pada saat *testing*, semakin sedikit *noise* yang terdeteksi, semakin mudah sensor mikrofon mendeteksi panggilan nama pengguna sistem.

7.2 Saran

Saran yang untuk pengembangan sistem ini antara lain:

1. Untuk pengembangan lebih lanjut, penerapan aplikasi masih menggunakan *library* dengan *database* bahasa Inggris sehingga kata-kata yang terdapat di kamus pengucapan harus disesuaikan dengan pelafalan orang Indonesia. Harapan untuk penelitian berikutnya dapat membuat *library* yang dapat mengenali pola pengucapan bahasa Indonesia murni.
2. Pada saat penambahan perbendaharaan kata, proses perekaman suara sebaiknya dilakukan di ruangan yang kedap suara agar tidak banyak *noise* yang ikut terekam dan menggunakan mikrofon yang kualitasnya bagus supaya suara terdengar lebih jelas.
3. Penggunaan Raspberry Pi memang sangat baik untuk perancangan sistem yang dibuat *mobile*, namun langkah lebih baik apabila pada penelitian berikutnya menggunakan mini PC dengan spesifikasi yang lebih tinggi seperti Odroid atau Roseapple Pi.
4. Perlu dilakukan otomatisasi agar sistem ini dapat digunakan oleh semua orang dengan proses *training* dan *testing* yang lebih mudah.
5. Proses *training* sangat penting untuk melakukan pendeteksian nama yang akan diuji saat *testing*. Semakin rumit pelafalan suatu nama, maka diperlukan intensitas *training* yang lebih sering pula dibandingkan dengan nama-nama yang mudah.

DAFTAR PUSTAKA

Adafruit Industries, 2015. *Introducing the Raspberry Pi 2 – B Model*.

Ahsan, K.M.T., 2011. "*Implementation of Bangla Speech Recognition System on Cell Phones*", *School of Engineering and Computer Science, BRAC University*.

Carnegie Mellon University, 2016. Tersedia melalui: <http://cmusphinx.sourceforge.net/wiki/tutorialam> [Diakses 30 Januari 2016]

Carnegie Mellon University, 2016. Tersedia melalui: <http://cmusphinx.sourceforge.net/wiki/tutorial*PocketSphinx*/> [Diakses 30 Januari 2016]

Marietha, dkk. 2012, "SMS suara *Application with Automatic Speech Recognition and Text to Speech on Mobile Phone*", Jurnal Sarjana ITB bidang Teknik Elektro dan Informatika Vol.1, No.1, hal 39-43.

Monika, V., 2012, "Perancangan Program Aplikasi *Android Speech To Text* Bahasa Indonesia dan Inggris Menggunakan *Metode Hidden Markov Model*", Universitas Binus.

Munawar, B., 2010, "*Word Identification Using Hidden Markov Model (HMM) Through Feature Extraction Linear Predictive Coding (LPC)*", *Undergraduate Theses from* Perpustakaan UNIKOM.

Negara, I.G.K., 2011. "Panduan Archlinux untuk Pemula", Jurusan Informatika Wearness Education Center Bali.

Noora M., dkk. 2010. "*Speech to Video* (S2v), Sistem Komunikasi Portabel Untuk Penderita Tunarungu", Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember.

Prasetyo, M.E.B., 2010. "Teori Dasar Hidden Markov Model", Program Studi Sistem dan Teknologi Informasi, Institut Teknologi Bandung.

Suryadarma, I.K., 2014. "Perancangan Aplikasi Speech to Text Bahasa Inggris ke Bahasa Bali Menggunakan Pocketsphinx Berbasis Android", Jurusan Teknik Telekomunikasi.

Tunarungu. "Anak Tunarungu". 16 Mei 2014. Tersedia di <http://tunarungu.com> [Diakses 25 Januari 2016]

WHO, 2016. Tersedia melalui: <http://www.who.int/pbd/deafness/estimates/en/> [Diakses 25 Januari 2016]

LAMPIRAN

Lampiran 1

Source code pocket.c

```
/* pocket.c */
#include <gst/gst.h>
#include <stdbool.h>
#include <stdio.h>

#define MODELDIR "./language"
#define NATIVE_MODELDIR
"/usr/local/share/pocketsphinx/model"

static GMainLoop *loop;

static GstElement *bin,     // the containing all the
elements
           *pipeline,                  // the pipeline for
the bin
           *alsa_src,                  // the microphone
input source
           *audio_convert,  // there should always be
audioconvert and audioresample elements before the audio
sink...
           *audio_resample,       // ...since the
capabilities of the audio sink usually vary depending on the
environment (output used, sound card, driver etc.)
           *vader,                     // for threasholding
the input to the pocketsphinx
           *asr,                  // the main asr (speech to
text) engine
           *fakesink;             // a working pipe must have
a source (in this case the microphone) and a sink. using a
dummy sink.

static GstBus *bus;   //the bus element te transport
messages from/to the pipeline

//the send error method WILL send errors back to the caller
static bool send_error (const char *errorMessage)
{
     printf("\n --------- ERRROR: ----------\n%s\n",
errorMessage);
     return false;
}

static gboolean bus_call(GstBus *bus, GstMessage *msg, void
*user_data)
{
```

```
//    g_print ("Got %s message\n", GST_MESSAGE_TYPE_NAME
(msg));

      //TODO: Error handling and reporting back to main
app...
      switch (GST_MESSAGE_TYPE(msg)) {
      case GST_MESSAGE_EOS: {
            g_message("End-of-stream");
            g_main_loop_quit(loop);
            break;
      }
      case GST_MESSAGE_ERROR: {
            GError *err;
            gst_message_parse_error(msg, &err, NULL);
            g_error("%s", err->message);
            g_error_free(err);

            g_main_loop_quit(loop);
            break;
      }
      case GST_MESSAGE_APPLICATION: {
            const GstStructure *str;
            str = gst_message_get_structure (msg);
                  if
(gst_structure_has_name(str,"partial_result"))
                        printf("");
                  else if
(gst_structure_has_name(str,"result"))
                  {
                        printf("Kata terucap: %s\n",
gst_structure_get_string(str,"hyp"));
                  }
                  else if
(gst_structure_has_name(str,"result"))
                  {
                        g_main_loop_quit(loop);
                  }
            break;
      }
      default:

            break;
      }

            //printf("info: %i %s\n", (int)(msg->timestamp),
GST_MESSAGE_TYPE_NAME (msg));

      return true;
}

//turns the engine off....
void turn_off()
{
```

```
	//hmm... there's probably a better way to create a
gvalue containing a string from the gchararray...
	GValue text_gv = {0};
	g_value_init (&text_gv, G_TYPE_STRING);
	g_value_set_string(&text_gv, "turn_off");

	//creates message structure for the partial result:
	GstStructure *messageStruct = gst_structure_empty_new
("turn_off");
	//set the hypothesis (the guessed text)
	gst_structure_set_value (messageStruct, "hyp",
&text_gv);

	//post message to the pipeline bus
	if (gst_bus_post(
		gst_element_get_bus(asr),
		gst_message_new_application((GstObject *) asr,
messageStruct)))
	{
		//TODO: something if post failed...
	}
}


//the partial result method is triggered when a
comprehensive part of the audio input is transcribed
void asr_partial_result (GstElement* asr,  gchararray text,
gchararray uttid, gpointer user_data)
{
	//hmm... there's probably a better way to create a
gvalue containing a string from the gchararray...
	GValue text_gv = {0};
	g_value_init (&text_gv, G_TYPE_STRING);
	g_value_set_string(&text_gv, text);

	//creates message structure for the partial result:
	GstStructure *messageStruct = gst_structure_empty_new
("partial_result");
	//set the hypothesis (the guessed text)
	gst_structure_set_value (messageStruct, "hyp",
&text_gv);

	//post message to the pipeline bus
	if (gst_bus_post(
		gst_element_get_bus(asr),
		gst_message_new_application((GstObject *) asr,
messageStruct)))
	{
		//TODO: something if post failed...
	}
}
```

```
//the partial result method is triggered when a full part of
the audio input is transcribed
void asr_result (GstElement* asr,  gchararray text,
gchararray uttid, gpointer user_data)
{
      //hmm... there's probably a better way to create a
gvalue containing a string from the gchararray...
      GValue text_gv = {0};
      g_value_init (&text_gv, G_TYPE_STRING);
      g_value_set_string(&text_gv, text);

      //creates message structure for the partial result:
      GstStructure *messageStruct = gst_structure_empty_new
("result");
      //set the hypothesis (the guessed text)
      gst_structure_set_value (messageStruct, "hyp",
&text_gv);

      //post message to the pipeline bus
      if (gst_bus_post(
            gst_element_get_bus(asr),
            gst_message_new_application((GstObject *) asr,
messageStruct)))
      {
            //TODO: something if post failed...
      }
}

bool initElements (const char *lmFile, const char *dictFile,
const char *hmmFile)
{

      gst_init (NULL, NULL);

      // create the main loop
      loop = g_main_loop_new(NULL, FALSE);

      pipeline = gst_pipeline_new ("asr_pipeline");
      bin = gst_bin_new ("asr_bin");

      //initializing elements
      alsa_src = gst_element_factory_make ("alsasrc",
"alsa_src");
      audio_convert = gst_element_factory_make
("audioconvert", "audio_convert");
      audio_resample = gst_element_factory_make
("audioresample", "audio_resample");
      vader = gst_element_factory_make ("vader", "vader");
      asr = gst_element_factory_make ("pocketsphinx",
"asr");
      fakesink = gst_element_factory_make ("fakesink",
"fakesink");
```

```
	//check for successfull creation of elements
	if(!alsa_src)
		return send_error("Unable create alsa src
(microphone input) object!");
	if(!audio_convert || !audio_resample)
		return send_error("Unable create
converter/resampler.");
	if(!vader)
		return send_error("Unable create vader.");
	if(!asr)
		return send_error("Unable create pocketsphinx
element. Is the gstpocketsphinx installed?");
	if(!fakesink)
		return send_error("Unable create fakesink...
strange.");

	//set up the vader to auto-threshold:
	g_object_set(G_OBJECT(vader), "auto_threshold", true,
NULL);

	//set the directory containing acoustic model
parameters
	g_object_set(G_OBJECT(asr), "hmm",  hmmFile , NULL);
	//set the language model of the asr
	g_object_set(G_OBJECT(asr), "lm",  lmFile , NULL);
	//set the dictionary of the asr
	g_object_set(G_OBJECT(asr), "dict",  dictFile , NULL);
	//set the asr to be configured before receiving data
	g_object_set(G_OBJECT(asr), "configured",  true ,
NULL);

	//add the bus message methods to the asr (complete &
partial)
	g_signal_connect (asr, "partial_result", G_CALLBACK
(asr_partial_result), NULL);
	g_signal_connect (asr, "result", G_CALLBACK
(asr_result), NULL);


	// create the bus for the pipeline:
	bus = gst_pipeline_get_bus(GST_PIPELINE(pipeline));
	//add the bus handler method:
	gst_bus_add_watch(bus, bus_call, NULL);

	gst_object_unref(bus);

	// Add the elements to the bin
	gst_bin_add_many (GST_BIN (bin),
		alsa_src,
		audio_convert,
		audio_resample,
		vader,
		asr,
```

```
            fakesink,
        NULL);

        // add the bin to the pipeline
        gst_bin_add (GST_BIN (pipeline), bin);

        // link the elements and check for success
        if (!gst_element_link_many (alsa_src,
            audio_convert,
            audio_resample,
            vader,
            asr,
            fakesink,
        NULL))
            return send_error("Unable to link elements!");

        //creation successfull
        return true;
}

void play ()
{

     //traverse...
     gst_element_set_state(GST_ELEMENT(pipeline),
GST_STATE_PLAYING);

     g_main_loop_run(loop);

     gst_element_set_state(GST_ELEMENT(pipeline),
GST_STATE_NULL);
     gst_object_unref(GST_OBJECT(pipeline));
     g_main_loop_unref (loop);

}

int main (int argc, char *argv[])
{

     if (initElements (
               MODELDIR "/0389.lm",
               MODELDIR "/uji.dic",
               NATIVE_MODELDIR "/hmm/en_US/hub4wsj_sc_8k"
))
          play ();
     else
     {
          printf("Init failed...");
          return -1;
     }

     return 0;
}
```

Lampiran 2

Source sphinx_train.cfg

```
# Configuration script for sphinx trainer                    -
*-mode:Perl-*-

$CFG_VERBOSE = 1;           # Determines how much goes to
the screen.

# These are filled in at configuration time
$CFG_DB_NAME = "ttdata";
# Experiment name, will be used to name model files and log
files
$CFG_EXPTNAME = "$CFG_DB_NAME";

# Directory containing SphinxTrain binaries
$CFG_BASE_DIR = "/root/tutorial/ttdata";
$CFG_SPHINXTRAIN_DIR = "/usr/local/lib/sphinxtrain";
$CFG_BIN_DIR = "/usr/local/libexec/sphinxtrain";
$CFG_SCRIPT_DIR = "/usr/local/lib/sphinxtrain/scripts";


# Audio waveform and feature file information
$CFG_WAVFILES_DIR = "$CFG_BASE_DIR/wav";
$CFG_WAVFILE_EXTENSION = 'wav';
$CFG_WAVFILE_TYPE = 'mswav'; # one of nist, mswav, raw
$CFG_FEATFILES_DIR = "$CFG_BASE_DIR/feat";
$CFG_FEATFILE_EXTENSION = 'mfc';
$CFG_VECTOR_LENGTH = 13;

# Feature extraction parameters
$CFG_WAVFILE_SRATE = 16000.0;
$CFG_NUM_FILT = 40; # For wideband speech it's 40, for
telephone 8khz reasonable value is 31
$CFG_LO_FILT = 133.3334; # For telephone 8kHz speech value
is 200
$CFG_HI_FILT = 6855.4976; # For telephone 8kHz speech value
is 3500

$CFG_MIN_ITERATIONS = 1;  # BW Iterate at least this many
times
$CFG_MAX_ITERATIONS = 10; # BW Don't iterate more than this,
somethings likely wrong.

# (none/max) Type of AGC to apply to input files
$CFG_AGC = 'none';
# (current/none) Type of cepstral mean
subtraction/normalization
# to apply to input files
$CFG_CMN = 'current';
# (yes/no) Normalize variance of input files to 1.0
$CFG_VARNORM = 'no';
# (yes/no) Train full covariance matrices
```

```
$CFG_FULLVAR = 'no';
# (yes/no) Use diagonals only of full covariance matrices
for
# Forward-Backward evaluation (recommended if CFG_FULLVAR is
yes)
$CFG_DIAGFULL = 'no';

# (yes/no) Perform vocal tract length normalization in
training.  This
# will result in a "normalized" model which requires VTLN to
be done
# during decoding as well.
$CFG_VTLN = 'no';
# Starting warp factor for VTLN
$CFG_VTLN_START = 0.80;
# Ending warp factor for VTLN
$CFG_VTLN_END = 1.40;
# Step size of warping factors
$CFG_VTLN_STEP = 0.05;

# Directory to write queue manager logs to
$CFG_QMGR_DIR = "$CFG_BASE_DIR/qmanager";
# Directory to write training logs to
$CFG_LOG_DIR = "$CFG_BASE_DIR/logdir";
# Directory for re-estimation counts
$CFG_BWACCUM_DIR = "$CFG_BASE_DIR/bwaccumdir";
# Directory to write model parameter files to
$CFG_MODEL_DIR = "$CFG_BASE_DIR/model_parameters";

# Directory containing transcripts and control files for
# speaker-adaptive training
$CFG_LIST_DIR = "$CFG_BASE_DIR/etc";

# Decoding variables for MMIE training
$CFG_LANGUAGEWEIGHT = "11.5";
$CFG_BEAMWIDTH      = "1e-100";
$CFG_WORDBEAM       = "1e-80";
$CFG_LANGUAGEMODEL  = "$CFG_LIST_DIR/$CFG_DB_NAME.lm.DMP";
$CFG_WORDPENALTY    = "0.2";

# Lattice pruning variables
$CFG_ABEAM              = "1e-50";
$CFG_NBEAM              = "1e-10";
$CFG_PRUNED_DENLAT_DIR  = "$CFG_BASE_DIR/pruned_denlat";

# MMIE training related variables
$CFG_MMIE = "no";
$CFG_MMIE_MAX_ITERATIONS = 5;
$CFG_LATTICE_DIR = "$CFG_BASE_DIR/lattice";
$CFG_MMIE_TYPE   = "rand"; # Valid values are "rand", "best"
or "ci"
$CFG_MMIE_CONSTE = "3.0";
$CFG_NUMLAT_DIR  = "$CFG_BASE_DIR/numlat";
```

```
$CFG_DENLAT_DIR  = "$CFG_BASE_DIR/denlat";

# Variables used in main training of models
$CFG_DICTIONARY     = "$CFG_LIST_DIR/$CFG_DB_NAME.dic";
$CFG_RAWPHONEFILE   = "$CFG_LIST_DIR/$CFG_DB_NAME.phone";
$CFG_FILLERDICT     = "$CFG_LIST_DIR/$CFG_DB_NAME.filler";
$CFG_LISTOFFILES    =
"$CFG_LIST_DIR/${CFG_DB_NAME}_train.fileids";
$CFG_TRANSCRIPTFILE =
"$CFG_LIST_DIR/${CFG_DB_NAME}_train.transcription";
$CFG_FEATPARAMS     = "$CFG_LIST_DIR/feat.params";

# Variables used in characterizing models

$CFG_HMM_TYPE = '.cont.'; # Sphinx 4, PocketSphinx
#$CFG_HMM_TYPE  = '.semi.'; # PocketSphinx
#$CFG_HMM_TYPE  = '.ptm.'; # PocketSphinx (larger data sets)

if (($CFG_HMM_TYPE ne ".semi.")
    and ($CFG_HMM_TYPE ne ".ptm.")
    and ($CFG_HMM_TYPE ne ".cont.")) {
  die "Please choose one CFG_HMM_TYPE out of '.cont.',
'.ptm.', or '.semi.', " .
    "currently $CFG_HMM_TYPE\n";
}

# This configuration is fastest and best for most acoustic
models in
# PocketSphinx and Sphinx-III.  See below for Sphinx-II.
$CFG_STATESPERHMM = 3;
$CFG_SKIPSTATE = 'no';

if ($CFG_HMM_TYPE eq '.semi.') {
  $CFG_DIRLABEL = 'semi';
# Four stream features for PocketSphinx
  $CFG_FEATURE = "s2_4x";
  $CFG_NUM_STREAMS = 4;
  $CFG_INITIAL_NUM_DENSITIES = 256;
  $CFG_FINAL_NUM_DENSITIES = 256;
  die "For semi continuous models, the initial and final
models have the same density"
    if ($CFG_INITIAL_NUM_DENSITIES !=
$CFG_FINAL_NUM_DENSITIES);
} elsif ($CFG_HMM_TYPE eq '.ptm.') {
  $CFG_DIRLABEL = 'ptm';
# Four stream features for PocketSphinx
  $CFG_FEATURE = "s2_4x";
  $CFG_NUM_STREAMS = 4;
  $CFG_INITIAL_NUM_DENSITIES = 64;
  $CFG_FINAL_NUM_DENSITIES = 64;
  die "For phonetically tied models, the initial and final
models have the same density"
```

```
    if ($CFG_INITIAL_NUM_DENSITIES !=
$CFG_FINAL_NUM_DENSITIES);
} elsif ($CFG_HMM_TYPE eq '.cont.') {
  $CFG_DIRLABEL = 'cont';
# Single stream features - Sphinx 3
  $CFG_FEATURE = "1s_c_d_dd";
  $CFG_NUM_STREAMS = 1;
  $CFG_INITIAL_NUM_DENSITIES = 1;
  $CFG_FINAL_NUM_DENSITIES = 8;
  die "The initial has to be less than the final number of
densities"
    if ($CFG_INITIAL_NUM_DENSITIES >
$CFG_FINAL_NUM_DENSITIES);
}

# Number of top gaussians to score a frame. A little bit
less accurate computations
# make training significantly faster. Uncomment to apply
this during the training
# For good accuracy make sure you are using the same setting
in decoder
# In theory this can be different for various training
stages. For example 4 for
# CI stage and 16 for CD stage
# $CFG_CI_TOPN = 4;
# $CFG_CD_TOPN = 16;

# (yes/no) Train multiple-gaussian context-independent
models (useful
# for alignment, use 'no' otherwise) in the models created
# specifically for forced alignment
$CFG_FALIGN_CI_MGAU = 'no';
# (yes/no) Train multiple-gaussian context-independent
models (useful
# for alignment, use 'no' otherwise)
$CFG_CI_MGAU = 'no';
# Number of tied states (senones) to create in decision-tree
clustering
$CFG_N_TIED_STATES = 200;
# How many parts to run Forward-Backward estimatinon in
$CFG_NPART = 1;

# (yes/no) Train a single decision tree for all phones
(actually one
# per state) (useful for grapheme-based models, use 'no'
otherwise)
$CFG_CROSS_PHONE_TREES = 'no';

# Use force-aligned transcripts (if available) as input to
training
$CFG_FORCEDALIGN = 'no';
```

```
# Use a specific set of models for force alignment.  If not
defined,
# context-independent models for the current experiment will
be used.
$CFG_FORCE_ALIGN_MDEF =
"$CFG_BASE_DIR/model_architecture/$CFG_EXPTNAME.falign_ci.md
ef";
$CFG_FORCE_ALIGN_MODELDIR =
"$CFG_MODEL_DIR/$CFG_EXPTNAME.falign_ci_$CFG_DIRLABEL";

# Use a specific dictionary and filler dictionary for force
alignment.
# If these are not defined, a dictionary and filler
dictionary will be
# created from $CFG_DICTIONARY and $CFG_FILLERDICT, with
noise words
# removed from the filler dictionary and added to the
dictionary (this
# is because the force alignment is not very good at
inserting them)

# $CFG_FORCE_ALIGN_DICTIONARY =
"$ST::CFG_BASE_DIR/falignout$ST::CFG_EXPTNAME.falign.dict";;
# $CFG_FORCE_ALIGN_FILLERDICT =
"$ST::CFG_BASE_DIR/falignout/$ST::CFG_EXPTNAME.falign.fdict"
;;

# Use a particular beam width for force alignment.  The
wider
# (i.e. smaller numerically) the beam, the fewer sentences
will be
# rejected for bad alignment.
$CFG_FORCE_ALIGN_BEAM = 1e-60;

# Calculate an LDA/MLLT transform?
$CFG_LDA_MLLT = 'no';
# Dimensionality of LDA/MLLT output
$CFG_LDA_DIMENSION = 29;

# This is actually just a difference in log space (it
doesn't make
# sense otherwise, because different feature parameters have
very
# different likelihoods)
$CFG_CONVERGENCE_RATIO = 0.1;

# Queue::POSIX for multiple CPUs on a local machine
# Queue::PBS to use a PBS/TORQUE queue
$CFG_QUEUE_TYPE = "Queue";

# Name of queue to use for PBS/TORQUE
$CFG_QUEUE_NAME = "workq";
```

```
# (yes/no) Build questions for decision tree clustering
automatically
$CFG_MAKE_QUESTS = "yes";
# If CFG_MAKE_QUESTS is yes, questions are written to this
file.
# If CFG_MAKE_QUESTS is no, questions are read from this
file.
$CFG_QUESTION_SET =
"${CFG_BASE_DIR}/model_architecture/${CFG_EXPTNAME}.tree_que
stions";
#$CFG_QUESTION_SET = "${CFG_BASE_DIR}/linguistic_questions";

$CFG_CP_OPERATION =
"${CFG_BASE_DIR}/model_architecture/${CFG_EXPTNAME}.cpmeanva
r";

# Configuration for grapheme-to-phoneme model
$CFG_G2P_MODEL= 'no';

# Configuration script for sphinx decoder

# Variables starting with $DEC_CFG_ refer to decoder
specific
# arguments, those starting with $CFG_ refer to trainer
arguments,
# some of them also used by the decoder.

$DEC_CFG_VERBOSE = 1;     # Determines how much goes to
the screen.

# These are filled in at configuration time

# Name of the decoding script to use (psdecode.pl or
s3decode.pl, probably)
$DEC_CFG_SCRIPT = 'psdecode.pl';

$DEC_CFG_EXPTNAME = "$CFG_EXPTNAME";
$DEC_CFG_JOBNAME  = "$CFG_EXPTNAME"."_job";

# Models to use.
$DEC_CFG_MODEL_NAME =
"$CFG_EXPTNAME.cd_${CFG_DIRLABEL}_${CFG_N_TIED_STATES}";

$DEC_CFG_FEATFILES_DIR = "$CFG_BASE_DIR/feat";
$DEC_CFG_FEATFILE_EXTENSION = '.mfc';
$DEC_CFG_VECTOR_LENGTH = $CFG_VECTOR_LENGTH;
$DEC_CFG_AGC = $CFG_AGC;
$DEC_CFG_CMN = $CFG_CMN;
$DEC_CFG_VARNORM = $CFG_VARNORM;

$DEC_CFG_QMGR_DIR = "$CFG_BASE_DIR/qmanager";
$DEC_CFG_LOG_DIR = "$CFG_BASE_DIR/logdir";
$DEC_CFG_MODEL_DIR = "$CFG_MODEL_DIR";
```

```
$DEC_CFG_DICTIONARY     =
"$CFG_BASE_DIR/etc/$CFG_DB_NAME.dic";
$DEC_CFG_FILLERDICT     =
"$CFG_BASE_DIR/etc/$CFG_DB_NAME.filler";
$DEC_CFG_LISTOFFILES    =
"$CFG_BASE_DIR/etc/${CFG_DB_NAME}_test.fileids";
$DEC_CFG_TRANSCRIPTFILE =
"$CFG_BASE_DIR/etc/${CFG_DB_NAME}_test.transcription";
$DEC_CFG_RESULT_DIR     = "$CFG_BASE_DIR/result";

# This variables, used by the decoder, have to be user
defined, and
# may affect the decoder output

$DEC_CFG_LANGUAGEMODEL  =
"$CFG_BASE_DIR/etc/${CFG_DB_NAME}.lm.DMP";
$DEC_CFG_LANGUAGEWEIGHT = "10";
$DEC_CFG_BEAMWIDTH = "1e-80";
$DEC_CFG_WORDBEAM = "1e-40";

$DEC_CFG_ALIGN = "builtin";

$DEC_CFG_NPART = 1;         #  Define how many pieces to
split decode in

# This variable has to be defined, otherwise utils.pl will
not load.
$CFG_DONE = 1;

return 1;
```

Lampiran 3

The ARPAbet

Vowels		
Phoneme	**Example**	**Transcription**
AA	bot	B AA T
AE	bat	B AE T
AH	but	B AH T
AO	bought	B AO T
AW	bout	B AW T
AY	bite	B AY T
EH	bet	B EH T
ER	bird	B ER D
EY	bait	B EY T
IH	bit	B IH T
IY	beat	B IY T
OW	boat	B OW T
OY	boy	B OY
UH	put	P UH T
UW	boot	B UW T

Consonants		
Phoneme	**Example**	**Transcription**
B	be	B IY
CH	cheese	CH IY Z
D	day	D EY
DH	that	TH AE T
F	fee	F IY
G	go	G OW
HH	he	HH IY
JH	just	JH AH S T
K	key	K IY
L	late	L EY T
M	me	M IY
N	knee	N IY
NG	sing	S IH NG
P	pay	P EY
R	read	R IY D
S	sea	S IY
SH	she	SH IY
T	tea	T IY
TH	thanks	TH AE NG K S
V	vain	V EY N
W	we	W IY
Y	yes	Y EH S
Z	zoo	Z UW
ZH	pleasure	P L EH ZH ER

Stress ini mempunyai beberapa tingkatan level dalam pengucapan di ARPAbet

Value	Level of stress
0	no stress
1	primary stress
2	secondary stress

Contoh:

in	AH0 N, IH1 N
the	DH AH0, DH AH1, DH IY0
dictionary	D IH1 K SH AH0 N EH2 R IY0
stress	S T R EH1 S
is	IH1 Z, AH0 Z
indicated	IH1 N D AH0 K EY2 T AH0 D
by	B AY1
digits	D IH1 JH AH0 T S
following	F AA1 L OW0 IH0 NG
stressed	S T R EH1 S T
vowels	V AW1 AH0 L Z

Sumber: The CMU Pronouncing Dictionary - Speech at CMU (http://www.speech.cs.cmu.edu/cgi-bin/cmudict)

Lampiran 4

1. Tabel Hasil Pengujian Jarak dengan TOA Mikrofon Jepit

   a. Jarak 1 Meter

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Akurasi
P1	O	O	X	O	O	O	O	O	O	O	90%
P2	O	O	O	O	O	O	O	O	O	O	100%
P3	O	O	O	O	O	O	O	O	O	O	100%
P4	O	O	O	X	O	O	O	O	O	O	90%
P5	O	X	X	O	O	O	O	O	O	O	80%
L1	X	O	O	O	O	O	O	O	O	O	90%
L2	O	X	O	O	O	O	O	O	O	O	90%
L3	O	O	O	O	O	O	O	O	O	X	90%
L4	O	O	O	O	O	O	O	O	O	O	100%
L5	O	O	O	O	O	O	O	O	O	O	100%
	Rata-rata Akurasi										93%

Keterangan:
Pn: Koresponden Perempuan ke-n
Ln: Koresponden Laki-laki ke-n
Tn: Tes pengujian sistem ke-n
O: Berhasil dideteksi
X: Tidak berhasil dideteksi

b. Jarak 2 Meter

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Akurasi
P1	X	X	O	O	O	X	O	O	X	O	60%
P2	X	O	O	O	O	O	O	X	O	O	80%
P3	X	O	O	O	O	O	X	O	O	O	80%
P4	X	O	O	X	O	O	O	O	O	O	80%
P5	X	O	O	O	O	O	O	O	O	X	80%
L1	X	X	X	O	X	O	O	O	O	O	60%
L2	X	X	O	O	O	O	O	O	X	O	70%
L3	X	O	O	O	O	O	O	O	O	X	80%
L4	X	X	O	X	O	X	O	O	O	O	60%
L5	X	X	O	O	O	O	O	X	O	X	60%
	Rata-rata Akurasi										71%

Keterangan:
Pn: Koresponden Perempuan ke-n
Ln: Koresponden Laki-laki ke-n
Tn: Tes pengujian sistem ke-n
O: Berhasil dideteksi
X: Tidak berhasil dideteksi

2. Tabel Hasil Pengujian Mikrofon

a. Sony *Mini Microphone*

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Akurasi
P1	X	X	O	O	O	O	O	O	O	O	80%
P2	O	X	O	O	O	O	O	O	O	O	90%
P3	X	O	O	O	O	O	O	O	O	O	90%
P4	X	O	O	X	O	O	O	O	O	O	80%
P5	X	O	O	O	O	O	O	O	O	O	90%
L1	X	O	O	O	O	O	O	O	X	O	80%
L2	O	X	O	O	O	O	O	O	O	O	90%
L3	X	O	O	O	O	O	O	O	O	X	80%
L4	O	O	O	O	O	O	O	O	O	O	100%
L5	O	X	O	O	O	O	O	X	O	X	70%
	Rata-rata Akurasi										85%

Keterangan:
Pn: Koresponden Perempuan ke-n
Ln: Koresponden Laki-laki ke-n
Tn: Tes pengujian sistem ke-n
O: Berhasil dideteksi
X: Tidak berhasil dideteksi

b. TOA Mikrofon Jepit

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Akurasi
P1	O	O	X	O	O	O	O	O	O	O	90%
P2	O	O	O	O	O	O	O	O	O	O	100%
P3	O	O	O	O	O	O	O	O	O	O	100%
P4	O	O	O	X	O	O	O	O	O	O	90%
P5	O	X	X	O	O	O	O	O	O	O	80%
L1	X	O	O	O	O	O	O	O	O	O	90%
L2	O	X	O	O	O	O	O	O	O	O	90%
L3	O	O	O	O	O	O	O	O	O	X	90%
L4	O	O	O	O	O	O	O	O	O	O	100%
L5	O	O	O	O	O	O	O	O	O	O	100%
Rata-rata Tingkat Keberhasilan											93%

Keterangan:
Pn: Koresponden Perempuan ke-n
Ln: Koresponden Laki-laki ke-n
Tn: Tes pengujian sistem ke-n
O: Berhasil dideteksi
X: Tidak berhasil dideteksi

3. Tabel Pengujian Uji Pengenalan Suara

a. Pengujian Nama Koresponden Perempuan

Ira	Roni	Tama	Wildan	Embris	Szczesny	Grobkreutz	Blaszczykowski	Schweinsteiger	Schwarzenegger
90%	80%	100%	80%	30%	70%	30%	10%	0%	0%
100%	90%	100%	60%	60%	50%	10%	0%	0%	0%
90%	80%	90%	90%	50%	40%	10%	10%	10%	0%
90%	100%	80%	90%	20%	20%	20%	0%	0%	0%
90%	80%	90%	100%	60%	50%	0%	0%	0%	0%

b. Pengujian Nama Koresponden Laki-laki

Ira	Roni	Tama	Wildan	Embris	Szczesny	Grobkreutz	Blaszczykowski	Schweinsteiger	Schwarzenegger
100%	90%	90%	60%	70%	80%	20%	0%	0%	0%
100%	90%	100%	90%	80%	90%	0%	0%	0%	10%
80%	80%	80%	80%	70%	100%	20%	0%	0%	0%
90%	80%	90%	80%	80%	60%	0%	0%	0%	0%
100%	70%	100%	80%	70%	90%	10%	0%	0%	0%