Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Programa de Pós-Graduação em Engenharia Elétrica

Victor Costa de Andrade Pimentel

Plataforma embarcada de reconhecimento automático da fala para o auxílio de pessoas com mobilidade reduzida

João Pessoa/PB
Fevereiro, 2016

Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Programa de Pós-Graduação em Engenharia Elétrica

Victor Costa de Andrade Pimentel

Plataforma embarcada de reconhecimento automático da fala para o auxílio de pessoas com mobilidade reduzida

Dissertação de mestrado submetida à Coordenação do Programa de Pós-Graduação em Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba como requisito necessário para obtenção do grau de Mestre em Ciências no domínio da Engenharia Elétrica.

Orientadora: Profª. Drª. Suzete Élida Nóbrega Correia
Coorientadora: Profª. Drª. Silvana Luciene do Nascimento Cunha Costa

João Pessoa/PB
Fevereiro, 2016

Dados Internacionais de Catalogação na Publicação – CIP
Biblioteca Nilo Peçanha – IFPB, *campus* João Pessoa

P644p Pimentel, Victor Costa de Andrade.
Plataforma embarcada de reconhecimento automático da fala para o auxílio de pessoas com mobilidade reduzida. – 2016.
239 f. : il.
Dissertação (Mestrado em Engenharia Elétrica) – Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB / Coordenação de Pós-Graduação em Engenharia Elétrica, 2016.
Orientador: Profª Drª. Suzete Élida Nóbrega Correia
Coorientadora: Profª. Drª. Silvana L. N. Cunha Costa

1. Automação. 2. Controle automático. 3. Tecnologia assistiva I. Título.

CDU 681.5

Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Programa de Pós-Graduação em Engenharia Elétrica

Plataforma embarcada de reconhecimento automático da fala para o auxílio de pessoas com mobilidade reduzida

Victor Costa de Andrade Pimentel

Trabalho aprovado. João Pessoa/PB, 29 de Fevereiro de 2016:

Profª. Drª. Suzete Élida Nóbrega Correia
Orientadora (IFPB-JP)

Profª. Drª. Silvana Luciene do Nascimento Cunha Costa
Coorientadora (IFPB-JP)

Profº Dr. Cleumar Moreira da Silva
Membro da Banca (IFPB-JP)

Profª Dra. Daniella Dias Cavalcante da Silva
Membro da Banca (IFPB-CG)

Profº Dr. Ilton Luiz Barbacena
Membro da Banca (IFPB-JP)

João Pessoa/PB
Fevereiro, 2016

Dedico esse trabalho a Deus e a meus pais, Fernando e Vânia.

Agradecimentos

Agradeço primeiramente a Deus, quem criou todas as coisas, pelos dons do Espírito Santo que nos sustenta na perseverante busca do Caminho, da Verdade e da Vida.

A meus pais, Fernando e Vânia, por terem me proporcionado a educação essencial ao longo de minha vida, através de valores de honestidade e humildade, e, juntamente com minha irmã Thaís, pela sua paciência, compreensão e incentivo.

À Mariana, por seu cuidado, sua presença, compreensão e incentivo.

À professora Dra. Suzete Élida Nóbrega Correia pela imensa paciência, confiança e compreensão durante a orientação que me proporcionou ao longo do desenvolvimento desse trabalho, e pelos valorosos ensinamentos no âmbito técnico, acadêmico, profissional e humano.

À professora Dra. Silvana Luciene do Nascimento Cunha Costa, por ter aceito coorientar esse trabalho, por seu incentivo e motivação, trazendo importantes contribuições à pesquisa no âmbito do processamento digital de sinais da fala.

Ao professor Msc. Michel Coura Dias, presente colaborador desse projeto, contribuindo sempre com novas ideias para a integração do controle através do uso de redes convergentes e Voz sobre IP, trazendo a macrovisão de um projeto conectado à Internet das Coisas.

Ao professor Dr. Ilton Luiz Barbacena, pelas contribuições na área de Sistemas Embarcados e pelo incentivo em inovar.

Aos demais avaliadores da banca, pelas importantes contribuições dadas à pesquisa.

Aos participantes da pesquisa, que generosa e pacientemente doaram um pouco de seu tempo e da sua voz para a realização dos testes do projeto.

Ao IFPB e à Coordenação do Programa de Pós-Graduação em Engenharia Elétrica do IFPB, nas pessoas dos professores Dr. Jefferson Costa e Silva e Dra. Silvana Luciene do Nascimento Cunha Costa, que estiveram à frente da coordenação do programa durante o tempo em que realizei o curso.

Aos colegas e professores do curso de Mestrado pelos momentos e conhecimento compartilhados no decorrer dessa caminhada.

"Vós, irmãos, não vos canseis de fazer o bem."
(2Ts 3,13)

Resumo

A busca por maior independência e autonomia para as pessoas com deficiência tem se apresentado como um fator decisivo ao proporcionar uma melhoria na qualidade de vida desses indivíduos através do uso de tecnologias assistivas. A fala se constitui na mais básica, comum e eficiente forma de comunicação entre os seres humanos, de modo que a entrada de comandos por voz pode ser uma alternativa para que pessoas com mobilidade reduzida, e que tenham preservada boa capacidade das habilidades da fala, realizem o controle do computador ou outros dispositivos. O objetivo deste trabalho consiste no desenvolvimento de uma interface de comandos por voz, através do reconhecimento automático da fala, que seja facilmente adaptada e incorporada a sistemas e ferramentas de auxílio ao controle do ambiente doméstico (domótica). Com esse intuito, foram executadas duas abordagens de desenvolvimento. A primeira consistiu de um experimento piloto realizado com o intuito de formar uma base inicial de conhecimento no desenvolvimento de aplicações utilizando o reconhecimento de comandos por voz. Esta etapa baseou-se na utilização de um módulo de *hardware* específico, que recebe os comando de voz diretamente através de um microfone, constituindo-se de um sistema dependente de locutor capaz de reconhecer comandos de palavras isoladas para o controle das luzes de um LED RGB. Já a segunda abordagem, integra componentes de *hardware* aberto e *software* livre e de código aberto, sendo os comandos de voz fornecidos ao sistema através de um *smartphone* configurado com *softphone* VoIP (Voz sobre IP). Nesse último caso, o *softphone*, então, se registra no servidor de comunicação Asterisk, que implementa uma central telefônica com unidade de resposta audível (URA). Integrada ao servidor, está a ferramenta de reconhecimento da fala, Julius. Esses componentes estão embarcados na plataforma Beaglebone Black, de baixo custo. O sistema é dependente de locutor e capaz de reconhecer frases com três palavras para o controle da iluminação, televisão e acesso a portas de um ambiente doméstico hipotético constituído de sala, cozinha, quarto, banheiro e área externa. Os resultados obtidos a partir dos testes realizados indicam taxas de acerto de 95,9% e 94,77% para as interfaces desenvolvidas na primeira e segunda abordagens, respectivamente. Esses índices sugerem que é viável o emprego dos módulos de reconhecimento desenvolvidos na implementação de soluções de tecnologias assistivas.

Palavras-chave: Tecnologias assistivas. Reconhecimento automático da fala. Sistemas embarcados. Automação residencial. Asterisk.

Abstract

The quest to achieve greater independence and autonomy for people with disabilities has emerged as a decisive factor in providing a better quality of life for these individuals through the use of assistive technologies. Once speech constitutes the most basic, common and efficient way of communication between human beings, the attachment of a voice control input can be a convenient alternative for people with reduced mobility, and who have stable speech skills, to control the computer and other devices. The goal of this work focuses on developing a voice command interface, through automatic speech recognition, which can easily be adapted and incorporated to systems and support tools for home environment control (domotic). To do so, two development approaches were performed. The first consists of a startup experiment conducted in order to form an initial knowledge base on developing speech recognition applications. This experiment was based on a specific hardware module that receives the voice commands through a microphone directly. It constitutes a speaker dependent system capable of recognize isolated words commands to control a RGB LED. Meanwhile, the second approach integrates open-source software and open-hardware components, so that the voice command input is provided to the system through a smartphone configured with a VoIP (Voice over IP) softphone. The softphone will then log into the communication server Asterisk, which implements an IVR (Interative Voice Response) PBX (Private Branch Exchange). Integrated to the server, there is the Julius speech recognition engine. These components are embeded in the Beaglebone Black low-cost platform. The system is speaker dependent and capable of recognizing phrases with three words for lighting, TV and access control in a hypothetical home environment consisting of hall, kitchen, chamber, bathroom and outdoor area. The results obtained from the tests indicates 95.9% and 94.77% of success rate for the developed interfaces on the first and second approaches, respectively. These rates suggest the viability of usage of the developed interfaces on implementing assistive technologies solutions.

Keywords: Assistive technology. Automatic speech recognition. Embedded systems. Home automation. Asterisk.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

AAL	*Ambient Assisted Living*
ANN	*Artificial Neural Network*
API	*Application Programming Interface*
ARM	*Advanced RISC Machine*
ASR	*Automatic Speech Recognition*
AT	*Assistive Technology*
ATK	*Real-Time API for HTK*
AT&T	*American Telephone and Telegraph*
BBB	*Beaglebone Black*
BBN	Bolt, Beranek e Newman *Technologies*
BCI	*Brain Communication Interfaces*
CALL	*Computer-assisted Language Learning*
CAT	Comitê de Ajudas Técnicas
CBEB	Congresso Brasileiro de Engenharia Biomédica
CEP	Comitê de Ética em Pesquisa
CMN	*Cepstral Mean Normalization*
CMOS	*Complementary metal-oxide-semiconductor*
CMU	Carnegie Mellon *University*
CNRTA	Centro Nacional de Referência em Tecnologia Assistiva
CNPTA	Catálogo Nacional de Produtos de Tecnologia Assistiva
CODEC	Codificador/Decodificador
CPU	*Central Processing Unit*
CSR	*Continuous Speech Recognition*
CUED	*Cambridge University Engineering Department*

DARPA	*Defense Advanced Research Projects Agency*
DHCP	*Dynamic Host Configuration Protocol*
DICT	Dicionário Fonético
DNN	*Deep Neural Network*
DNS	*Domain Name System*
DSP	*Digital Signal Processor*
EAGI	*Enhanced Asterisk Gateway Interface*
ECG	Eletrocardiograma
EEG	Eletroencefalográfico
EHS	*European Home System*
EMG	Eletromiográfico
eMMC	*Embedded MultiMedia Card*
FAQ	*Frequently Asked Questions*
FAT	*Fat Allocation Table*
FFT	*Fast Fourier Transform*
FPGA	*Field Programmable Gate Array*
G2P	*Grapheme to Phoneme*
GMM	*Gaussian Mixture Models*
GND	*Ground*
GPIO	*General Purpose Input/Output*
GSM	*Global System for Mobile*
HDMI	*High-Definition Multimedia Interface*
HMM	*Hidden Markov Model*
HTK	*Hidden Markov Model Toolkit*
IBGE	Instituto Brasileiro de Geografia e Estatística
IETF	*Internet Engineering Task Force*

IFPB	Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
IHC	Interação Humano-Computador
IoT	*Internet of Things*
IPA	*Information-technology Promotion Agency*, Japão
IRC	*Internet Relay Chat*
IVR	*Interative Voice Response*
JSAPI	*Java Speech Application Programming Interface*
KDE	*K Desktop Environment*
LED	*Light Emitting Diode*
LPC	*Linear Predictive Coding*
LaPS	*Laboratório de Processamento de Sinais*
LSI	*Large-Scale Integration*
LTCON	Laboratório de Telecomunicações e Redes Convergentes
LVCSR	*Large Vocabulary Continuous Speech Recognition*
MAC	*Media Access Control*
MDF	*Medium-Density Fiberboard*
MFCC	*Mel Frequency Cepstral Coefficient*
MIT	*Massachusetts Institute of Technology*
MRCP	*Media Resource Control Protocol*
NIST	*National Institute of Standards and Technology*
NTP	*Network Time Protocol*
OGI	*Oregon Graduate Institute*
OSI	*Open Source Initiative*
PBX	*Private Branch Exchange*
PC	*Personal Computer*
PCI	Placa de Circuito Impresso

PIC	*Peripheral Interface Controller*
RAF	Reconhecimento Automático da Fala
RESNA	*Rehabilitation Engineering and Assistive Technology Society of North America*
RGB	*Red, Green and Blue*
RISC	*Reduced Instruction Set Computer*
RXD	*Receive Data*
SAPI	*Speech Application Programming Interface*
SD	*Secure Digital Card*
SDC	*Systems Development Corporation*
SIP	*Session Initiation Protocol*
SRAM	*Static Random Access Memory*
SRE	*Speech Recognition Engine*
SRI	*Stanford Research Institute*
SSH	*Secure Shell*
TA	Tecnologia Assistiva
TCLE	Termo de Consentimento Livre e Esclarecido
TDM	*Time-Division Multiplexing*
TECI	*Thomson Electronics Computer Interface*
TTL	*Transistor-Transistor Logic*
TXD	*Transmit Data*
UART	*Universal asynchronous receiver/transmitter*
UFRGS	Universidade Federal do Rio Grande do Sul
URA	Unidade de Resposta Audível
USB	*Universal Serial Bus*
VoIP	*Voice over Internet Protocol*
VORERO	*Voice Recognition Robust*

XML	*eXtensible Markup Language*
WAN	*Wide Area Network*
WAV	*Waveform Audio Format File*
WBSN	*Wireless Body Sensor Network*
WHAS	*Wireless Home Automation System*
WPAN	*Wireless Personal Area Networks*

Lista de símbolos

s_n	Enésima amostra numa janela ou fluxo de dados de áudio
k	Coeficiente de pré-ênfase
f	Frequência
$Mel(f)$	Frequência na escala Mel
m_j	Magnitude do j-ésimo coeficiente de um canal do banco de filtros triangulares
N	Número de canais do banco de filtros
L	Valor do deslocamento para reescalonamento dos coeficientes cepstrais
c_n	Enésimo coeficiente cepstral calculado
c'_n	Enésimo coeficiente cepstral deslocado
C_0	Energia do MFCC c_0
d_t	Coeficiente delta computado num tempo t
t	Tempo
Θ	Tamanho da janela delta
$c_{t-\Theta}$	Coeficiente estático da janela delta anterior
$c_{t+\Theta}$	Coeficiente estático da próxima janela delta
$b_j(\mathbf{o}_t)$	Distribuição Gaussiana observável de probabilidades
$\mathbf{o}_t$	Observação no tempo t
a_{ij}	Transição provável associada ao par de estados ij
S	Quantidade de fluxos independentes que compõe um vetor de observação
$\mathbf{o}_{st}$	Fluxo de dados de observação independente
M_{js}	Número de componentes da mistura gaussiana no estado j para o fluxo s
c_{jsm}	Peso do *m-ésimo* componente da mistura gaussiana no estado j para o fluxo s

$\mathcal{N}\left(\cdot;\boldsymbol{\mu},\boldsymbol{\Sigma}\right)$	Gaussiana multivariada com vetor de médias $\boldsymbol{\mu}$ e matriz de covariância $\boldsymbol{\Sigma}$
$\boldsymbol{\mu}_{jsm}$	*M-ésimo* vetor de médias da Gaussiana no estado j para o fluxo s
$\boldsymbol{\Sigma}_{jsm}$	*M-ésima* matriz de covariância da Gaussiana no estado j para o fluxo s
n	Dimensionalidade do vetor de observação
γ_s	Ganho do fluxo de dados de observação

Sumário

1 Introdução

As mais recentes conquistas tecnológicas nas áreas da informação e comunicação vêm causando profundas mudanças na relação entre os usuários e os sistemas de computador. A cada dia, visando melhorar a qualidade de vida das pessoas, surgem novos conceitos e paradigmas que determinam o predomínio das máquinas e dispositivos eletroeletrônicos em praticamente todos os cenários do cotidiano (SILVA, 2011; YANG et al., 2014; SEBESTYEN et al., 2014).

Esses conceitos geram um forte impacto no desenvolvimento de novas tecnologias que permitam a integração e intercomunicação entre diferentes dispositivos, ou simplesmente "coisas", através da Internet, caracterizando soluções computacionais ubíquas e pervasivas, associadas à concepção de ambientes inteligentes. Emerge daí uma ampla gama de aplicações potenciais que tem ganhado significativa atenção por parte da indústria e do meio acadêmico (PERERA et al., 2014), dentre as quais as tecnologias assistivas (TA), em diversas modalidades, são apenas uma parte.

A computação ubíqua agrega a habilidade de comunicação entre esses diversos objetos, em qualquer lugar, a qualquer momento. A pervasividade significa o melhoramento desses objetos, adicionando-lhes capacidade de processamento, de modo que o ambiente ao redor responde como um sistema computacional. Finalmente, a inteligência dos ambientes se configura na capacidade desses objetos em registrar mudanças no ambiente físico e interagir ativamente numa situação ou processo (DOHR et al., 2010).

Nesse contexto, a Internet das Coisas (do termo inglês *Internet of Things* – IoT), abarca soluções em *eHealth*, sistemas de *home healthcare*, *elderly aid* e *ambient assisted living*[1] *(AAL)*, que apresentam uma relação muito próxima aos conceitos mencionados, podendo agregar ainda funcionalidades de controle e automação num ambiente doméstico, constituindo, por exemplo, residências inteligentes.

Em meio a esse mundo de integração tecnológica e interconectividade, não se tem medido esforços no sentido de proporcionar modalidades de entrada através das quais os humanos interajam da maneira mais simples e natural possível com as máquinas. Aqui observa-se o "... crescente interesse por *softwares* e equipamentos que compreendam, reconheçam e simulem a voz humana..."(SILVA, 2011).

A popularização dessa forma de interação com as máquinas envolve a disponibilização de recursos para o desenvolvimento de interfaces de voz acessíveis e portáveis, que possam ser facilmente adaptadas às mais diversas aplicações (BATISTA, 2013), inclusive, integrando-se a poderosos servidores de comunicação (DIAS; LUCENA; SANTOS, 2014).

[1] DOHR, A. et al. The Internet of Things for Ambient Assisted Living. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY, 7., 2010, Estados Unidos. *Proceedings...* Las Vegas: IEEE, 2010.

Foram implementadas neste trabalho interfaces de comandos por voz para sistemas embarcados, flexíveis e modulares, alinhadas ao contexto da IoT, e que auxiliem indivíduos idosos e outras pessoas com deficiência motora ou mobilidade restrita a executar tarefas domésticas cotidianas.

1.1 Formulação do problema

Segundo o censo realizado pelo Instituto Brasileiro de Geografia e Estatística (IBGE) no ano de 2010 (IBGE, 2012), cerca de 7% da população brasileira, que representa por volta de 14 milhões de pessoas, apresentam algum tipo de deficiência motora.

Outro estudo, publicado pela Secretaria de Direitos Humanos da Presidência da República, aponta dados sobre o envelhecimento no Brasil referindo-se a uma pesquisa do IBGE que confirma a cristalização do envelhecimento da população brasileira, onde apresenta uma população idosa totalizando cerca de 23,5 milhões de pessoas em 2012 (SECRETARIA DE DIREITOS HUMANOS, 2012).

Esse cenário de mudanças demográficas com acelerado envelhecimento populacional, causa um impacto no que se refere aos cuidados requeridos por pessoas idosas e com mobilidade reduzida. Tendo em vista a crescente demanda por cuidados pessoais e auxílio na execução das tarefas do seu dia-a-dia, a sociedade mundial precisa encontrar estratégias para lidar com esses desafios.

De acordo com pesquisadores, projetistas, engenheiros, e outros especialistas, o desenvolvimento de tecnologias assistivas nos dias de hoje permite que pessoas idosas vivam em seus lares usufruindo de maior independência na realização de tarefas cotidianas (FLANDORFER, 2012), sendo isto também observado no que se refere às pessoas que apresentam algum tipo de debilidade motora (BAINBRIDGE ISLAND REVIEW, 2008).

Acompanhando o envelhecimento da população, se apresenta um aumento no número de ocorrências relacionadas ao aparecimento de algum tipo de deficiência. Nessa mesma situação encontram-se também as pessoas que por diversos outros motivos, que não a velhice, foram acometidas por algum tipo de debilidade motora.

As restrições corporais enfrentadas por esses indivíduos acabam incapacitando-os a realizar certos tipos de tarefas, chegando, em certos casos, a oferecer riscos à sua segurança, o que compromete sua qualidade de vida e os torna completamente dependentes de um suporte provido por cuidadores (CORNELL et al., 2008).

Com o intuito de prover ao indivíduo com *mobilidade reduzida* (BRASIL, 2009) uma maior sensação de segurança, controle e independência a partir do uso de tecnologias assistivas abertas de baixo custo, pretende-se desenvolver neste trabalho uma pesquisa envolvendo as áreas de engenharia e tecnologias assistivas.

Este projeto apresenta o desenvolvimento de uma interface de comandos por voz que seja facilmente adaptada e incorporada aos já existentes sistemas e ferramentas de auxílio

ao controle do ambiente doméstico (domótica). Esses sistemas atuam fundamentalmente permitindo o controle remoto das facilidades do lar por indivíduos que apresentam alguma restrição motora ou dificuldade quanto a sua mobilidade.

Essas ferramentas representam, assim, formas de auxiliar no cotidiano dessas pessoas dando suporte à execução de tarefas como: controlar eletrodomésticos, iluminação ambiente, dispositivos de cuidados com a saúde, dentre outros. Nesses casos, convém utilizar interfaces alternativas de controle. Um exemplo é o uso do Reconhecimento Automático da Fala (RAF) para a interação com tais facilidades, de modo a superar a exigência de habilidades de coordenação motora para a entrada de comandos aos referidos sistemas.

Os compromissos a serem atingidos pela interface desenvolvida nesse projeto envolvem, portanto, o atendimento de critérios de portabilidade e mobilidade, exigindo que apresentem um caráter multifuncional, através de uma estrutura *embarcável* capaz de prover a entrada de comandos de voz para a tomada de decisões de controle. Além disso, seu projeto observa a requisitos de baixo custo e alta flexibilidade, utilizando-se de tecnologias livres e de código aberto que contribuam para o desenvolvimento de aplicações de tecnologia assistiva acessíveis.

1.2 Objetivos

1.2.1 Geral

Desenvolver uma plataforma de reconhecimento automático da fala para o auxílio de pessoas com mobilidade reduzida.

1.2.2 Específicos

a) Realizar estudo sobre tecnologias eficientes, portáveis e de baixo custo, que se mostrem viáveis ao desenvolvimento de interfaces de comandos por voz para a interação humano-computador;

b) Elaborar concepção de interface de baixo custo, portável e de uso pessoal, para aplicações de comandos através do reconhecimento automático da fala (RAF) em tecnologias assistivas que atuem no controle de dispositivos domésticos;

c) Desenvolver recursos de base de texto (gramática) e base de áudio para a criação de modelo acústico específico para aplicações de comando e controle em domótica;

d) Avaliar a interface concebida com fins de validar sua eficiência.

1.3 Motivação no âmbito das tecnologias assistivas

A busca por maior independência e autonomia para as pessoas com deficiência tem se apresentado como um fator decisivo, ao proporcionar mais qualidade de vida a esses indivíduos. Abarca melhoramentos da forma como se comunicam, de sua mobilidade, de passarem a ter um maior controle sobre o ambiente onde estão inseridos e melhores experiências de aprendizado e interação com a sociedade (SARTORETTO; BERSCH, 2014).

A superação desses desafios se evidencia na importância de se combinar o uso de tecnologias assistivas com o suporte humano tradicional. O emprego dessas tecnologias não deve ser visto como uma prática que substitua o cuidado humano.

Contudo, existem diversas aplicações em que as tecnologias assistivas atendem necessidades que os cuidados humanos não seriam capazes de dar o devido suporte. Como, por exemplo, o monitoramento dos pacientes vinte e quatro horas por dia, com o intuito de prevenir acidentes e auxiliar em tarefas diárias simples (CORNELL et al., 2008).

O suporte proporcionado pelas tecnologias assistivas consiste em dispositivos para monitoramento remoto de pacientes (*telecare*, telessaúde e telemedicina), sistemas de residências inteligentes (*smart homes*), robôs domésticos, terapia assistida por robôs, dentre outros. Robôs interativos são ainda capazes de cooperar com as pessoas provendo assistência nas atividades do dia a dia, realizando lembretes do horário de tomar um medicamento e até ajudando-os a preparar a comida, comer e realizar a limpeza (BROADBENT; STAFFORD; MACDONALD, 2009 apud FLANDORFER, 2012).

Os sistemas adjacentes associados a esses dispositivos, são capazes de prover: sistemas de detecção de quedas, testes de pressão sanguínea, reconhecimento do esforço para respirar ou de problemas do coração. Podem antecipar os riscos à saúde dos pacientes (MIORI; RUSSO, 2012), inclusive enviando alertas imediatamente aos cuidadores nos casos de emergência.

Essas funcionalidades representam uma ampla variedade de aplicações de elevado potencial e interesse por parte da indústria de *healthcare* e inserem-se numa moderna abordagem de aplicação das tecnologias da Internet das Coisas (IoT) em casos de uso de *eHealth* (POENARU; POENARU, 2013; PERERA et al., 2014). Atualmente ainda tem sua ampla utilização e integração tecnológica limitada por, estar condicionada a soluções proprietárias. Demanda, assim, esforços voltados para o emprego de tecnologias, padrões e protocolos abertos (SEBESTYEN et al., 2014).

A integração entre os sistemas de uma residência envolve a configuração de controladores centrais Cornell et al. (2008). Dessa forma, associam-se sistemas de comunicação (por exemplo, o telefone ou a internet), com sistemas de entretenimento (televisão, rádio, etc.), controle de iluminação e acesso (luzes, portas, janelas), dentre outros dispositivos (EUROPEAN COMISSION - INFORMATION SOCIETY AND MEDIA, 2010). Assim, englobam várias categorias de produtos, conforme a classificação ISO 9999:2007 adotada pelo CNPTA (Catálogo Nacional de Produtos de Tecnologia Assistiva).

Nesse sentido, o desenvolvimento de tecnologias assistivas tem sido amplamente

estimulado – a exemplo do Plano Viver sem Limite (BRASIL, 2011), lançado pelo Governo Federal como um Plano Nacional dos Direitos da Pessoa com Deficiência, o qual teve seu mecanismo de implantação no âmbito da ciência e da tecnologia marcado pela criação do Centro Nacional de Referência em Tecnologia Assistiva (CNRTA), tendo este último o propósito de "... articular nacionalmente uma rede cooperativa de pesquisa, desenvolvimento e inovação na área de Tecnologia Assistiva" (BRASIL, 2012).

Definida pelo Comitê de Ajudas Técnicas (CAT) como:

> uma área do conhecimento, de característica interdisciplinar, que engloba produtos, recursos, metodologias, estratégias, práticas e serviços que objetivam promover a funcionalidade, relacionada à atividade e participação, de pessoas com deficiência, incapacidades ou mobilidade reduzida, visando sua autonomia, independência, qualidade de vida e inclusão social (BRASIL, 2009),

A tecnologia assistiva não possui ainda um padrão nacional de classificação dos seus produtos. A adoção de um sistema de classificação é de fundamental importância e deve servir como diretriz para os diversos ramos de atuação associados ao desenvolvimento de *recursos* (SARTORETTO; BERSCH, 2014) apropriados ao atendimento das necessidades funcionais dos portadores de deficiência.

Levando em conta os aspectos supracitados, foi estruturado nacionalmente um serviço de informação sobre produtos de Tecnologia Assistiva. O Catálogo Nacional de Produtos de Tecnologia Assistiva (CNPTA) "... consiste numa ferramenta web que possibilita a realização de buscas sobre os produtos de Tecnologia Assistiva fabricados ou distribuídos no Brasil" (BRASIL, 2014). Disponibiliza essencialmente um serviço de informação de produtos de TA (Tecnologias assistivas), de modo a organizá-los com base na classificação ISO 9999:2007.

Através de uma consulta ao referido catálogo, foi observado um quantitativo ainda baixo de produtos cadastrados, conforme a Tabela 1 (BRASIL, 2014). Destacou-se as categorias relacionadas a "Produtos de apoio para atividades domésticas", "Mobiliário e adaptações para habitação e outros edifícios", "Produtos de apoio para manuseamento de objetos e dispositivos" e "Produtos de apoio para melhoria do ambiente, máquinas e ferramentas".

Tabela 1 – Quantitativo de produtos de TA cadastrados no CNPTA.

Tipo de produto	**Quantidade**
Apoio para atividades domésticas	25
Mobiliário e adaptações para habitação e outros edifícios	91
Apoio para manuseio de objetos e dispositivos	31
Apoio para melhoria do ambiente, máquinas e ferramentas	26

Fonte: Adaptado de Brasil (2014).

É importante salientar que grande parte dos produtos cadastrados são importados e não implementam funcionalidades com interfaces adaptadas para o controle remoto das facilidades do lar.

Além disso, os custos para aquisição e manutenção dos equipamentos de tecnologia assistiva ainda são muito altos. Pessoas que sofreram algum tipo de dano na coluna vertebral, perdendo parte ou a totalidade dos movimentos do corpo, por exemplo, necessitam de cadeiras de rodas motorizadas e diversos outros dispositivos, envolvendo baterias para alimentação dos sistemas de comunicação sem fio e motorização, componentes geralmente caros (BAINBRIDGE ISLAND REVIEW, 2008).

Assim, um gargalo determinante para que se alcance a garantia dos diretos das pessoas com deficiência se configura nos altos gastos relacionados aos cuidados com sua saúde.

Diante desse contexto, o desenvolvimento de tecnologias assistivas de baixo custo se apresenta como um relevante papel a ser desempenhado pelas diversas instituições constituídas na sociedade, configurando-se como uma promissora área de estudo, pesquisa e inovação. A pesquisa, de caráter multidisciplinar, integra diferentes áreas – como saúde, engenharia, educação e inclusão social – em benefício de melhorias na *funcionalidade* (SARTORETTO; BERSCH, 2014) de pessoas com mobilidade reduzida.

1.4 Motivação no âmbito das tecnologias de comandos por voz

A entrada de comandos por voz pode ser uma conveniente alternativa para que pessoas com mobilidade reduzida, que tenham preservada boa capacidade das habilidades da fala, realizem o controle do computador ou outros dispositivos (CHANDRAMOULI; AGARWAL, 2009; EUROPEAN COMISSION - INFORMATION SOCIETY AND MEDIA, 2010).

A fala se constitui na mais básica, comum e eficiente forma de comunicação entre os seres humanos. Por isso, alcançar mais naturalidade na interação com máquinas e computadores através de comandos por voz tem sido alvo de grande esforço de pesquisa e desenvolvimento ao longo das últimas décadas. Nesse aspecto, destaca-se o desenvolvimento de ferramentas e sistemas de reconhecimento automático da fala (PRABHAKAR; SAHU, 2013).

Experiências vivenciadas por Qidwai e Shakir (2012) em colaboração com centros de reabilitação e hospitais indicaram que pacientes idosos e pessoas com mobilidade reduzida usualmente apresentam uma voz estável, até mesmo nos casos de tetraplegia e idade muito avançada. Observou-se, ainda, maior afinidade à pronúncia de comandos em sua língua materna.

Na maioria desses casos a voz é razoavelmente inalterada em comparação com as debilidades de coordenação motora. Dessa forma, a incorporação de interfaces de controle por voz se apresenta como uma conveniente alternativa às interfaces de controle tradicionais, como *joystick*, teclado e mouse, uma vez que essas últimas são inadequadas para os casos de indivíduos acometidos por esse tipo de debilidade, com sérios problemas de coordenação motora.

Com os avanços ocorridos nas últimas décadas na área de RAF, sua utilização encontra hoje uma vasta gama de aplicações que requerem a interação entre homens e máquinas. Exemplos disso são o processamento automático de chamadas telefônicas e a interação para o controle de funções no computador e outros equipamentos (BENTHIN, 2013).

Outro promissor exemplo de aplicação para a utilização de comandos por voz se verifica na automatização dos recursos do ambiente doméstico. A indústria da automação residencial tem crescido rapidamente, alavancada pela necessidade de se prover sistemas de suporte para pessoas idosas e com algum tipo de deficiência motora. Busca-se proporcionar controle remoto da iluminação e eletrodomésticos, em casa ou no escritório, através do RAF (ALSHU'EILI; GUPTA, 2011).

Vislumbra-se potencialmente a concepção de interfaces de voz flexíveis ao ponto de poderem ser acessadas a partir dos mais variados dispositivos, como: PCs (*Personal Computers*), *tablets*, telefones celulares convencionais, telefones fixos convencionais, telefones VoIP (*Voice over Internet Protocol*) fixos, *softphones* e *smartphones* (MADSEN; MEGGELEN; BRYANT, 2011; BATISTA, 2013; DIAS; LUCENA; SANTOS, 2014).

Tevah (2006) destaca que a aplicação das técnicas de reconhecimento automático da fala para sistemas de interface é extremamente motivadora por não exigir alto grau de aprofundamento na área de RAF, uma vez que as aplicações utilizam linguagens de alto nível, fazendo uso do complexo processamento estatístico já implementado em camadas de *software* inferiores (nas próprias ferramentas de reconhecimento). Contribuem, aliado ao rápido desenvolvimento de computadores pessoais e dispositivos portáteis, para a popularização dos sistemas com interfaces de voz.

Nessa perspectiva, agregar capacidades de execução de comandos por voz aos sistemas domóticos se apresenta como uma promissora solução em tecnologia assitiva. A adoção de diferentes abordagens para o desenvolvimento de interfaces de reconhecimento automático da fala vai desde a utilização de módulos de reconhecimento (PIMENTEL et al., 2014), até a integração de softwares de RAF a servidores de comunicação, como o Asterisk^{TM} - que provê serviços de central telefônica, servidores de conferência, e outras soluções customizadas, interconectadas através da Internet (MADSEN; MEGGELEN; BRYANT, 2011; BATISTA, 2013; DIAS; LUCENA; SANTOS, 2014).

Tais aplicações integradas de telefonia, automação residencial e serviços de cuidados com a saúde (*Home Health Care*), inserem-se no paradigma atual de desenvolvimento para a IoT (DOHR et al., 2010; MIORI; RUSSO, 2012; POENARU; POENARU, 2013; SEBESTYEN et al., 2014; PERERA et al., 2014; YANG et al., 2014).

1.5 Organização do trabalho

Este documento descreve as ações realizadas ao longo do trabalho desenvolvido, as quais estão organizadas no texto como descrito a seguir.

O Capítulo 2 apresenta um levantamento do estado da arte no âmbito das tecnologias assistivas, delineando posteriormente um estudo da evolução e atual estado da arte na área das aplicações e tecnologias de comandos por voz.

Em seguida, o Capítulo 3 elucida a fundamentação teórica que permeia os sistemas de RAF, apresentando, a partir de um estudo das ferramentas que compõe esses sistemas, os princípios e técnicas de processamento empregados no treinamento de modelos acústicos.

Já o Capítulo 4, descreve o processo metodológico que norteou o desenvolvimento do projeto, destacando as características dos materiais e tecnologias utilizados. Em seguida, os resultados obtidos são apresentados no Capítulo 5.

Por fim, no Capítulo 6, são expostas as conclusões, contribuições do trabalho e sugestões para trabalhos futuros, seguidas das referências bibliográficas em que a pesquisa realizada se fundamenta.

2 Estado da arte

2.1 Tecnologias assistivas

Tradicionalmente, os modelos de cuidados com aqueles indivíduos que apresentam necessidades especiais tiveram ênfase na ajuda fornecida pela própria família, por profissionais de saúde e outros profissionais cuidadores. No entanto, a sustentabilidade desses modelos fica comprometida quando se observam tendências de profundas mudanças demográficas (SECRETARIA DE DIREITOS HUMANOS, 2012), observadas com o envelhecimento populacional crescente. Essa realidade imprime certa pressão sobre a força tarefa e mão de obra necessárias para os cuidados com idosos e portadores de necessidades especiais, numa sociedade cada vez mais idosa (FLANDORFER, 2012).

Um estudo realizado pela *European Comission - Information Society and Media (2010)* apresenta as tendências em TA para pessoas com debilidades motoras, ou seja, indivíduos que apresentam dificuldades em movimentar seu corpo, desde limitações nos movimentos de um membro em particular até a paralisia total. Os autores comentam que a debilidade motora pode estar relacionada à perda de energia ou tonacidade dos músculos, pouco controle de movimentos voluntários complexos, pouca coordenação de movimentos voluntários ou disfunções mentais em regiões do cérebro responsáveis pelos movimentos.

> Tecnologias assistivas (TA) podem ser definidas como qualquer item, equipamento ou aplicação que seja usada para aumentar, manter ou aperfeiçoar as capacidades funcionais de pessoas portadoras de alguma necessidade especial. Os produtos de TA podem ajudar essas pessoas na execução de suas tarefas cotidianas, comunicação, educação ou trabalho. Tais produtos dão a esses indivíduos alternativas de acesso a dispositivos eletrônicos avançados como computadores ou telefones celulares. Assim, ajudam-lhes a gozar de maior independência e melhoram sua qualidade de vida (EUROPEAN COMISSION - INFORMATION SOCIETY AND MEDIA, 2010).

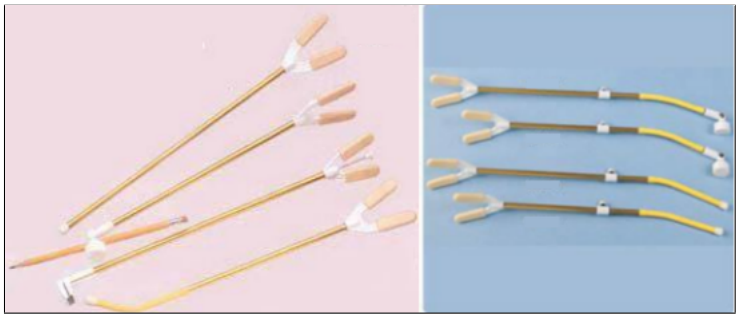
Podem ainda ser decorrentes de doenças, como: lesões na coluna vertebral, lesões traumáticas no cérebro, desordens neurológicas, paralisia cerebral, esclerose múltipla, distrofia muscular, esclerose lateral amiotrófica, trauma cranial ou paralisia. O referido estudo, então, descreve vários sistemas de TA que utilizam interfaces alternativas para a interação humano-computador.

Dispositivos bem simples, em sua maioria puramente mecânicos, porém de grande utilidade para o controle de interfaces de entrada como teclado, *mouse* e telas sensíveis ao toque, são os conhecidos por *headpointers* (Figura 1) e *mouth sticks* (Figura 2). Sendo os primeiros acoplados à cabeça das pessoas, enquanto o segundo possui uma peça adaptada para ser segurada com a boca e dentes do usuário (EUROPEAN COMISSION - INFORMATION SOCIETY AND MEDIA, 2010).

Figura 1 – *Headpointer* ajustável e de visão limpa (*clear vision*).

Fonte: *Wisdom King (2015 apud EUROPEAN COMISSION - INFORMATION SOCIETY AND MEDIA, 2010)*

Figura 2 – *Mouthstick* e *telescope mouthstick*.



Fonte: *Winsdom King (2015 apud EUROPEAN COMISSION - INFORMATION SOCIETY AND MEDIA, 2010)*

Outro projeto envolve a utilização de um ímã grudado à língua do paciente para determinar o movimento de uma cadeira de rodas de acordo com a direção em que a língua se movimenta (MADRIGAL, 2008 apud LU; CHEN, 2012); em alguns dispositivos utilizou-se *joysticks* controlados pela boca (AYLOR et al.; PELLEGRINI et al., 1979, 2004 apud LU; CHEN, 2012).

Plaza, Avilés e Aperador (2013) propõe uma abordagem utilizando sensores infravermelhos para o desenvolvimento de um *mouse* oral que possa ser utilizado para controlar dispositivos do lar, bem como cadeira de rodas, através da detecção dos movimentos da boca.

Essas interfaces, no entanto, podem ser bastante inconvenientes para as pessoas que as utilizam, restringindo sua comunicação verbal ou dificultando seus momentos de refeição por ocupar sua boca ou utilizar máscaras para serem acopladas à cabeça (LU; CHEN, 2012).

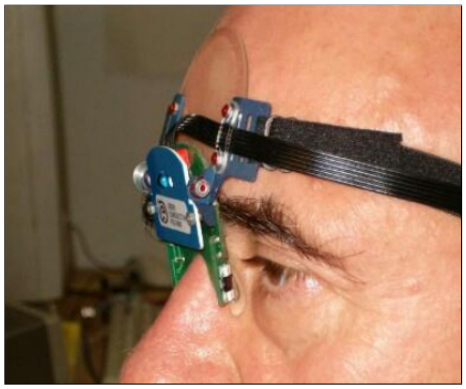
Em seu relatório do estado da arte de tecnologias assistivas, a *European Comission - Information Society and Media (2010)* apresenta uma série de dispositivos, interfaces e siste-

mas alternativos de entrada, como: mouses e teclados especiais, teclados virtuais altamente adaptáveis e configuráveis, mecanismos para melhorar a eficiência na digitação de textos (por exemplo, o autocompletar), dispositivos com chaves e botões (*switches*) com design adaptado, técnicas de escaneamento (*scanning*) que destacam sequencialmente os itens num terminal de acesso ou ambiente gráfico, como também interfaces para o controle a partir dos movimentos da cabeça, dos olhos, do nariz ou da face.

As interfaces que utilizam os movimentos da cabeça e da face empregam diversas tecnologias, como: acelerômetros, sensores e câmeras infravermelhos, técnicas e algoritmos de visão computacional, medição de sinais bioelétricos. Um exemplo são os mouses infravermelhos, que utilizam sensores desses tipos para detectar os movimentos do corpo do usuário, como o *SmartNav*[1] e o *HeadMouse extreme*[2] .

Existem ainda alternativas que permitem ao usuário realizar o controle desejado a partir do movimento do piscar de olho. Um exemplo dessas interfaces é o *Blink-IT*[3] , ilustrado na Figura 3.

Figura 3 – Blink-IT.



Fonte: *European Comission - Information Society and Media (2010)*

Ainda outras modalidades alternativas de entrada são observadas na literatura, como o sensoriamento dos sinais elétricos produzidos pelos músculos (Eletromiográficos – EMG) e o controle através do pensamento, com o sensoriamento de sinais eletroencefalográficos (EEG). Existem dois métodos principais de implementação para esses tipos de interface humano-computador. Um deles utiliza sensoriamento não invasivo, enquanto o outro envolve o implante cirúrgico do sensor na superfície do músculo ou do cérebro humano. Em ambos

[1] NATURAL POINT. *SmartNav*. 2015. Disponível em: <http://www.naturalpoint.com/smartnav/>. Acesso em: 03 jun. 2015.

[2] ORIGIN INSTRUMENTS. *HeadMouse® Extreme*. 2015. Disponível em: <http://www.naturalpoint.com/smartnav/>. Acesso em: 03 jun. 2015.

[3] OBER CONSULTING. *Blink-It*. 2015. Disponível em: <http://www.ober-consulting.com/product/blink-it/>. Acesso em: 03 jun. 2015.

os casos, no entanto, os sinais apresentam alta complexidade, dificultando alcançar níveis satisfatórios de confiabilidade (GALAN et al.; PHILIPS et al., 2008, 2007 apud LU; CHEN, 2012).

Um método para o controle de prótese robótica que produz um torque proporcional àquele gerado pelas contrações musculares do usuário, utilizando os sinais EMG, é proposto em Hayashi, Kawamoto e Sankai (2005).

Starr (2015) noticia um trabalho desenvolvido pela equipe de cientistas do Instituto de Tecnologia da Califórnia, onde foi realizado o implante neural cirúrgico de sensores numa região do cérebro que controla a intenção do paciente em se movimentar, na tentativa de criar uma prótese controlada por sinais EEG, tendo obtido sucesso na movimentação de um braço robótico, conforme ilustra a Figura 4.

Figura 4 – Paciente utilizando braço robótico controlado por sinais do cérebro.

Fonte: Starr (2015)

Lu e Chen (2012) se propuseram a investigar um sistema para o controle de cadeira de rodas que integra diversas modalidades de entrada buscando atender a critérios de interface amigável, de baixo custo, não intrusiva, rápida e fácil de implementar, e que seja independente da linguagem do usuário. Para tanto, apresenta um sistema flexível, que possa incorporar facilmente novos módulos de interfaceamento na entrada, como também atender novas necessidades de controle. O sistema proposto implementa o controle a partir dos movimentos da face (sobrancelhas, nariz e orelha) associado ao controle por reconhecimento da fala independente da linguagem, utilizando uma determinada codificação sonora a ser pronunciada pelo usuário.

Ainda relacionado ao suporte a cadeirantes, Yang et al. (2014) apresentam um sistema móvel de *healthcare* baseado nas tecnologias emergentes da IoT, focando-se numa arquitetura e projeto de redes sem fio de sensores corporais (*Wireless Body Sensor Network*

– WBSNs) para monitoramento dos batimentos cardíacos, sensores de eletrocardiograma (ECG), sensoriamento do ambiente e controle de atuadores, dentre outras funcionalidades.

De acordo com *European Comission - Information Society and Media (2010)*, uma conveniente alternativa para o desenvolvimento de interfaces em tecnologias assistivas é a entrada de comandos por voz. A fala é a principal forma de comunicação entre os seres humanos. Assim, utilizar ferramentas de entrada de comandos por voz como interface humano-computador, se apresenta como uma solução altamente desejável para tecnologias assistivas por ser independente das habilidades de coordenação e controle de movimentos voluntários complexos dos membros superiores ou inferiores do corpo humano.

O sistema operacional Microsoft Windows®, a partir de sua versão 7, por exemplo, já incluiu a possibilidade de se utilizar ferramentas de comandos por voz que permitam ao usuário controlar seu ambiente operacional através da fala. Esse sistema está disponível para várias línguas, dentre elas: inglês, espanhol, alemão, francês e japonês (MICROSOFT, 2015).

Dias, Lucena e Santos (2014) propuseram uma solução de baixo custo e baseada em plataformas bem estabelecidas de *software* e *hardware* livre e de código aberto, com alto potencial de empregabilidade para sistemas de automação residencial, utilizando o servidor *Asterisk* e placas *Arduino* com *shields Ethernet* integrados. Os autores sugerem que uma interface de reconhecimento de comandos por voz seja agregada ao sistema implementado, com o intuito de estender suas funcionalidades para que atenda ao contexto das aplicações de tecnologias assistivas.

Maiores detalhes quanto ao estado da arte das ferramentas e interfaces de reconhecimento automático de comandos por voz, suas aplicações e sua relação com as tecnologias assistivas serão elucidadas na seção 2.2, a seguir.

2.2 Aplicações e tecnologias de comandos por voz

Segundo Spaans (2004), indiscutivelmente, a primeira máquina a responder a estímulos da voz humana foi um cachorro de brinquedo chamado *Radio Rex*, fabricado na década de 20, projetado para responder ao reconhecer a pronúncia de seu nome. O autor chama a atenção para o fato de que a interação entre os seres humanos e a tecnologia está em constante mudança, e o reconhecimento da fala é uma força motriz nesse processo.

As décadas de 30 e 40 apresentaram avanços limitados no campo das pesquisas em reconhecimento da fala, embora tenha-se alcançado significativos melhoramentos no que diz respeito a compressão de voz e técnicas de análise da fala. O primeiro sistema computadorizado para o reconhecimento de palavras foi desenvolvido na década de 50 pelo laboratório Bell da AT&T (*American Telephone and Telegraph*), sendo capaz de reconhecer uma entrada de dígitos de zero a nove, pronunciados por um único locutor, com consideráveis pausas entre as elocuções.

Nesse mesmo período, foi introduzido o uso de fonemas como unidades linguísticas

básicas dos sistemas de reconhecimento, o que levou ao desenvolvimento de um sistema pelo MIT (*Massachusetts Institute of Technology*), no ano de 1959, capaz de reconhecer o som de vogais com 93% de acurácia (SPAANS, 2004).

A alta variabilidade da voz humana - influenciada por diversos fatores, como: o sotaque, sexo, idade, entonação e a variabilidade do próprio ambiente e do canal de comunicação, como também a similaridade fonética entre determinadas elocuções - tem ainda se apresentado como desafios no desenvolvimento de sistemas de RAF. A influência desses fatores no reconhecimento depende do sistema que se deseja implementar (BETTELHEIM; STEELE, 2010; PRABHAKAR; SAHU, 2013).

Para o controle de dispositivos pessoais e aplicações de comandos simples embarcadas, que consomem poucos recursos de processamento e não requerem grande capacidade de armazenamento com dados de dicionários complexos, os modelos acústicos são baseados em gravações da voz do usuário relacionadas aos comandos envolvidos na aplicação; um modelo dependente de locutor. Por outro lado, os sistemas baseados em PC possuem recursos para suportar amplos vocabulários e modelos acústicos complexos, dando margem para a criação de modelos independentes de locutor (BETTELHEIM; STEELE, 2010).

Durante a década de 60, o foco manteve-se na modelagem acústica. Em 1966, o sistema desenvolvido pelo MIT foi aperfeiçoado, sendo agora capaz de lidar com um vocabulário de 50 palavras. Já na década de 70, com a introdução da teoria de Modelos Ocultos de Markov (*Hidden Markov models* – HMMs) para a modelagem dos sons da fala, os pesquisadores Jim e Janet Baker, da Universidade de Carnegie Mellon (CMU), aplicaram essa teoria para o reconhecimento automático contínuo da fala (ENGINEERING AND TECHNOLOGY HISTORY WIKI, 2015).

A partir de então, o DARPA (*Defense Advanced Research Projects Agency*) realizou investimentos em programas para o desenvolvimento de sistemas computacionais capazes de compreender fala contínua, envolvendo instituições como a CMU, o Instituto de Pesquisa de Stanford (*Stanford Research Institute* – SRI), o Laboratório de pesquisa Lincoln do MIT, a *Systems Development Corporation* (SDC) e a empresa Bolt, Beranek e Newman (BBN) (SPAANS, 2004).

Nesse período, a CMU desenvolveu os sistemas HEARSAY-I e DRAGON, posteriormente lançando o HEARSAY-II e o HARPY, sendo este último o que mais impressionava naquele tempo. Capaz de reconhecer sentenças completas que consistiam de um número limitado de estruturas gramáticas, necessitava de cerca de 50 computadores do estado da arte, à época, para realizar seus cálculos e reconhecer por volta de 1011 palavras com 95% de acurácia.

Finalmente, no final da década de 80, a CMU lançou o SPHINX, que obteve êxito ao ser submetido aos padrões e orientações do sistema de avaliação realizada pelo DARPA em cooperação com o *National Institute of Standards and Technology* (NIST). Então, a partir da década de 90, as empresas privadas passam a determinar os rumos da pesquisa em RAF

(SPAANS, 2004).

Um sistema de propósito geral baseado em PC para o controle do ambiente por pessoas com deficiência foi proposto por Aguilera et al. (1992). Implementando funcionalidades para atender o telefone e controlar luzes e eletrodomésticos, combinou como interfaces de entrada o teclado juntamente com o controle por voz, decodificando a elocução de caracteres pronunciados em uma tecla correspondente. Para tanto, utilizou um módulo externo de *hardware* conectado ao PC para o reconhecimento da fala, o *Intravoice VI Voice recognition module*. Em seu artigo, o autor sugere ainda como trabalho futuro, a possibilidade de utilizar o RAF baseado em *software*, destacando o programa *Dragon System Dictate*.

Vallés et al. (1996) propõe em seu artigo um sistema de controle do ambiente com múltiplas modalidades de entrada para idosos e deficientes físicos. A ferramenta utilizada para o reconhecimento da fala não é descrita no artigo, sendo destacados a metodologia utilizada, que envolveu a participação de idosos e deficientes no processo. Foi desenvolvida uma interface gráfica de alta usabilidade, implementada numa arquitetura de comunicação que permite o gerenciamento de interfaces com o usuário baseando-se nos protocolos EHS (*European Home System*) e TECI (*Thomson Electronics Computer Interface*) para o envio dos comandos aos dispositivos a serem controlados.

Já no ano de 1997, no âmbito do projeto de desenvolvimento de um *toolkit* de ditado para a língua japonesa, a Agência de Promoção da Tecnologia da Informação do Japão (*Information-technology Promotion Agency* - IPA), passou a desenvolver o programa de computador Julius, para fins de pesquisa científica. Esse *software* se constitui num decodificador para o reconhecimento contínuo da fala com amplo vocabulário (*Large Vocabulary Continuous Speech Recognition* - LVCSR), envolvendo outras instituições japonesas, como o Laboratório Kawahara, da Universidade de Kioto; o Laboratório Shikano, do Instituto Nara de Ciência e Tecnologia; e o Instituto Nagoya de Tecnologia, onde se estabeleceu a equipe do projeto Julius (JULIUS, 2014; LEE, 2010).

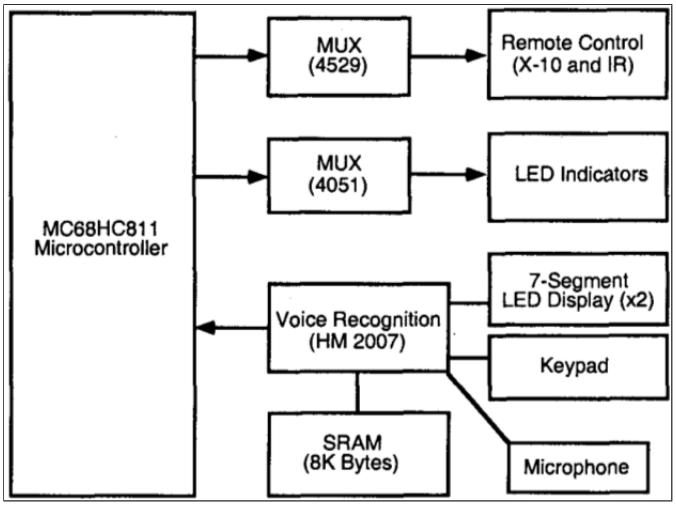
Um dispositivo portátil, facilmente acoplável a uma cabeceira de cama ou uma cadeira de rodas, para auxiliar pessoas com deficiência no controle do ambiente através de comandos de voz é apresentado em Jiang et al. (2000). Destacando as vantagens da solução com relação à utilização de *softwares* em PCs, principalmente com relação ao baixo custo, os autores exploraram o estado da arte em circuitos integrados de reconhecimento de voz, à época, utilizando o módulo HM2007[4] , que é um *chip* CMOS (*Complementary metal-oxide-semiconductor*) para o reconhecimento da fala em circuito LSI (*Large-Scale Integration*), implementando um *front-end* analógico, análise do sinal de entrada, o processo de reconhecimento e funções de controle.

O dispositivo desenvolvido trabalha com dois modos de reconhecimento, permitindo reconhecer até 40 palavras isoladas ou 20 frases compostas por pares de palavras conectadas.

[4] SPEECH Recognition System HM2007. Disponível em: <http://www.sunrom.com/p/speech-recognition-system-hm2007>. Acesso em: 12 jun. 2015.

O sistema consiste ainda de um microprocessador central MC68HC11[5] , da *Freescale*, que utiliza radiofrequência para a comunicação com os eletrodomésticos e módulos que implementam o protocolo X-10, a serem controlados, conforme ilustra seu diagrama de blocos, apresentado na Figura 5.

Figura 5 – Diagrama de blocos do sistema de controle do ambiente ativado por voz.



Fonte: Jiang et al. (2000)

Pelo fato de a tecnologia de RAF ser atrativa no âmbito das tecnologias assistivas, a redução dos custos de TA tem sido uma preocupação sempre presente, buscando o desenvolvimento de produtos e soluções cada vez mais acessíveis. Em um projeto apresentado à Conferência Internacional Anual da Sociedade de Engenharia de Reabilitação e Tecnologia Assistiva da América do Norte (*Rehabilitation Engineering and Assistive Technology Society of North America* - RESNA), em 2003, Hall e Molloy (2003) publicaram o desenvolvimento de uma unidade de controle do ambiente com custo inferior a $500 dólares.

A comunicação entre os dispositivos a serem controlados também utiliza módulos X-10, enquanto a ferramenta de reconhecimento da fala empregada foi configurada a partir do computador com a utilização do *software Dragon Naturally Speaking*[6] , através do qual foram treinadas elocuções empregadas como atalhos para os comandos do teclado que permitem o controle de dispositivos de iluminação individualmente.

Ainda no âmbito de *software*, voltando-se principalmente para sistemas embarcados (*embedded systems*), foi desenvolvida pela empresa *Asahi Kasei*, a plataforma VORERO^{TM} (*Voice Recognition Robust*)[7] , que é essencialmente um *middleware* de RAF utilizado em

[5] MC68HC11 Family Datasheet. Disponível em: <http://www.freescale.com/files/microcontrollers/doc/data_sheet/M68HC11E.pdf>. Acesso em: 12 jun. 2015.

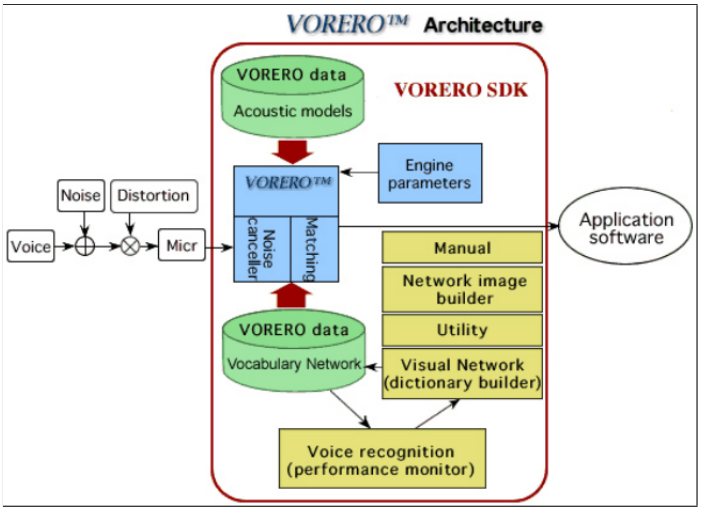
[6] NUANCE COMMUNICATIONS. *Dragon Speech Recognition Software*. Disponível em: <http://www.nuance.co.uk/dragon/index.htm>.

[7] ASAHI KASEI. *VORERO*. 2000. Disponível em: <http://www.asahi-kasei.co.jp/asahi/en/services_products/other/>.

sistemas de navegação de automóveis, telefones celulares e *smartphones*, com possibilidade de expansão para incluir outros aparelhos e produtos eletrônicos para o lar. Baseia-se nos HMMs e permite a utilização simultânea dos modelos de fonemas e de palavras.

Esse *middleware* inclui avançada análise acústica, que provê robustez através do uso de técnicas de redução de ruído e cancelamento de eco, sendo ainda um sistema independente de locutor. Sua arquitetura, ilustrada na Figura 6 apresenta componentes similares aos dos sistemas de reconhecimento da fala típicos, com um *front-end* de processamento do sinal e demais componentes que implementam o RAF. Utiliza, no entanto, dados proprietários, que consistem de um conjunto de modelos acústicos, um dicionário e uma rede de vocabulário (SPAANS, 2004).

Figura 6 – Arquitetura do *middleware* VORERO^{TM}.



Fonte: Spaans (2004)

De acordo com o estudo realizado por Sampaio Neto (2006), a tecnologia de processamento de fala (ou voz) encontra-se bastante avançada e, em escala mundial, existe vasta disponibilidade de *software*. O referido autor comenta ainda a escassez de recursos de reconhecimento automático da fala para o português brasileiro, uma vez que procura destacar em seu trabalho nas tecnologias para o desenvolvimento de aplicações de reconhecimento da fala, sem aprofundar os aspectos da ciência da fala, como os HMMs, relacionados às camadas de abstração mais baixas.

O autor apresenta um protótipo de aplicativo para o ensino da língua inglesa, o CALL (*Computer-assisted Language Learning*), implementado no desenvolvimento de seu projeto utilizando a interface de programação SAPI (*Speech Application Programming Interface*) da *Microsoft* para síntese e reconhecimento automático da fala em inglês, síntese em português e utilizando também agentes visuais. Além disso, foi apresentada uma proposta para integração de síntese e RAF ao aplicativo de *chat* IRC (*Internet Relay Chat*), utilizando a interface de

programação JSAPI (*Java Speech Application Programming Interface*).

Já Tevah (2006), motivado pela inspiração dos sistemas robóticos na comunicação humana através da fala, realizou um estudo sobre as diversas técnicas empregadas nos sistemas de fala contínua que implementam o estado da arte. Apresenta a implementação de um sistema de reconhecimento contínuo da fala (*continuous speech recognition* - CSR) com amplo vocabulário para o português brasileiro.

Para isso, utilizou-se das ferramentas e bibliotecas disponibilizadas nos pacotes HTK (*Hidden Markov Model Toolkit*) e ATK (*Real-Time API for HTK*). O primeiro direcionado ao treinamento de modelos acústicos e linguísticos, enquanto o segundo aplicou-se aos testes do sistema CSR desenvolvido, por apresentar desempenho superior ao primeiro e possuir uma arquitetura orientada a objetos que opera em tempo real.

Com um foco mais voltado para as questões de segurança quanto ao acesso não autorizado, Uzunai e Bicakci (2007) desenvolveram um sistema de residência inteligente segura voltado para os pacientes com tetraplegia. Os autores concentram-se na preocupação com a segurança das aplicações de residências inteligentes envolvendo o controle e sistemas de informação para saúde.

O sistema é capaz de autenticar o usuário a partir da pronúncia de uma senha e o controle é proporcionado pela conversão de comandos de voz em padrões de comandos do teclado, através de um *middleware* que realiza a tradução de voz para texto. Foi empregado o protocolo X-10 para a comunicação com os dispositivos a serem controlados, substituindo os transmissores tradicionais por um *gateway* de autenticação, tendo sido ainda implementado um sistema de detecção de intrusão na comunicação entre os dispositivos X-10.

Em seu projeto para a implementação de um método de redução de ruídos baseado em subespaços vetoriais através de decomposições matriciais triangulares, Santos Júnior (2009) utilizou como cenário de simulação um sistema de reconhecimento da fala embarcado para o controle de funcionalidades em um automóvel, empregando o decodificador Julius para o reconhecimento de comandos no idioma inglês, uma vez que não foram encontrados dados gratuitos disponíveis para a modelagem no idioma português. Dessa forma, aplicou a seu cenário o modelo acústico disponibilizado pelo projeto VoxForge[8] . A definição do cenário culminou na especificação e criação de uma gramática, ilustrada no diagrama de comandos da Figura 7.

Com o intuito de permitir o controle de uma cadeira de rodas por indivíduos com paralisia, Qadri e Ahmed (2009) desenvolveram uma solução integrando uma interface de entrada e RAF para o controle dos movimentos do motor de passo anexo à cadeira. A interface se constitui de um microfone, ligado ao kit DSP (*Digital Signal Processor*) TMS320C6711[9] da Texas Instruments. A entrada de áudio analógica é processada pelo DSP, que realiza o reconhecimento da voz e, então, gera um sinal analógico específico para cada palavra reconhecida

[8] VOXFORGE. Disponível em: <http://www.voxforge.org/>. Acesso em: 11 jun. 2015.

[9] DSP Starter Kit (DSK) for TMS320C6711. Disponível em: <http://www.kanecomputing.co.uk/dsk_c6711.htm>. Acesso em: 12 jun. 2015.

Figura 7 – Diagrama de comandos no idioma inglês para o controle das funcionalidades de um automóvel.

AC	Plus Minus	One Two Three Four Five	Six Seven Eight Nine Ten	Eleven Twelve Thirteen Fourteen Fifteen	Sixteen Seventeen Eighteen Nineteen Twenty
	Turn		On	Off	
CD	Track	One Two Three Four Five	Six Seven Eight Nine Ten	Eleven Twelve Thirteen Fourteen Fifteen	Sixteen Seventeen Eighteen Nineteen Twenty
	Turn		On	Off	
Radio	Turn		On	Off	
Window	Left Right		Open	Close	

Fonte: Santos Júnior (2009)

em sua saída, o qual é digitalizado em seguida e transmitido a um microcontrolador PIC (*Peripheral Interface Controller*) 16F876A[10] , onde é implementado um código para o controle do motor de passo a partir de 5 comandos: *LEFT*, *RIGHT*, *REVERSE*, *FRONT* e *STOP*.

Como parte do esforço para o desenvolvimento de um sistema de reconhecimento automático da fala para o português brasileiro, Silva, Nelson Neto e Klautau (2009) apresentam contribuições quanto à geração de dados de texto, baseados na extração automática a partir de jornais, como também de dados de voz, baseados em *audiobooks*, para criação de modelos. Acrescenta ainda a geração de dados de voz para testes e avaliação do desempenho de sistemas. Adicionalmente, contribui ao apresentar técnicas de adaptação ao locutor, com a interessante característica de que todos os recursos produzidos são disponibilizados publicamente.

Pensando na necessidade de se disponibilizar interfaces de programação para o desenvolvimento de aplicações integrando o RAF, Nelson Neto et al. (2010) publicaram APIs (*Application Programming Interfaces*) criadas para contribuir com os programadores que desejam integrar ferramentas de reconhecimento automático da fala a seus programas, como é o caso da API Coruja[11] . Escrita em linguagem de programação C++, permite comunicação entre as linguagens suportadas pela plataforma Microsoft .NET.

[10] PIC16F876A. Disponível em: <http://www.microchip.com/wwwproducts/Devices.aspx?product=PIC16F876A>. Acesso em: 12 jun. 2015.

[11] LABORATÓRIO DE PROCESSAMENTO DE SINAIS. *Download de softwares*. Disponível em: <http://www.laps.ufpa.br/falabrasil/downloads.php>. Acesso em: 18 jun. 2015.

Proporciona o controle em tempo real do decodificador Julius e da interface de áudio do sistema operacional. A partir dessa API, é possível o interfaceamento de uma série de programas de computador para o controle por voz como, por exemplo o controle de uma apresentação de slides do *Microsoft Office PowerPoint*[12] através da aplicação PPTControler, desenvolvida pela equipe do referido projeto.

O desenvolvimento de *corpora* de áudio e texto, dicionário fonético, conversor grafema-fonema (*Grapheme to Phoneme* – G2P) e modelos acústico e de linguagem para o reconhecimento automático da fala no idioma português brasileiro é descrito em Nelson Neto et al. (2011). Os itens produzidos foram disponibilizados publicamente no repositório do projeto FalaBrasil[13] , em conjunto com uma interface de programação utilizada para a criação de diversas aplicações, inclusive um módulo de voz para o pacote *OpenOffice*[14] .

Descreve ainda uma aplicação que utiliza síntese e RAF associados em um módulo de processamento de linguagem natural, dedicado a tradução automática estatística. Foi produzido um dicionário fonético com 65.532 palavras, um conversor G2P baseado em regras, um *corpus* de texto com aproximadamente 672 mil sentenças formatadas, dois *corpora* de áudio com múltiplos locutores, correspondendo a aproximadamente 16,5 horas de áudio.

E ainda, *templates* de *software* para o projeto estatístico de modelos acústico e de linguagem, um modelo acústico independente de locutor no formato do HTK e um modelo de linguagem no formato trigrama da ARPA e uma API em código aberto para o controle do decodificador Julius, contendo um conversor de gramáticas do tipo SAPI XML (*eXtensible Markup Language*) para o tipo suportado pelo Julius.

A escassez de recursos de RAF para o português brasileiro têm motivado o desenvolvimento e disponibilização de ferramentas nessa área buscando fomentar a inovação com aplicativos para *desktop* e sistemas embarcados. Para tanto, Oliveira et al. (2011) descreve a concepção de uma interface de programação, chamada de JLaPSAPI, seguindo a especificação *Java Speech API* para o sistema Coruja e um modelo acústico no formato do CMU Sphinx.

Seu intuito é o de promover o surgimento de aplicações em dispositivos móveis. A utilização do modelo acústico criado, no entanto, apresentou uma alta taxa de erros, com o melhor resultado em 46,25% para reconhecimento contínuo da fala. Entretanto, o sistema se mostrou eficiente para aplicações com gramáticas de comando e controle.

Visando conceber a arquitetura de um sistema de reconhecimento contínuo da fala para o português brasileiro voltado para implementação em sistemas embarcados com recursos computacionais limitados, Silva (2011) reutilizou a arquitetura de pré-processamento do sinal de voz desenvolvida em seu trabalho de mestrado (SILVA, 2006 apud SILVA, 2011). A partir dela, aplicou a extração de parâmetros LPC (*Linear Predictive Coding*) e técnicas

[12] MICROSOFT Office Power Point. Disponível em: <https://products.office.com/pt-br/powerpoint>. Acesso em: 12 jun. 2015.

[13] FALABRASIL: Reconhecimento de Voz para o Português Brasileiro. Disponível em: <http://www.laps.ufpa.br/falabrasil/>. Acesso em: 12 jun. 2015.

[14] APACHE Open Office. Disponível em: <http://www.openoffice.org/pt-br/>. Acesso em: 12 jun. 2015.

derivadas dessa análise para investigar o conjunto de características acústicas mais adequado à criação de modelos. Empregou também um método estatístico baseado nos HMMs para geração do modelo acústico a ser utilizado.

Manteve sua investigação atenta à relação entre a configuração dos modelos acústico e de linguagem e a eficiência e custo computacional do reconhecimento. Em suas contribuições destaca a avaliação dos parâmetros e configuração de um sistema CSR utilizando um número reduzido de coeficientes MFCC (*Mel Frequency Cepstral Coefficient*). Além disso, ressalta as simplificações alcançadas para o sistema proposto, bem como o detalhamento do fluxo de desenvolvimento proporcionado pela arquitetura definida.

Ao desenvolver um sistema de controle do ambiente utilizando o padrão 802.15.4 para redes WPAN (*Wireless Personal Area Networks*), Burda e Wietfeld (2007) investigou a capacidade de transmissão de dados multimídia através dessas redes, tendo obtido sucesso no envio de mensagens de voz e fluxo de áudio contínuo através delas. Seguindo essa linha da automação residencial sem fio, Alshu'eili e Gupta (2011) descrevem a concepção de um sistema de automação residencial sem fio (*Wireless Home Automation System* – WHAS) baseado em reconhecimento de comandos da fala para o controle da iluminação e eletrodomésticos.

Nesse último trabalho, o áudio digitalizado é enviado a partir do módulo microfone, através da rede *ZigBee*[15] , para um microcontrolador central. Posteriormente, é processado em um computador para gerar os comandos convertidos em texto. O RAF empregado utiliza a interface de programação *Microsoft Speech API*, não tendo sido especificado a ferramenta de reconhecimento utilizada.

No âmbito dos dispositivos para o controle a partir de comandos de voz, Qidwai e Shakir (2012) apresenta uma interface de reconhecimento automático da fala no idioma árabe para o auxílio de pessoas com deficiência. O sistema consiste de um microfone padrão ligado ao módulo de reconhecimento de voz EasyVR[16] , um microcontrolador Atmega8[17] da Atmel, uma placa de circuito impresso (PCI) com indicadores e pinos de controle digital e uma fonte de tensão regulada em 5V. A interface foi aplicada e testada em três diferentes estudos de caso: para o controle de *mouse* e teclado de um PC, para o controle de uma cadeira de rodas e para o controle de um braço robótico industrial.

Lu e Chen (2012) discutem o sistema de controle de uma cadeira de rodas versátil para utilização por tetraplégicos. Foi desenvolvido um protótipo utilizando o kit *LEGO Mindstorms NXT*[18] com variadas modalidades de entrada. Para a entrada de comandos por voz, buscou-se utilizar comandos independentes de idioma, tendo sido empregado o sensor de sons LEGO (*LEGO Sound Sensor*).

Com o propósito de proporcionar às pessoas portadoras de necessidades especiais o

[15] ZIGBEE Alliance. Disponível em: <http://www.zigbee.org/>. Acesso em: 12 jun. 2015.
[16] EASYVR 3. Disponível em: <http://www.veear.eu/products/easyvr3/>. Acesso em: 12 jun. 2015.
[17] ATMEGA8. Disponível em: <http://www.atmel.com/devices/atmega8.aspx>. Acesso em: 12 jun. 2015.
[18] LEGO Mindstorms NTX 2.0. Disponível em: <http://shop.lego.com/en-US/LEGO-MINDSTORMS-NXT-2-0-8547>. Acesso em: 12 jun. 2015.

uso do computador, o sistema de RAF Simon[19] , desenvolvido pela organização Simon Listens, permite o controle de diversos programas e funcionalidades, dentre as quais: o navegador de internet, clientes de email, programas para apresentação de slides, reprodução de músicas e vídeos, dentre outros. Inclui um teclado virtual controlado por voz e a numeração de links em páginas web e de funções de *webmail* para acioná-las através da voz.

O desenho de uma grade dividindo a área de trabalho do computador permite o controle do mouse por comandos de voz, como também uma calculadora controlada através dos comandos pronunciados pelo usuário é disponibilizada. Para tanto, apresenta em sua suíte de componentes as ferramentas de RAF CMU Sphinx e Julius com o *Hidden Markov Toolkit* (HTK) (BENTHIN, 2013).

Nessa linha de *software* livre e de código aberto, o grupo de pesquisa FalaBrasil da Universidade Federal do Pará, vem desenvolvendo o SimonBR[20] , disponibilizando para o programa Simon: modelo acústico, dicionário fonético, interfaces traduzidas e cenários específicos para o português brasileiro.

Batista (2013) descreve a criação de bases de áudio transcrito e API para o desenvolvimento de aplicações, bem como a utilização dessa API para a integração de RAF a dois casos distintos: uma aplicação de ditado e edição de textos para a ferramenta *Writer*[21] do pacote *LibreOffice* em sistema *desktop*, chamado de *SpeechOO*[22] , e outra para o reconhecimento automático da fala em um *call center* através da integração da API Coruja com o *software* PBX (*Private Branch Exchange*) Asterisk.

Desenvolveu uma URA (Unidade de Resposta Audível) para o atendimento automático por RAF. O autor apresenta alguns resultados para o funcionamento da URA. Primeiramente, utilizando um *softphone* com *headset* para realização da chamada, alcançando 80% de acerto para vozes masculinas e 78% de acerto para vozes femininas. Em seguida, é utilizado um telefone VoIP fixo para realização da chamada, obtendo-se uma taxa de 70% de acerto para voz feminina. Finalmente, realizando a chamada a partir de um aparelho celular GSM (*Global System for Mobile*), se alcançou uma taxa de 98% de acerto para voz feminina.

Em Pimentel et al. (2014) é apresentada uma plataforma para o reconhecimento de comandos por voz em sistemas embarcados para tecnologias assistivas. Foi utilizado o módulo de reconhecimento de voz *Voice Recognition Module V2*[23] e a plataforma de desenvolvimento *Beaglebone Black*[24] . O sistema operacional configurado foi o Ubuntu Linux, tendo sido implementado um aplicativo, escrito em linguagem Python[25] , capaz de proporcionar a validação da plataforma para o acendimento de um LED (*Light Emitting*

[19] SIMON LISTENS. *Simon listens*: non-profit organization for research and apprenticeship. 2012. Disponível em: <http://simon-listens.org/index.php?id=122&L=1>.

[20] SIMONBR. Disponível em: <http://www.laps.ufpa.br/falabrasil/simonbr.php>. Acesso em: 12 jun. 2015.

[21] WRITER. Disponível em: <https://www.libreoffice.org/discover/writer/>. Acesso em: 12 jun. 2015.

[22] SPEECHOO. Disponível em: <http://www.laps.ufpa.br/falabrasil/speechoo.php>. Acesso em: 12 jun. 2015.

[23] VOICE Recognition Module V2. Disponível em: <http://www.elechouse.com/elechouse/index.php?main_page=product_info&cPath=&products_id=2151>. Acesso em: 12 jun. 2015.

[24] BEAGLEBONE Black. 2015. Disponível em: <http://beagleboard.org/black>. Acesso em: 12 jun. 2015.

[25] PYTHON. Disponível em: <https://www.python.org>. Acesso em: 12 jun. 2015.

Diode) RGB (*Red, Green and Blue*). O autor destaca o baixo custo e alta portabilidade da plataforma, que pode ser facilmente adaptável a uma cabeceira de cama ou a uma cadeira de rodas. Os testes realizados apresentaram uma taxa de 95,9% de acertos, para amostras de pronúncias com vozes masculinas e femininas.

Trabalhando com uma linha de pesquisa em tecnologias da fala que vem tornando realidade a comunicação entre as pessoas e os dispositivos ao seu redor (em casa, no carro, etc), bem como com os dispositivos que as vestem (um relógio, por exemplo) e dispositivos que elas usam no cotidiano (como um celular ou um *tablet*), o grupo de pesquisa em processamento da fala da empresa Google® focam-se em uma característica que consideram fazer da Google® uma empresa única: a computação de dados em larga escala.

Partindo, então, do uso desses poderosos recursos computacionais, esse grupo desenvolve e repensa a arquitetura e algoritmos de reconhecimento automático da fala, com o emprego de métodos que, no passado vinham sendo considerados inviáveis em função dos elevado custo para sua implementação. Associado a isso, fazem uso de computação em *cluster* e computação paralela, uma vez que o processamento da fala é uma área da computação que exige um grande volume de dados (GOOGLE, 2016c).

A integração entre trabalho desenvolvido ao longo dos anos pelo referido grupo e o trabalho produzido nos outros diversos grupos da empresa – distribuídos em 15 áreas de pesquisa e desenvolvimento – é capaz de proporcionar à sociedade modernas aplicações como, por exemplo o aplicativo Google Now®[26] .

Esse *software* se constitui em um assistente pessoal inteligente, desenvolvido pela empresa Google®, o qual possui uma interface que utiliza linguagem natural para responder questionamentos do usuário, fazer recomendações, e executar ações e pedidos de forma integrada a um conjunto de serviços *web*.

A página do referido grupo de pesquisa na *web*[27] , disponibiliza cerca de 159 publicações produzidas ao longo dos inúmeros trabalhos desenvolvidos na área de RAF. Podemos citar ainda as mais de 288 publicações disponibilizadas na página de outro grupo de pesquisa da empresa: o de processamento de linguagem natural[28] .

2.3 Discussão

Diante de um cenário de envelhecimento populacional crescente, bem como da realidade enfrentada pelas pessoas que apresentam restrições em sua mobilidade, as tecnologias assistivas surgem como uma tendência no melhoramento das capacidades funcionais desses indivíduos.

[26] GOOGLE. *Google Now*. 2016. Disponível em: <https://www.google.com/landing/now/>. Acesso em: 05 jan. 2016.

[27] GOOGLE. *Speech Processing Group*. 2016. Disponível em: <http://research.google.com/pubs/SpeechProcessing.html>. Acesso em: 05 jan. 2016.

[28] GOOGLE. *Natural Language Processing Group*. 2016. Disponível em: <http://research.google.com/pubs/NaturalLanguageProcessing.html>. Acesso em: 05 jan. 2016.

O provimento de acessibilidade às novas tecnologias tem levado ao desenvolvimento de diversos dispositivos e sistemas de TA. É apresentado um conjunto amostral desses dispositivos e sistemas, observando uma sequência evolutiva desde os equipamentos puramente mecânicos até interfaces alternativas de comando remoto suportadas por modernos servidores de comunicação.

Dentre as capacidades de interfaceamento disponíveis para a interação humano-computador, destaca-se as aplicações e tecnologias de comandos por voz. É, então, delineado um apanhado histórico da evolução dos sistemas de comandos por voz, até os modernos sistemas de reconhecimento automático da fala.

O constante esforço de desenvolvimento visando a inovação de aplicações integrando-as aos sistemas de RAF é, então, representado na descrição de inúmeros projetos, bem como apresentando-se seus mais relevantes resultados. O levantamento realizado neste capítulo aponta as atuais tendências de convergência tecnológica, observadas na integração entre diferentes dispositivos e plataformas de comunicação para a implementação de aplicações de comando e controle nos mais diversos cenários.

Observa-se duas linhas de desenvolvimento presentes ao longo desse estudo: a primeira baseada na utilização de módulos de *hardware* específicos que agregam funcionalidades de reconhecimento da fala com recursos limitados para a execução de comandos, enquanto a segunda contempla ferramentas de *software* para o desenvolvimento de aplicações de reconhecimento de fala contínua em PCs ou embarcadas em plataformas com arquiteturas de *hardware* otimizadas.

Trazendo um resumo sobre as aplicações e tecnologias de comandos por voz, a Tabela 2 resume brevemente o histórico do reconhecimento da fala até a década de 90, enquanto as Tabelas 3 e 4 trazem, a partir daí, as principais informações sobre pesquisas desenvolvidas no âmbito do RAF referenciadas no presente capítulo.

Assim, com o intuito de se formar uma base inicial de conhecimento no desenvolvimento de aplicações com reconhecimento de comandos por voz, foi realizado neste trabalho um experimento piloto que consistiu na implementação de uma aplicação de comandos por voz utilizando um módulo de *hardware*, conforme descrito no Capítulo 4, seção 4.2.

A plataforma embarcada de reconhecimento automático da fala para o auxílio de pessoas com mobilidade reduzida proposta nesta pesquisa é baseada na utilização de ferramentas de *software*, descritas no Capítulo 3, embarcadas em uma placa de desenvolvimento baseada em processador ARM (*Advanced RISC Machine*). Sua implementação e aplicação estão descritas no Capítulo 4, seção 4.3.

No capítulo seguinte é apresentada uma visão geral sobre o RAF e são descritas ferramentas de *software* para o reconhecimento automático da fala, destacando-se as utilizadas no desenvolvimento da interface proposta no objetivo principal desta pesquisa.

Tabela 2 – Quadro resumo com breve histórico sobre o reconhecimento da fala.

Década	**Marcos de desenvolvimento**
20	• Radio Rex
30 e 40	• Compressão de voz e técnicas de análise da fala
50	• Primeiros sistemas computadorizados reconhecem dígitos (AT&T) • Uso de fonemas como unidades linguísticas básicas • Reconhecimento de vogais com 93% de acurácia (MIT)
60	• Sistema de reconhecimento com vocabulário de 50 palavras (MIT)
70	• Modelos Ocultos de Markov • HEARSAY-I, DRAGON, HEARSAY-II e HARPY (CMU) • Reconhecimento de sentenças (1011 palavras com 95% de acurácia)
80	• Lançamento do SPHINX (CMU) • Desenvolvimento da primeira versão do HTK (CUED)
90	• P&D em RAV dominado pelas empresas privadas • Em 1997 tem início o desenvolvimento do LVCSR Julius

Fonte: Adaptado de Spaans (2004).

Tabela 3 – Quadro resumo com pesquisas desenvolvidas no âmbito do reconhecimento automático da fala.

Autoria (Ano)	Tecnologias empregadas	Resultados alcançados
Aguilera et al. (1992)	Computador pessoal. Módulo *Intravoice* VI.	Decodificação da voz em comandos do teclado. Atender o telefone, controlar luzes e eletrodomésticos.
Jiang et al. (2000)	Módulo HM2007. Microcontrolador MC68HC811 (Freescale). Multiplexadores.	Dispositivo portátil para auxiliar pessoas com deficiência no controle do ambiente através de comandos por voz.
Asahi Kasei (2000)	Modelo acústico e rede de vocabulário proprietários. *Front-end* para tratamento de ruído e cancelamento de eco.	*Middleware* de reconhecimento de voz para sistemas embarcados. Navegação de automóveis, *smartphones*.
Hall e Molloy (2003)	Computador pessoal. *Software Dragon Naturally Speaking*.	Unidade de baixo custo para o controle do ambiente pela voz (menos de $500 dólares).
Sampaio Neto (2006)	*Speech* API da Microsoft®. Java *Speech* API.	Aplicativo para o ensino de inglês. Reconhecimento de voz integrado ao *chat* IRC.
Tevah (2006)	HTK e ATK.	Sistema LVCSR para o português brasileiro.
Uzunai e Bicakci (2007)	*Software* cliente/servidor. *Middleware* de reconhecimento de voz. *Gateway* de autenticação.	Decodificação da voz em comandos do teclado. Sistema de automação residencial seguro ativado por voz para pessoas com deficiência.
Santos Júnior (2009)	Subespaços vetoriais. Julius. Modelos acústicos do Voxforge.	Redução de ruído. Controle das funcionalidades de um automóvel (idioma inglês).
Qadri e Ahmed (2009)	DSP TMS320C6711 (Texas). Microcontrolador PIC.	Interface de voz para o controle de uma cadeira de rodas.
Silva, Nelson Neto e Klautau (2009)	Extração automática de texto. *Audiobooks*. HTK. SRILM.	Corpora de voz digitalizada e texto. Técnicas de adaptação ao locutor. Modelos acústicos e de linguagem.
Nelson Neto et al. (2010)	C++. Julius.	API Coruja. PPTControler.
Nelson Neto et al. (2011)	HTK. SRILM. *Corpus Spoltech*, *West Pointer* e CETEN-Folha. C++.	Corpora de texto e áudio, dicionário fonético, conversor grafema-fonema, modelos acústicos e de linguagem.
Oliveira et al. (2011)	API Coruja. Java *Speech* API. CMU PocketSphinx. Corpora LapsStory, *West Pointer*, *Spoltech* e LapsBenchmark. SphinxTrain.	API Java para utilização do Coruja. Modelo acústico para o CMU Sphinx em português brasileiro.
Silva (2011)	Análise LPC e técnicas derivadas (MFCC). HMMs com Densidades Contínuas. Base de dados Ynoguti e Violaro. CMU Sphinx. FPGA.	Modelagem da arquitetura de um sistema embarcado de reconhecimento de fala contínua em *hardware*.

Fonte: Autoria própria.

Tabela 4 – Continuação do quadro resumo com pesquisas desenvolvidas no âmbito do reconhecimento automático da fala.

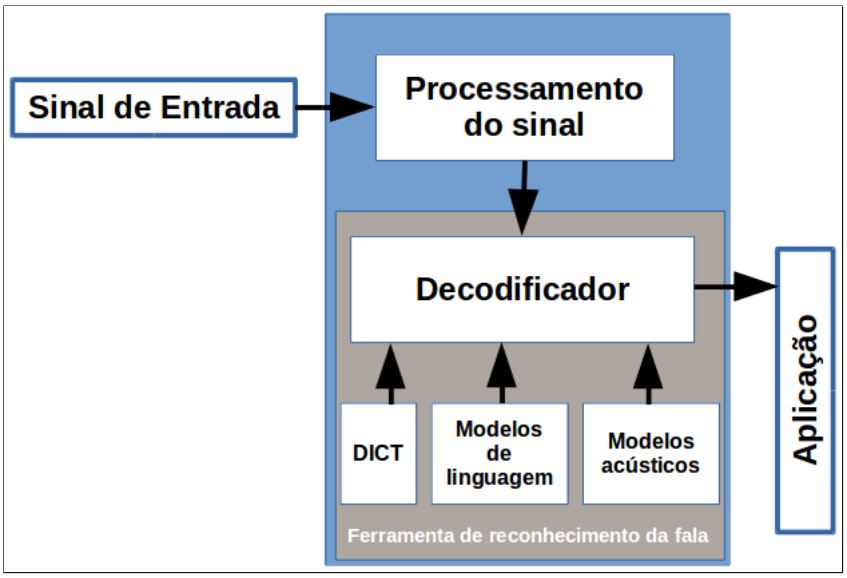
Autoria (Ano)	**Tecnologias empregadas**	**Resultados alcançados**
Alshu'eili e Gupta (2011)	*Speech* API da Microsoft®. Redes ZigBee.	Sistema de automação residencial sem fio baseado em reconhecimento de voz.
Qidwai e Shakir (2012)	Módulo EasyVR. Microcontrolador Atmega8.	Controle por voz de: cadeira de rodas, mouse e teclado, braço robótico.
Simon listens (2012)	CMU Sphinx. Julius. HTK.	Controle de programas e funcionalidades do computador.
Batista (2013)	Java *Speech* API. Libre *Office Writer*. Julius. Python. Asterisk. DigiVoice VB0408PCI. Dados de áudio e texto do projeto FalaBrasil. HTK. SRILM.	Melhoramento dos modelos acústico e de linguagem. Extensão de ditado para o Libre *Office Writer*. Reconhecimento automático de voz em *call* center.
Google (2016a)	*Deep Neural Networks*. *Custering* e processamento paralelo. Processamento de linguagem natural.	Assistente pessoal inteligente Google Now®

Fonte: Autoria própria.

3 Ferramentas de reconhecimento da fala

O reconhecimento automático da fala (ASR) se configura essencialmente em um processo de mapeamento de um sinal acústico – captado por um microfone, telefone ou outro transdutor acústico – em uma sequência de informações discretas, como fonemas ou palavras. Na Figura 8, são ilustrados os componentes que constituem um sistema de reconhecimento da fala típico (SPAANS, 2004).

Figura 8 – Sistema de reconhecimento da fala típico.



Fonte: Adaptado de Spaans (2004)

O processamento do sinal, também conhecido como estágio de *front-end*, se constitui em um processo de conversão dos sinais de voz para uma representação através de um vetor de características, podendo ser utilizadas diferentes abordagens para a obtenção desses parâmetros. As técnicas que têm se mostrado eficientes para execução desse processamento são: a análise de banco de filtros, LPC e os MFCC (GOMÉZ, 2011; SILVA, 2011; BATISTA, 2013).

As ferramentas de reconhecimento da fala (*Speech Recognition Engines* – SREs) são constituídas de modelo de linguagem (ou gramática), modelo acústico, dicionário fonético (DICT) e decodificador.

Os modelos de linguagem se referem às palavras que o sistema é capaz de reconhecer, quais delas são suscetíveis de aparecerem juntas e em qual ordem. Para tanto, contém uma vasta lista de palavras, sendo mais utilizados em aplicações de ditado. No caso de aplicações de Unidade de Resposta Audível (URA) ou aplicações de comando e controle, esses modelos são chamados gramáticas, constituindo-se de arquivos com conjuntos menores ou combinações

predefinidas de palavras. Em ambos os casos, cada palavra é associada a uma lista de fonemas (dicionário fonético) que correspondem aos distintos sons que a formam. Esses modelos levam em consideração regras de gramática e semântica (SPAANS, 2004; VOXFORGE, 2016).

Os modelos acústicos registram o conhecimento sobre a acústica, a fonética, a variabilidade do sinal, etc. Normalmente, contém uma representação estatística – em geral apresentada por modelos ocultos de Markov – dos distintos sons que formam as palavras (SPAANS, 2004; VOXFORGE, 2016).

Já o decodificador utiliza os modelos acústicos e de linguagem para gerar a sequência de palavras mais provável que represente um dado vetor de características de entrada. Ou seja, ele recebe o vetor de características do fluxo de áudio na entrada e compara com os modelos acústicos de sua base de dados, localizando os fonemas que apresentam maior correspondência com essa entrada até encontrar uma pausa (silêncio). Em seguida, realiza uma busca nos modelos de linguagem e retorna o texto correspondente (SPAANS, 2004; VOXFORGE, 2016).

Assim, o dicionário fonético anteriormente mencionado define o vocabulário que será reconhecido no modelo de linguagem após a conversão da entrada de áudio numa sequência fonética. Nesse dicionário é possível que uma mesma palavra seja armazenada mais de uma vez, possibilitando o reconhecimento de diferentes pronúncias da mesma palavra (LEE, 2010).

Nas seguintes seções do capítulo serão apresentadas uma ferramenta para a criação de modelos acústicos e duas ferramentas de reconhecimento da fala contínua, sendo dada certa ênfase à caracterização da ferramenta Julius, utilizada nesse projeto.

3.1 Criação de modelos acústicos

Para a utilização de comandos através da fala, os modelos acústicos devem representar o mais fidedignamente possível os sinais acústicos dos comandos recebidos na entrada do sistema. Tais modelos são gerados a partir de uma complexa análise estatística representada nos modelos acústico e de linguagem, bem como no dicionário fonético.

Uma referência para a representação estatística dos modelos acústico e de linguagem são os HMMs, os quais representam o sinal de voz digitalizado numa distribuição de parâmetros de modo a maximizar a probabilidade de se encontrar uma determinada sequência de fonemas presente no modelo acústico que seja semelhante à sequência de fonemas do vetor de entrada (BATISTA, 2013).

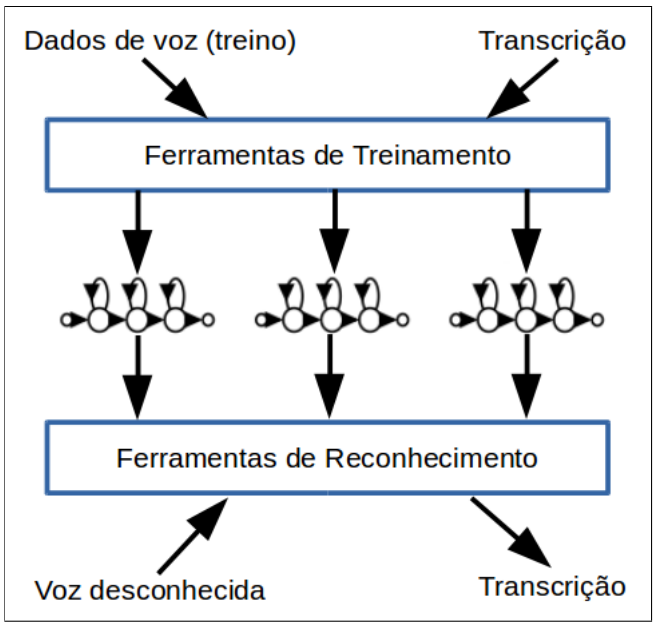
3.1.1 *Hidden Markov Model Tookit* (HTK)

O *toolkit* de modelos ocultos de Markov, HTK, mantido pelo departamento de engenharia da universidade de Cambridge (*Cambridge University Engineering Department* – CUED), consiste de ferramentas para a criação e manipulação de HMMs. Esse pacote pode

ser utilizado para modelar quaisquer séries temporais, tendo sido, no entanto, seu projeto voltado principalmente para o suporte à construção de ferramentas de processamento de sinais de voz baseados nos modelos ocultos de Markov (GOMÉZ, 2011).

Nesse sentido, o HTK apresenta dois estágios de processamento principais, conforme ilustrado na Figura 9. O primeiro é associado a ferramentas de treinamento para estimação dos parâmetros de um conjunto de HMMs a partir de elocuções de treino associadas a suas respectivas transcrições. Já o segundo estágio está relacionado a uma ferramenta de transcrição de elocuções desconhecidas a partir do emprego das ferramentas de reconhecimento (YOUNG et al., 2009; SILVA, 2010).

Figura 9 – Principais estágios de processamento do HTK.



Fonte: Adaptado de Young et al. (2009), Silva (2010)

Apesar de ser um *software* disponibilizado publicamente após registro do usuário, sua distribuição é restrita ao uso pessoal e para fins de pesquisa, conforme a licença de registro para utilização do *software* (HTK3, 2015). No entanto, uma vez que o HTK tenha sido utilizado para o treinamento de modelos, as restrições de sua licença não se aplicam aos modelos criados.

3.1.1.1 Processamento e codificação da fala

O *toolkit* HTK é capaz de receber os dados de entrada com o áudio da fala de três formas distintas (YOUNG et al., 2009)

- A partir de um arquivo com os parâmetros da fala previamente codificada.

- A partir de um arquivo de áudio, que será codificado durante o processamento de entrada.

- Diretamente a partir de um dispositivo de áudio, sendo a fala codificada durante o processamento de entrada.

O *software* integra módulos que dão suporte ao pré-processamento do sinal baseado em predição linear (LPC), processamento baseado na transformada rápida de Fourier (*Fast Fourier Transform* – FFT) e quantização vetorial. Quando aplicável, os dados do áudio de entrada são convertidos na forma de vetores de parâmetros característicos, que são a unidade básica de dados processada pelas ferramentas de treino e reconhecimento do HTK (SPAANS, 2004).

Possui ainda formas de se ampliar os parâmetros que caracterizarão o sinal da fala através de medidas da energia, coeficientes delta e coeficientes de aceleração (delta-delta), permitindo ainda a divisão de cada vetor de parâmetros em múltiplos fluxos de dados para gerar *observações* (YOUNG et al., 2009).

Para tanto, é necessário especificar em um arquivo de configuração os valores referentes aos parâmetros que se deseja utilizar. A ideia central é a de que existe um tipo de parâmetro fonte (*SOURCEKIND*) e um tipo de parâmetro alvo (*TARGETKIND*). O primeiro se refere ao formato natural dos dados de entrada, enquanto o segundo se refere ao formato de dados requirido pelo HTK. Young et al. (2009) apresenta uma descrição detalhada das ferramentas que constituem o pacote, bem como dos parâmetros a serem configurados.

O processo geral de entrada de dados é apresentado na Figura 10, que ilustra o sinal de áudio da fala amostrado sendo convertido em uma sequência de blocos de amostras (SPAANS, 2004).

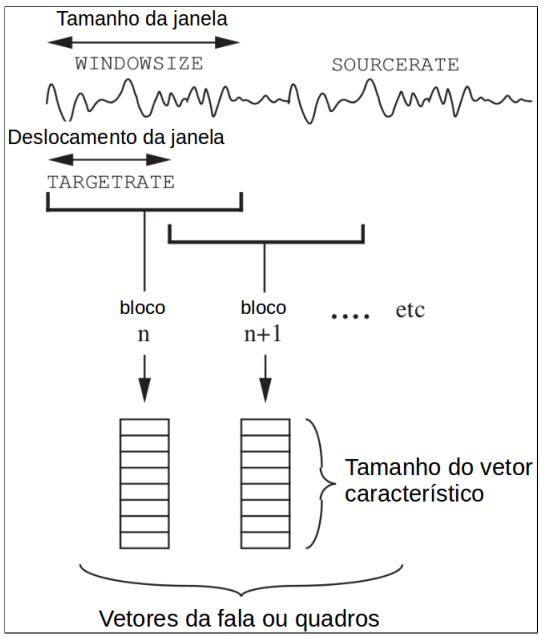
A taxa de amostragem do sinal de entrada é normalmente determinada pelo próprio arquivo de entrada. No entanto, pode ser definida explicitamente através do parâmetro de configuração *SOURCERATE*.

O tamanho do segmento da forma de onda original que define cada vetor característico é usualmente chamado de janela (RABINER; SCHAFER, 1978; PROAKIS; MANOLAKIS, 1996), sendo configurado através do parâmetro *WINDOWSIZE*. O período entre o início de cada vetor de parâmetros e o seu próximo determina o deslocamento da janela, que deve ser informado ao *software* através do parâmetro de configuração *TARGETRATE*.

Normalmente, o tamanho da janela é maior que o seu deslocamento, de modo que haverá uma sobreposição (*overlap*) entre janelas sucessivas. Os valores desses parâmetros são dados em unidades de 100ns.

É uma prática comum aplicar-se uma pré-ênfase ao sinal de entrada através do emprego da equação de diferenças de primeira ordem (Equação 3.1) para as amostras $\{s_n, n = 1, N\}$ em cada janela.

Figura 10 – Processo de codificação da fala.



Fonte: Adaptado de Young et al. (2009), Spaans (2004)

$$s'_n = s_n - k s_{n-1} \tag{3.1}$$

O termo k é o coeficiente de pré-ênfase, tal que $0 \leq k < 1$. Sua configuração pode ser informada através do parâmetro *PREEMCOEF*.

Além disso, através do parâmetro *USEHAMMING* é possível se aplicar a janela de Hamming, dada pela Equação 3.2, às amostras $\{s_n, n = 1, N\}$ com o intuito de suavizar as descontinuidades no janelamento (RABINER; SCHAFER, 1978; PROAKIS; MANOLAKIS, 1996).

$$s'_n = \left\{0.54 - 0.46\cos\left(\frac{2\pi\,(n-1)}{N-1}\right)\right\} s_n \tag{3.2}$$

3.1.1.2 Análise espectral dos dados da fala

O HTK suporta tanto a análise espectral baseada em banco de filtros utilizando a transformada rápida de Fourier (FFT), como a análise baseada nos coeficientes de predição linear (LPC) (RABINER; SCHAFER, 1978; DELLER JR; HANSEN; PROAKIS, 1993).

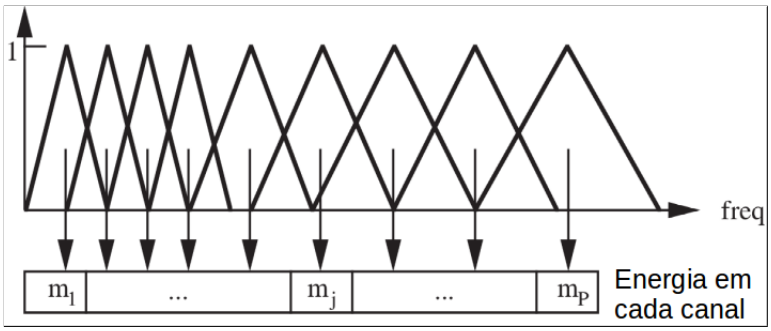
O ouvido humano é capaz de lidar com a não-linearidade espectral do áudio, e evidências empíricas sugerem que o emprego de um *front-end* que possa lidar de maneira semelhante com essas não-linearidades melhora o desempenho do reconhecimento. Nesse sentido, a análise baseada em banco de filtros é uma interessante alternativa à LPC, uma

vez que a primeira provê um caminho muito mais simples para a obtenção da resolução não-linear de frequências desejada (YOUNG et al., 2009).

Entretanto, as amplitudes do banco de filtros são altamente correlacionadas e, por isso, o uso da transformação cepstral nesse caso é virtualmente obrigatória no caso de os dados serem utilizados em um reconhecedor baseado nas HMMs com covariâncias diagonais (DELLER JR; HANSEN; PROAKIS, 1993; YOUNG et al., 2009).

O HTK provê um banco de filtros baseado na transformada rápida de Fourier (FFT) projetado para dar uma resolução aproximadamente similar, em uma escala Mel (FURUI, 1993; SPAANS, 2004; YOUNG et al., 2009). Na Figura 11 é ilustrada a forma geral desse banco de filtros. Como pode ser visto, os filtros utilizados são triangulares e igualmente espaçados ao longo da escala Mel, que é definida pela Equação 3.3.

Figura 11 – Banco de filtros triangulares na escala Mel.



Fonte: Adaptado de Young et al. (2009)

$$Mel\left(f\right) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{3.3}$$

Para implementar esse filtro, a janela de dados da fala é transformada utilizando-se a FFT e, então, sua magnitude é registrada. Os coeficientes de magnitude são posteriormente "encaixotados"(*binned*), correlacionando-os com cada filtro triangular. Aqui, o "encaixotamento"significa que cada coeficiente de magnitude da FFT é multiplicado pelo ganho do filtro correspondente e o resultado é acumulado. Assim, cada "caixa"armazena um somatório de pesos representando a magnitude espectral no respectivo canal do banco de filtros (RABINER; SCHAFER, 1978; DELLER JR; HANSEN; PROAKIS, 1993; PROAKIS; MANOLAKIS, 1996; FURUI, 1993).

Frequentemente, a utilização dos parâmetros cepstrais é recomendada (DELLER JR; HANSEN; PROAKIS, 1993; FURUI, 1993; YOUNG et al., 2009), sendo isso indicado ao *software* configurando-se o *TARGETKIND* com o valor *MFCC* para os Coeficientes Cepstrais em Frequência-Mel. Esse coeficientes são calculados a partir das amplitudes $\{m_j\}$ do banco de filtros com o emprego da transformada discreta do cosseno, conforme a Equação 3.4.

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}\left(j - 0.5\right)\right) \tag{3.4}$$

Sendo N o número de canais no banco de filtros, que pode ser configurado através do parâmetro *NUMCHANS*. A quantidade de coeficientes cepstrais a serem gerados pode ser definida no parâmetro de configuração *NUMCEPS*, podendo ser diferente do número de coeficientes do filtro.

A principal vantagem dos coeficientes cepstrais é que eles são, geralmente, descorrelacionados, o que permite a utilização de covariâncias diagonais nos HMMs. Contudo, um problema menor desses coeficientes é que os coeficientes cepstrais de mais alta ordem são numericamente bem pequenos, e isso resulta em uma faixa muito ampla de variância desde os coeficientes cepstrais de ordem mais baixa até os de mais alta ordem (FURUI, 1993; YOUNG et al., 2009).

É, então conveniente realizar-se um reescalonamento para que os coeficientes cepstrais tenham magnitudes similares. Isso pode ser aplicado aos MFCCs, através do parâmetro de configuração *CEPLIFTER*, para algum valor L de deslocamento, de acordo com a Equação 3.5 (YOUNG et al., 2009).

$$c'_n = \left(1 + \frac{L}{2} \sin\frac{\pi n}{L}\right) c_n \tag{3.5}$$

Segundo Young et al. (2009), os MFCCs são o tipo de parametrização empregada em muitas aplicações de reconhecimento da fala. Eles fornecem uma boa discriminação e podem ser utilizados em uma série de manipulações. Em particular, o efeito de inserir-se a fala de entrada através de um canal de transmissão é dado pela multiplicação entre o espectro da fala e a função de transferência do canal. Já no domínio cepstral logarítmico, essa multiplicação se torna uma simples adição, que pode ser removida pela subtração da média cepstral em todos os vetores de entrada (DELLER JR; HANSEN; PROAKIS, 1993; FURUI, 1993; SPAANS, 2004).

Na prática, claramente, a média deve ser estimada a partir de uma quantidade limitada de dados da fala, de modo que a subtração não será perfeita. Mesmo assim, essa simples técnica é muito efetiva na prática, uma vez que permite a compensação em longo prazo dos efeitos espectrais, como aqueles causados por diferentes microfones e canais de áudio (YOUNG et al., 2009).

Para executar essa normalização da média cepstral (*Cepstral Mean Normalization* – CMN) nos coeficientes MFCCs, basta adicionar o qualificador *_Z* ao valor do parâmetro *TARGETKIND* informado. Essa compensação não é suportada para o caso de entrada de áudio diretamente a partir de um dispositivo de áudio.

Além disso, pode-se ainda adicionar o qualificador *_0* ao valor do parâmetro *TARGETKIND* para agregar o coeficiente cepstral C_0 aos dados, provendo informação simplificada em termos de energia do sinal (NELSON NETO et al., 2011; YOUNG et al., 2009).

Por fim, o desempenho do sistema de reconhecimento da fala pode ser grandemente aperfeiçoado adicionando-se as derivadas aos parâmetros estáticos básicos. Por exemplo, a adição do qualificador *_D* ao valor do parâmetro *TARGETKIND* indica que os coeficientes de regressão de primeira ordem (coeficientes delta) serão agregados aos dados de parametrização. Esse coeficientes delta são computados através do emprego da Equação 3.6 (YOUNG et al., 2009).

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta \left(c_{t+\theta} - c_{t-\theta}\right)}{2\sum_{\theta=1}^{\Theta} \theta^2} \tag{3.6}$$

Em que, d_t é um coeficiente delta computado no tempo t em termos dos coeficientes estáticos $c_{t-\Theta}$ a $c_{t+\Theta}$. O valor de Θ, que refere-se ao tamanho da janela delta, pode ser configurado através do parâmetro de configuração *DELTAWINDOW*.

Uma vez que a Equação 3.6 depende de valores passados e futuros dos parâmetros estáticos da fala, é necessário se adequar os coeficientes no início e no final da fala. Uma solução padrão consiste em se replicar o primeiro ou o último vetor, conforme necessário, de modo a preencher a janela de regressão.

Quando o coeficiente delta é utilizado, ele é computado para todos os parâmetros estáticos presentes, incluindo a energia. Em algumas aplicações, no entanto, a energia absoluta não é util, mas sua derivada no tempo pode ser. Através da inclusão do qualificador *_N* no valor do parâmetro *TARGETKIND* a informação sobre a energia absoluta é suprimida, permanecendo somente o coeficiente delta da energia (YOUNG et al., 2009).

3.1.1.3 Criação dos modelos da fala

O princípio funcional do HTK é manipular conjuntos de HMMs. Foi projetado inicialmente para modelagem de parâmetros contínuos utilizando distribuições de saída multivariada de densidade contínua. Também é capaz de manipular sequências de observações consistindo de símbolos discretos em que as distribuições de saída são probabilidades discretas (SILVA, 2010).

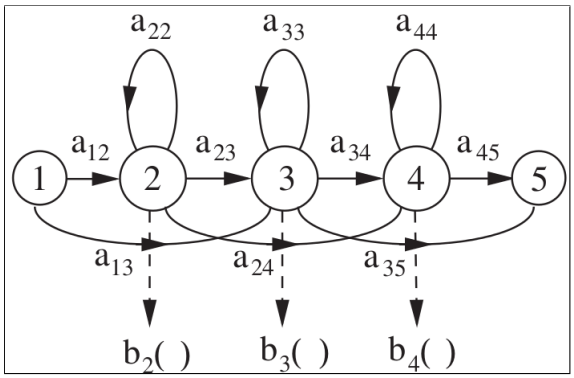
As funções de densidade contínua podem ser apresentadas na forma de modelos de misturas Gaussianas (*Gaussian Mixture Models* – GMMs), ou por modelos de redes neurais artificiais (Artificial Neural Networks – ANNs), que são respectivamente denotados por GMM-HMM e ANN-HMM (DELLER JR; HANSEN; PROAKIS, 1993; YOUNG et al., 2009).

Um HMM consiste de um determinado número de estados. Cada estado tem uma distribuição de probabilidades observável $b_j(\mathbf{o}_t)$ que determina a probabilidade de se gerar a observação $\mathbf{o}_t$ no tempo t e cada par de estados i e j possuem uma transição provável a_{ij} associada. No HTK, o estado de entrada 1 e o estado de saída N, de uma HMM de N estados,

são não-emissivas (não emitem estados, de modo que não possuem uma distribuição de probabilidades observável associada a elas) (DELLER JR; HANSEN; PROAKIS, 1993; FURUI, 1993; SILVA, 2010).

A Figura 12 ilustra um HMM *left-to-right* com 5 estados. A matriz de transição para esse modelo terá 5 linhas e 5 colunas (YOUNG et al., 2009).

Figura 12 – HMM *Left-to-Right*.



Fonte: Young et al. (2009)

O principal intuito do HTK é representar as distribuições de saída através de densidade de misturas Gaussianas (*Gaussian Mixture Densities*). No caso desse *software*, no entanto, é feita uma generalização, de modo que ele permite que cada vetor de observações em um tempo t seja dividido em S fluxos de dados independentes $\mathbf{o}_{st}$. O cálculo da distribuição $b_j(\mathbf{o}_t)$ é dado pela Equação 3.7 (YOUNG et al., 2009).

$$b_j(\mathbf{o}_t) = \prod_{s=1}^{S} \left[\sum_{m=1}^{M_{js}} c_{jsm} \mathcal{N}\left(\mathbf{o}_{st}; \boldsymbol{\mu}_{jsm}, \boldsymbol{\Sigma}_{jsm}\right) \right]^{\gamma_s} \tag{3.7}$$

Sendo M_{js} o número de componentes da mistura no estado j para o fluxo s, c_{jsm} é o peso do $m-ésimo$ componente e $\mathcal{N}\left(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$, dada pela Equação 3.8, é uma Gaussiana multivariada com vetor de médias $\boldsymbol{\mu}$ e matriz de covariância $\boldsymbol{\Sigma}$.

$$\mathcal{N}\left(\mathbf{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \frac{1}{\sqrt{(2\pi)^n \mid \boldsymbol{\Sigma} \mid}} e^{-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{o}-\boldsymbol{\mu})} \tag{3.8}$$

Em que, n é a dimensionalidade de $\mathbf{o}$. O expoente γ_s é o ganho do respectivo fluxo e seu valor padrão é 1. Outros valores podem ser utilizados para enfatizar fluxos particulares. No entanto, essa variável não é manipulável no HTK.

Para determinar os parâmetros de um HMM, o HTK inicialmente assume uma suposição grosseira do que ela pode ser. Feito isso, parâmetros mais exatos podem ser alcançados empregando-se a fórmula de re-estimação de Baum-Welch (NELSON NETO et al., 2011). Um maior detalhamento dessa função é apresentada em (YOUNG et al., 2009).

Primeiramente, baseia-se no fato de que a inclusão de múltiplos fluxos de dados não altera a questão da re-estimação significantemente, uma vez que cada fluxo pode ser considerado estatisticamente independente. Além disso, os componentes da mistura podem ser considerados uma forma especial de sub-estado, em que a transição de probabilidades são os pesos da mistura. Então, o problema essencial é estimar as médias e as variâncias de um HMM no qual cada estado de saída é uma única componente Gaussiana, dada pela Equação 3.8.

Em seguida, determina-se o estado de máxima verossimilhança, utilizando-se o algoritmo de Viterbi, descrito em Young et al. (2009), e se reatribui o novos vetores de observação aos estados, aplicando-se novamente o processo com valores iniciais mais precisos. Isso, então, é repetido até que a estimativa não mais mude, ou seja, atinja o estado de máxima verossimilhança.

Em sistemas de reconhecimento automático da fala, HMMs são utilizados para representar subunidades de palavras (como os fonemas, ou *fones*). Para o idioma inglês, tipicamente se tem 40 modelos de fonemas. Seu conjunto exato depende do dicionário fonético utilizado. Modelos de palavras podem ser construídos como uma combinação dos modelos de sub-palavras (RESCH, 2011).

Então, essa estimação pode ser inicialmente aplicada para o treino de HMMs monofones com uma única Gaussiana partindo-se de um conjunto de monofones com médias e variâncias idênticas. Elas são, posteriormente, re-estimadas e os modelos de pausa curta são adicionados, como também os modelos de silêncio são expandidos suavemente. Até que, novamente, os monofones são re-estimados (SPAANS, 2004; YOUNG et al., 2009).

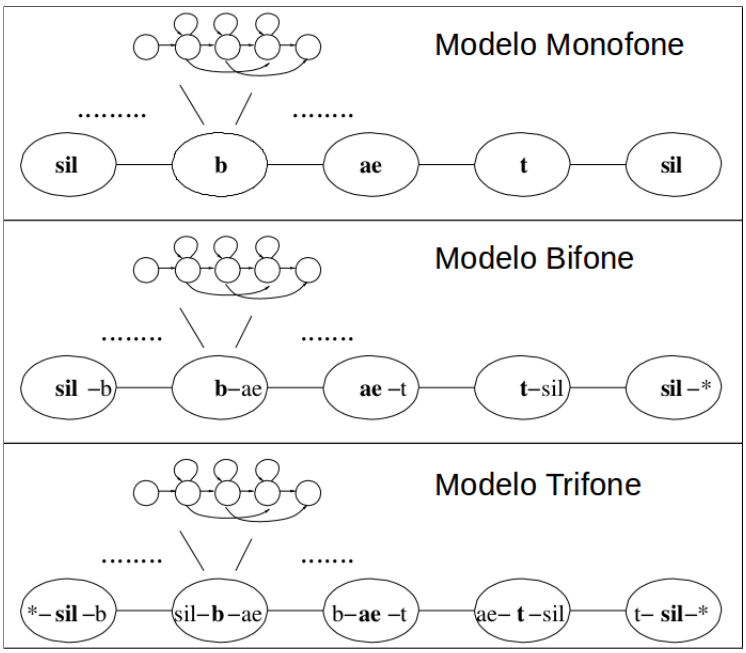
O processamento envolvido para a criação de um plano de monofones iniciais, fixação dos modelos de silêncio e realinhamento dos dados de treino são descritos com detalhes por Young et al. (2009) em um exemplo de utilização do pacote HTK.

Na prática, no entanto, as variações de um mesmo *fone* diferem bastante, dependendo de seus vizinhos (contexto dos fonemas). Portanto, modelos de *fones* dependentes de contexto são mais amplamente utilizados (SILVA; NELSON NETO; KLAUTAU, 2009; NELSON NETO et al., 2011).

Modelos bifone consideram o fonema à esquerda (precedente) ou à direita (sucessor). Já os modelos trifones levam em consideração ambos os *fones* vizinhos, e, para cada um, modelos diferentes são utilizados em diferentes contextos. Na Figura 13, a palavra em inglês *bat* [*b ae t*] é mostrada nas representações monofone, bifone e trifone. A representação *'sil'* indica silêncio no início e no final da elocução, também sendo modelado como um *fone* (RESCH, 2011).

Os modelos subjacentes para os fones, ou suas combinações (nos casos dos bi e trifones), são geralmente HMMs. Devido a uma imensa lacuna de ocorrências suficientes de todas as combinações trifones nos dados de treinamento – um alfabeto fonético de 40 fones resulta em um número de $40^3 = 64000$ trifones possíveis. Técnicas de de agrupamento

Figura 13 – HMMs monofone, bifone e trifone para a palavra *bat*, em inglês.



Fonte: Resch (2011).

(*clustering*), por exemplo, pela utilização de árvores de regressão binária, são frequentemente utilizadas para a obtenção de modelos mais confiáveis com relação a combinações raras de ocorrer (RESCH, 2011).

Dessa forma, a partir do conjunto de monofones estimado, pode-se criar os modelos trifones dependentes de contexto, que levam em consideração os efeitos co-articulatórios presentes nas transições entre palavras. Os vínculos das matrizes de transição dos trifones derivados dos mesmos monofones são, então, estabelecidos, e, através da geração de uma árvore de decisão fonética, são realizados os vínculos entre os estados (*tied-state*) levando em consideração regras de classificação fonética preestabelecidas (SILVA; NELSON NETO; KLAUTAU, 2009).

Assim, a saída do sistema será uma hipótese para a transcrição do sinal da fala.

3.2 Julius

O Julius se configura em um *software* que implementa uma ferramenta de reconhecimento da fala contínua de amplo vocabulário (*Large Vocabulary Continuous Speech Recognition* – LVCSR) distribuído sem limitações de licença. A partir de sua versão 4, agrega funcionalidades que o fizeram adequado tanto para aplicações de ditado, como para aplicações de comando e controle utilizando gramáticas de estado finito (*finite-state grammar*) (JULIUS, 2014).

Essa ferramenta foi desenvolvida tanto para Linux como para Windows, suporta ainda variados outros sistemas operacionais, sendo também adequada para o desenvolvimento de aplicações embarcadas (BATISTA, 2013). Seu código foi escrito em linguagem C, provendo suporte ao reconhecimento diretamente a partir da entrada de áudio padrão do sistema, bem como permitindo outras modalidades de entrada, por exemplo, a partir da leitura de um descritor de arquivo diferente do padrão utilizado pelo sistema operacional, ou a partir de um arquivo de áudio previamente gravado.

O *software* se constitui de um decodificador independente de idioma, podendo funcionar para qualquer linguagem, desde que sejam fornecidos o dicionário fonético, o modelo de linguagem e o modelo acústico para a língua desejada. A confiabilidade do reconhecimento depende diretamente da qualidade dos modelos utilizados (LEE, 2010).

Suporta modelos acústicos no formato *ascii* HTK, dicionário fonético em um formato semelhante ao do HTK, e modelos de palavras trigrama (*3-gram*) no formato padrão ARPA. A versão mais recente do Julius também dá suporte à decodificação em tempo real baseada nas *Deep Neural Networks* (DNNs) (HINTON et al., 2012; JULIUS, 2015).

É possível realizar o *download* do Julius na *home page* do projeto (JULIUS, 2014). As opções existentes para distribuições Linux são o *download* dos arquivos fonte ou do pacote com os arquivos binários.

Dentre as principais características do Julius, destacam-se (JULIUS, 2014):

- *Software* LVCSR de código aberto.
- Baseado em reconhecimento preciso, de alta velocidade e em tempo real.
- Baixo requisito de memória: menos de 32MBytes por área de trabalho.
- Suporta modelo de linguagem N-grama com N arbitrário. Também suporta gramática baseada em regras e lista de palavras para o reconhecimento de palavras isoladas.
- Altamente configurável: vários parâmetros de busca podem ser setados. Além disso, algoritmos de decodificação alternativos podem ser escolhidos.
- Comunidade de desenvolvimento ativa e com vasto material de suporte.

3.2.1 Formatos de áudio suportados

O Julius suporta entradas de forma de onda (analógicas), como também de vetores com as características extraídas do áudio (digitais). Para o caso de dados em forma de onda, pode ser fornecido um arquivo de gravação de áudio ou o fluxo de áudio provindo do dispositivo de captura(LEE; KAWAHARA, 2009).

A quantização das amostras deve ser realizada com 16 bits, não havendo suporte para 8 ou 24 bits. Deve-se utilizar somente 1 canal (mono) tanto para a gravação dos arquivos,

como para o reconhecimento realizado diretamente do fluxo provindo do microfone (LEE, 2010).

A taxa de amostragem do áudio de entrada deve ser dada explicitamente na configuração do *software*. Caso nenhuma opção seja fornecida ao Julius, a taxa escolhida por padrão é de 16 KHz. No arquivo de configuração do Julius, as opções *-smpFreq* ou *-smpPeriod* podem ser utilizadas para especificar uma taxa de amostragem nas unidades Hertz ou 100ns, respectivamente [1] .

A taxa de amostragem correta deve ser fornecida de acordo com o modelo acústico utilizado para o reconhecimento. Ela deve ser a mesma utilizada nas condições de treinamento do modelo. No caso de se utilizarem diferentes modelos com condições acústicas diferentes, a taxa de amostragem desses modelos deve ser a mesma. Isso funciona como requisito para a entrada de áudio. Se for utilizado um tipo de entrada "ao vivo", como o microfone, o dispositivo de entrada será configurado com a taxa de amostragem configurada no Julius.

Caso a taxa de amostragem especificada não seja suportada pelo dispositivo de entrada de áudio do sistema, o Julius apresentará uma mensagem de erro. Quando a entrada for disponibilizada a partir de um arquivo de áudio, a frequência de amostragem do arquivo é examinada e comparada com a taxa configurada no Julius; se os valores forem diferentes, o arquivo é rejeitado – para arquivos no formato *.raw* isso não funciona, uma vez que esses arquivos não possuem cabeçalho com as informações características (LEE, 2010).

3.2.1.1 Entrada por arquivo de áudio

A opção *-input rawfile*, no arquivo de configuração do Julius, informa que a entrada do áudio deve ser realizada a partir da leitura de um arquivo. Pode ser fornecido o nome do arquivo a ser processado na entrada padrão do Julius. Múltiplos arquivos podem ser processados, um por um, listando-se os nomes dos arquivos em um arquivo de texto, sendo esse especificado no parâmetro *-filelist* (LEE, 2010).

Por padrão, o Julius assume que um arquivo se refira a uma sentença pronunciada, com uma parte de silêncio no início e no final do arquivo. No entanto, especificando-se algumas opções de configuração, é possível aplicar detecção da atividade de voz (*voice activity detection*), supressão de silêncio (*silence cutting*) e outras funções normalmente usadas para a entrada do microfone (LEE; KAWAHARA, 2009).

Essa ferramenta de RAF pode realizar a leitura dos seguintes formatos de arquivo:

- Formato WAV (*Waveform Audio Format File*) da Microsoft, extensão WAV (16 bits por amostra, PCM (sem compressão), monaural)
- Formato RAW: sem cabeçalho, *signed-short* 16 bits por amostra, *big endian*, monoaural

[1] VOXFORGE. *Running Julius Live*. Disponível em: <http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/tutorial/run-julius>.

Caso seja utilizado o pacote *libsndfile* com o Julius, podem ser admitidos outros formatos, como AU, NIST, ADPCM, etc. O *libsndfile* será usada com o Julius caso seus arquivos de desenvolvimento (cabeçalhos e bibliotecas) estejam instalados no sistema quando o Julius for compilado (LEE; KAWAHARA, 2009).

Deve-se ter certa atenção quando da utilização do formato RAW. O Julius aceita somente o formato *Big Endian*. Caso seja fornecido à entrada do Julius um arquivo RAW no formato *Little Endian*, isso pode não ser detectado e gerar resultados errados sem alertas de erro. É possível converter o *endianness* de um arquivo, através do utilitário *sox*, no Linux, conforme o comando a seguir (LEE, 2010):

```
1   $ sox -t .raw -s -w -c 1 infile -t .raw -s -w -c 1 -x outfile
```

Onde, *infile* e *outfile* devem ser substituídos pelos nomes dos arquivos de entrada (*little endian*) e saída (*big endian*) do *sox*.

Deve-se observar se o arquivo RAW está formatado também com as demais características adequadas (taxa de amostragem, bits por amostra, etc) ao modelo acústico utilizado, uma vez que o arquivo RAW não possui cabeçalho de informações, impedindo que o Julius possa verificar automaticamente.

3.2.2 Opções de configuração

As configurações do Julius, incluindo modelos, parâmetros e configurações devem ser indicadas através das opções do *software*. Essas opções podem ser especificadas como argumentos da linha de comando, ou podem ser escritas em um arquivo de texto, que deve ser informado com o argumento -C quando da execução no *prompt* de comando. O arquivo de texto contendo as opções do Julius é chamado de "arquivo de configuração *jconf*"(LEE, 2010; BATISTA, 2013).

Quando da especificação do caminho do arquivo de configuração *jconf*, deve-se estar atento à indicação de caminhos relativos informados nos parâmetros do arquivo, uma vez que esses caminhos serão tratados com relação ao próprio *jconf*, e não ao diretório atual.

LEE (2010) apresenta uma lista de todas a opções e suas explicações, agrupadas de acordo com as classes: opções de aplicação, opções globais, opções de declaração de exemplo, opções de modelo de linguagem, modelo acústico, opções de características e opções de busca.

3.3 *Pocketsphinx*

O *Pocketsphinx* se configura em um sistema para o reconhecimento da fala contínua em dispositivos portáteis, desenvolvido pelo *Carnegie Mellon University Language Technology*

Institute (HUGGINS-DAINES et al., 2006 apud GOMÉZ, 2011). Sua implementação teve como base o sistema Sphinx-II (HUANG et al., 1993 apud GOMÉZ, 2011), sendo baseada nos HMMs.

Essa ferramenta de reconhecimento depende da biblioteca *SphinxBase*, que provê funcionalidades comuns a todos os projetos desenvolvidos junto ao CMUSphinx. É compatível com sistemas Linux, Windows, MacOS, iPhone e Android (CMU SPHINX, 2015a). Assim, o pocketsphinx é uma versão moderna do Sphinx-II, especialmente otimizada para sistemas embarcados e portáteis. Ele consome, em média, 20% menos memória e de 5 a 20% menos processamento do que o Sphinx-II (CMU ROBUST GROUP, 2008).

O projeto CMUSphinx implementa um decodificador independente de idioma, podendo fucionar para qualquer linguagem, desde que sejam fornecidos um modelo acústico e um modelo de linguagem adequados. São disponibilizados juntamente com as ferramentas do projeto modelos acústicos de alta qualidade no idioma inglês, tanto para entrada através do microfone, como para a fala através do canal telefônico. Assim, o pocketsphinx é adequado para a maioria das aplicações de comando e controle, como também para as aplicações de amplo vocabulário.

O grupo de pesquisa CMUSphinx alerta para o fato de que, no entanto, algumas aplicações não funcionarão com os modelos acústicos disponíveis, podendo apresentar uma acurácia reduzida. No entanto, podem ser tomadas certas medidas para melhorar a acurácia da ferramenta utilizando esses modelos, conforme apresenta Grimmett (2014).

Em tutorial para a criação de modelos acústicos, o grupo CMU Sphinx (2015b) sugere ocasiões em que se mostra necessário realizar o treinamento de um novo modelo específico, como para o caso da necessidade de um modelo especializado para aplicações com pequeno vocabulário.

Uma vez que, para aplicações em dispositivos portáteis os recursos de memória e processamento são escassos, Huggins-Daines et al. (2006) apresenta alguns *tradeoffs* concernentes a aplicações móveis, embarcadas e portáteis que nortearam as otimizações requeridas no desenvolvimento do pocketsphinx. Dentre elas, destaca:

- A busca por maior eficiência no uso da memória e no acesso aos dados da plataforma de *hardware*.
- A utilização de operações em ponto fixo visando melhorar o desempenho das aplicações.
- Otimização dos algoritmos de reconhecimento.

O CMUSphinx conta com um conjunto de ferramentas para criação de modelos acústicos, constantes no pacote SphinxTrain. Os algoritmos de extração das características dos sinais de áudio utilizados nesse pacote diferenciam-se do HTK quanto ao banco de filtros e transformada utilizados. No entanto, o pocketsphinx apresenta um suporte específico

para prover a capacidade de utilizar os modelos criados a partir do HTK, conforme destaca Shmyrev (2011), desenvolvedor da equipe CMUSphinx, em seu blog.

Shmyrev (2014) aponta ainda para o fato de que resultados aproximadamente similares podem ser alcançados com a utilização das *tookits* HTK e CMUSphinx. Cada uma implementa uma diversidade de funcionalidades específicas para a criação de modelos, de modo que os modelos criados com o HTK possuem uma estrutura mais flexível, enquanto o CMUSphinx disponibiliza um decodificador com estrutura mais compacta.

3.3.1 Formatos de áudio suportados

Em exemplo disponibilizado pelo projeto CMUSphinx para a construção de aplicações utilizando o pocketsphinx, é apresentada a opção de suporte à entrada a partir de arquivos de áudio. Nesse exemplo, é sugerida a utilização de um arquivo no formato *.raw*, chamando-se a atenção para o fato de que a gravação deve possuir as características adequadas ao modelo acústico utilizado. No caso, canal mono, *little-endian*, sinal *signed* PCM 16 bits por amostra sem cabeçalho e taxa de amostragem de 16kHz (CMU SPHINX, 2015a).

Outros exemplos de aplicação do pocketsphinx também apresentam a utilização de arquivos de entrada de áudio no formato *.wav* (CMU SPHINX FAQ, 2016).

Grimmett (2014) apresenta o processo de instalação e configuração do pocketsphinx na plataforma Beaglebone Black (BEAGLEBONE..., 2015). O procedimento apresentado engloba a utilização de um dispositivo de entrada a partir de uma placa de áudio USB (*Universal Serial Bus*), realizando a captura diretamente a partir de um microfone para o reconhecimento de voz com a ferramenta pocketsphinx.

3.4 Discussão

A partir do conhecimento dos componentes que constituem um sistema de reconhecimento da fala típico, forma-se uma visão geral sobre a teoria do reconhecimento automático da fala.

No que tange à criação de modelos acústicos foi apresentada a ferramenta HTK que encontra-se com uma comunidade de desenvolvimento ativa e se constitui em um dos principais *softwares* que implementam o estado da arte em modelagem acústica baseada nos HMMs.

Os princípios basilares quanto ao processamento e codificação da fala, análise espectral dos dados da fala e a criação de modelos da fala foram delineados voltando-se para as técnicas de processamento com maior afinidade ao trabalho desenvolvido nesse projeto, concentrando-se na utilização do HTK.

Baseando-se na experiência da equipe de desenvolvimento do *toolkit*, foram apresentados os métodos de análise e parametrização mais aplicados no desenvolvimento de

sistemas de reconhecimento automático da fala, baseados nos MFCCs em conjunto com parâmetros adicionais contendo a informação da energia do sinal e coeficientes delta, que são recomendados na busca de se maximizar o desempenho do reconhecimento.

Uma fundamentação quanto à criação de HMMs para representação da fala foi abordada trazendo aspectos quanto à estimação das unidades básicas que compõe a fala, bem como do contexto em que essas unidades podem estar inseridos em função de combinações que determinam os efeitos co-articulatórios sobre elas. Foram apresentados os modelos monofone, bifone e trifone para a palavra *bat*, em inglês, exemplificando essas questões.

Quanto aos decodificadores, foram apresentados o Julius e o Pocketsphinx, que possuem características similares no âmbito do suporte a sistemas operacionais embarcados e formatos de áudio de entrada. A caracterização apresentada das ferramentas indica que podem ser alcançados resultados aproximadamente similares com o emprego de ambos decodificadores restringindo-se a parametrização dos vetores característicos de entrada de áudio com o emprego de modelos criados a partir do HTK.

O capítulo seguinte apresenta os materiais e métodos empregados no desenvolvimento deste trabalho, que envolve uma aplicação específica com pequeno vocabulário, direcionada ao controle de dispositivos em um ambiente doméstico.

4 Materiais e métodos

A metodologia adotada considera o emprego de duas abordagens distintas para o desenvolvimento da interface de comandos por voz proposta:

1. Utilização do módulo *Voice Recognition Module* associado à plataforma Beaglebone Black.

2. Implementação do RAF através do mecanismo de reconhecimento Julius integrado ao servidor de comunicação Asterisk, ambos embarcados na plataforma Beaglebone Black.

Com o intuito de realizar um acompanhamento mais preciso das atividades desenvolvidas no projeto, foi organizado um processo de desenvolvimento reduzido, o qual consistiu de três fases, conforme apresentado na Figura 14. Essa metodologia de desenvolvimento é inspirada em processos como o RUP (*Rational Unified Process*)[1] , o *ipProcess*[2] , o PRODIP (Processo de Desenvolvimento de Produtos Industriais)[3] , ou, até mesmo, em processos de desenvolvimento rápido (RAD)[4] , como o *Scrum*[5] . Ela deve evoluir na medida em que evoluem também as interfaces implementadas.

Na fase de concepção, é definido um cenário de aplicação, como também construído um diagrama de blocos representando os componentes da interface e a integração entre eles. Conforme o cenário de aplicação estabelecido é gerado um vocabulário e gramática de comandos.

Em seguida, os artefatos produzidos na fase de concepção da interface são utilizados como entrada para a fase de implementação, guiando o procedimento de treinamento dos comandos de voz. Isso exige ainda a preparação do ambiente de desenvolvimento com as ferramentas adequadas, de acordo com cada respectiva abordagem. Então, já com a plataforma treinada para receber os comandos de voz específicos, é realizada a programação da aplicação específica.

A partir daí, com a aplicação implementada na plataforma de desenvolvimento, realiza-se uma fase de testes com o intuito de se avaliar o desempenho da interface quanto a sua eficiência na execução dos comandos de voz programados.

---

1 RATIONAL Unified Process. 2003. Disponível em: <http://www.wthreex.com/rup/portugues/index.htm>. Acesso em: 15 jan. 2016.

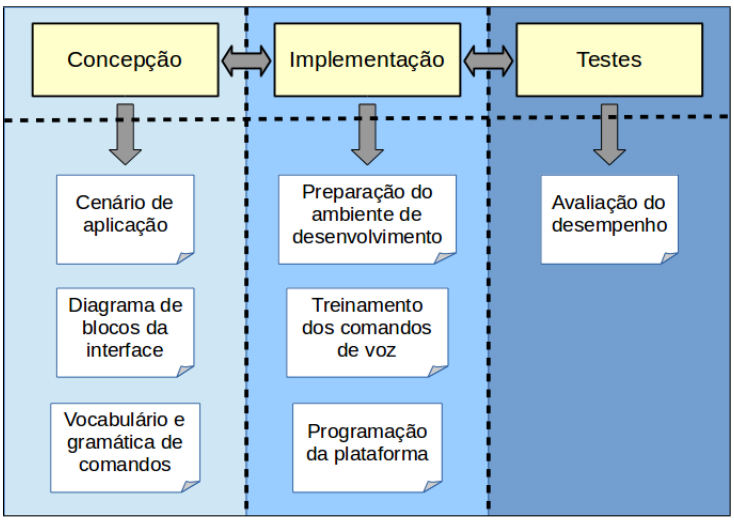
2 SANTOS, F. et al. *ipProcess*: A Usage of an IP-core Development Process to Achieve Time-to-Market and Quality Assurance in a Multi Project Environment. Disponível em: <http://www.design-reuse.com/articles/21746/ip-core-development-process.html>. Acesso em: 16 jan. 2016.

3 UNIVERSIDADE FEDERAL DE SANTA CATARINA. *Modelo PRODIP*. 2016. Disponível em: <http://emc5302.ogliari.prof.ufsc.br/artigo/modelo-prodip>. Acesso em: 16 jan. 2016.

4 WIKIPÉDIA. *Rapid Application Development*. Disponível em: <https://pt.wikipedia.org/wiki/Rapid_Application_Development>. Acesso em: 16 jan. 2016.

5 DESENVOLVIMENTO ÁGIL. *Scrum*. 2014. Disponível em: <http://www.desenvolvimentoagil.com.br/scrum/>. Acesso em: 16 jan. 2016.

Figura 14 – Fases de desenvolvimento das interfaces.



Fonte: Autoria própria.

É importante notar que os produtos gerados nas fases de implementação e testes realimentam a fase de concepção da interface, como também a fase de testes realimenta a fase de implementação. Assim, é possível realizar adaptações e melhoramentos ao longo da execução do projeto.

A plataforma com arquitetura de processamento ARM escolhida para o desenvolvimento deste projeto foi a placa Beaglebone Black (Rev C). Essa escolha se deu por estar mais acessível ao uso, uma vez que foi disponibilizada pelo Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB), além de ser uma plataforma de *hardware* aberto, baixo custo, e que conta com uma comunidade de desenvolvimento ativa[6] .

Apesar de se constituir em uma atividade relacionada à preparação do ambiente de desenvolvimento, que ocorre durante a fase de implementação, antes de iniciar o detalhamento das atividades desenvolvidas em cada abordagem de projeto é interessante apresentar uma descrição das características da placa de desenvolvimento Beaglebone Black. Esta plataforma é um ponto comum para ambas as abordagens empregadas neste trabalho.

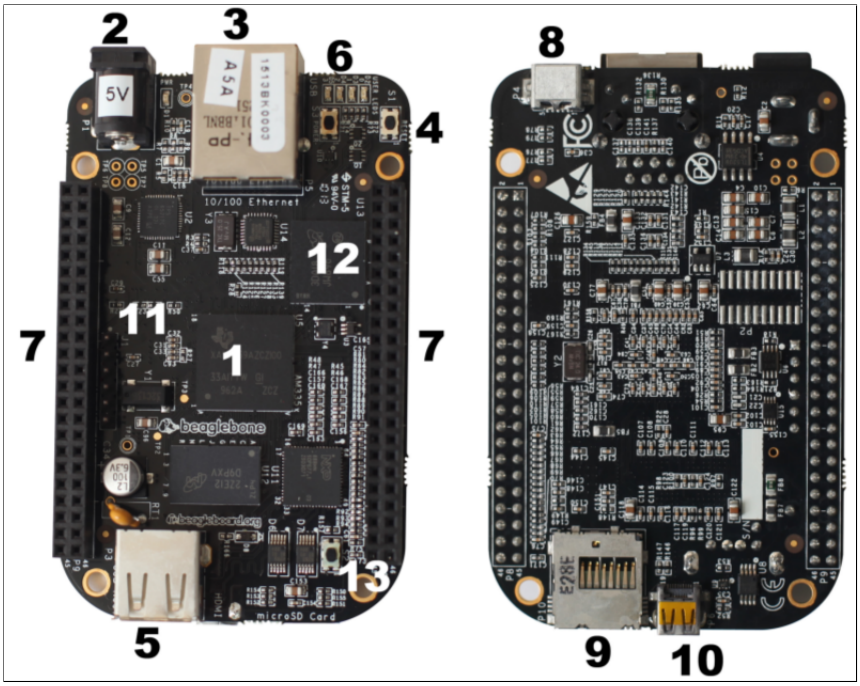
Assim, na seção 4.1, é feita uma apresentação sucinta das características da placa de desenvolvimetno Beaglebone Black, descrevendo-se a configuração realizada. Na seção 4.2 são descritas as atividades realizadas na implementação da primeira abordagem supracitada, enquanto que a seção 4.3 descreve as atividades referentes à implementação da segunda abordagem.

[6] BEAGLEBONE Black. 2015. Disponível em: <http://beagleboard.org/black>. Acesso em: 12 jun. 2015.

4.1 A placa Beaglebone Black e sua configuração

Os principais componentes da placa Beaglebone Black, apresentada na Figura 15, são descritos por Richardson (2014) conforme segue:

Figura 15 – Principais componentes da placa Beaglebone Black.



Fonte: Richardson (2014)

1. Processador AM335x ARM Cortex-A8 de 1GHz, equipado com 512MB de memória RAM DDR3.
2. Conector da fonte, que opera recebendo 5V e 500mA, especificado para receber adaptadores com pino *jack* de 2.1mm
3. Porta Ethernet no padrão RJ45 possibilitando usar conexões de Internet compartilhadas.
4. Botão de reinicialização.
5. Porta USB, permitindo conectar diferentes dispositivos, como: teclado, *mouse*, bem como adaptadores de rede sem fio (WiFi 802.11) ou ainda todos esses dispositivos conectados a um *hub* USB.
6. LEDs *OnBoard*. Um próximo ao conector da fonte para indicar quando a placa estiver ligada. E outros quatro LEDs próximos ao botão de *reset*, os quais podem ser programados via *software*.

7. Conectores de expansão, denominados P8 e P9, que permitem a integração da Beagleboard com os mais diversos projetos, podendo ser configurados de acordo com as funcionalidades desejadas.

8. Porta mini USB, que permite a conexão da Beagleboard com o computador, funcionando também como fonte de alimentação para a placa, bem como provendo comunicação serial entre PC e Beagleboard. Provê ainda a capacidade de emular conexão de rede local para acesso via SSH (*Secure Shell*)[7] .

9. Leitor de cartão SD (*Secure Digital Card*).

10. Porta micro HDMI (*High-Definition Multimedia Interface*) para conexão da Beagleboard a um monitor ou televisão.

11. Conector Serial para acesso via terminal.

12. Memória *flash* interna de 4GB onde pode ser instalado o sistema operacional.

13. Chave de *boot*, que ao ser pressionada na inicialização do sistema permite que a placa carregue o sistema operacional do cartão SD ao invés da memória *flash interna*.

A Beagleboard apresenta ainda um acelerador gráfico 3D, dentre outras funcionalidades e recursos que ampliam seu potencial para o desenvolvimento de aplicações de sistemas embarcados. Além disso, é compatível com diversos sistemas operacionais *open-source* – como Debian, Ubuntu e Android, dentre outros.

Por se tratar de uma plataforma de *hardware* aberto, a Beaglebone Black apresenta alta flexibilidade quanto às ferramentas de desenvolvimento, podendo adicionar facilmente novos módulos de *hardware*, abrindo oportunidades para o desenvolvimento das mais diversas aplicações, com alto potencial para sistemas ubíquos e pervasivos, ou ainda, alinhados ao conceito de Internet das Coisas (*Internet of Things* – IoT).

Para o propósito de desenvolvimento deste projeto, a Beagleboard foi configurada com o sistema operacional Linux embarcado. No caso da primeira abordagem desenvolvida, foi instalada a distribuição Ubuntu 13.10 Linux. Já durante o desenvolvimento da segunda abordagem, no entanto, foram encontradas dificuldades relacionadas à instabilidade dos repositórios de pacotes desta distribuição, tendo se optado pela mudança da distribuição Linux instalada na segunda abordagem, que passou a ser a Debian 7.8 armv7l, kernel 3.8.13-bone70.

O motivo dessa escolha reside na reconhecida confiabilidade e qualidade de código do sistema operacional Linux, de sua estrutura modular e alta escalabilidade, apresentando amplo suporte a *hardware*, bem como por ser um sistema *open-source* (YAGHMOUR et al.,

[7] SSH to Beaglebone Black over USB. Disponível em: <https://learn.adafruit.com/ssh-to-beaglebone-black-over-usb/overview>. Acesso em: 03 jul. 2015.

2014). Além disso, a comunidade de suporte e desenvolvimento da distribuição Debian, provê repositórios de *software* mais estáveis, estando ativa há mais de 15 anos[8] .

Para o desenvolvimento das aplicações de reconhecimento de comandos por voz, foi instalado e configurado na Beagleboard o suporte à linguagem de programação Python, em sua versão 2.7.3. A escolha justifica-se uma vez que é uma linguagem de alto nível bastante intuitiva, e conta com ampla documentação e comunidade de suporte. Também por ser uma linguagem de código aberto, desenvolvida sob uma licença aprovada para iniciativas de sistemas abertos (*Open Source Initiative* – OSI[9]), conhecida como *Python Sofware Foundation License*.

A alta flexibilidade do Python, que oferece suporte para as mais variadas aplicações – desde *Web* e desenvolvimento para a Internet, ambientes *desktop*, programação para redes, até computação científica e numérica, dentre outras – conta ainda com uma grande variedade de bibliotecas disponíveis livremente[10] .

Configurada com as capacidades aqui apresentadas, a Beagleboard está pronta para ser utilizada na implementação das abordagens propostas ao desenvolvimento deste projeto.

Apesar de terem sido utilizadas diferentes distribuições Linux para o desenvolvimento de cada abordagem, os procedimentos de instalação e configuração realizados em ambos os casos são similares. No Apêndice A é apresentado um detalhamento desses procedimentos para a distribuição Debian 7.8 Linux.

4.2 Abordagem 1: Utilização do módulo *Voice Recognition Module* V2 associado à Beaglebone Black

Para o caso da primeira abordagem, foi realizado um experimento piloto, que consistiu na implementação de uma aplicação de reconhecimento da fala utilizando um módulo de *hardware*. Essa etapa elucidou alguns fundamentos práticos – como a familiarização com a plataforma de desenvolvimento Beagleboard e com a linguagem de programação Python – e metodológicos – como no caso da concepção da interface e da avaliação do seu desempenho – que contribuíram significativamente para alcançar o objetivo principal do projeto.

A subseção 4.2.1 traz as atividades desenvolvidas durante a fase de concepção dessa primeira abordagem. Na subseção 4.2.2 serão detalhados aspectos relacionados a sua implementação. A subseção 4.2.2.1 apresenta o módulo *Voice Recognition Module* V2, destacando seu emprego no desenvolvimento da interface. Já na subseção 4.2.3 é descrita a metodologia de testes para avaliação do desempenho da plataforma de comandos por voz no cenário de aplicação definido.

[8] BEAGLEBOARD. *Debian*. Disponível em: <http://beagleboard.org/project/debian/>. Acesso em: 21 jan. 2016.

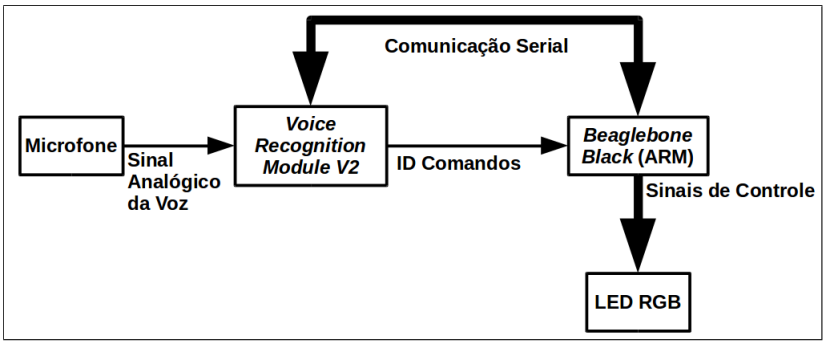
[9] OPEN SOURCE INITIATIVE. Disponível em: <http://opensource.org/>. Acesso em: 04 jul. 2015.

[10] ADAFRUIT. *Setting up IO Python Library on BeagleBone Black*. 2013. Disponível em: <https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/overview>. Acesso em: 21 abr. 2014.

4.2.1 Concepção da interface baseada em módulo de *hardware*

Com o intuito de ilustrar o reconhecimento dos comandos de voz executados pela plataforma, o cenário de aplicação consistiu de um circuito que realiza o controle das luzes de um LED RGB. A Figura 16 ilustra o diagrama de blocos dessa primeira interface.

Figura 16 – Diagrama de blocos da interface de comandos por voz baseada em módulo de reconhecimento de voz.



Fonte: Adaptado de Pimentel et al. (2014)

O módulo de *hardware* empregado é conhecido comercialmente como *Voice Recogniton Module* V2(SHEN, 2013). A opção pela utilização desse módulo foi motivada pela questão de estar mais acessível ao uso, uma vez que constava dentre os recursos previamente disponíveis para o projeto.

A partir do cenário de aplicação especificado foi, então, definido o vocabulário de comandos, associando-se cada comando a um pino de saída do módulo de reconhecimento de voz, bem como correlacionando tais comandos com a respectiva resposta esperada, conforme ilustrado na Tabela 5.

Tabela 5 – Comandos para a validação da interface baseada em módulo de *hardware*.

Comando de voz	Pino	Resposta esperada do Sistema
/rubro/	O1	Acendimento da luz vermelha
/verde/	O2	Acendimento da luz verde
/azul/	O3	Acendimento da luz azul
/branco/	O4	Acendimento das luzes vermelha, verde e azul
/apaga/	O5	Desliga as luzes do LED

Fonte: Pimentel et al. (2014).

4.2.2 Implementação da interface baseada em módulo de *hardware*

Na seção 4.1, já descrevemos alguns procedimentos relacionados à parte da configuração e preparação do ambiente de desenvolvimento, como a instalação do sistema operacional,

bem como das bibliotecas que proveem o suporte à linguagem de programação Python. Aquelas configurações são similares quanto a implementação de ambas abordagens executadas neste trabalho.

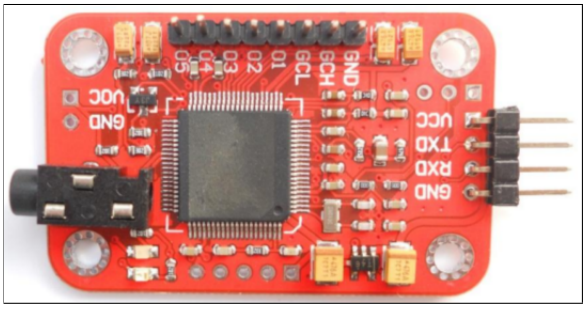
Para o caso dessa primeira abordagem, que consiste na construção da interface de comandos por voz baseada em módulo de *hardware* foram, no entanto, utilizadas bibliotecas específicas para o controle das portas GPIO (*General Purpose Input/Output*) da Beagleboard utilizando a linguagem Python. A instalação dessa biblioteca em ambiente operacional Linux embarcado é descrita com detalhes em Cooper (2015).

Adicionalmente, as características e princípios de funcionamento do módulo de reconhecimento de voz utilizado determinaram diversas outras necessidades concernentes ao ambiente de desenvolvimento, conforme descrito na subseção 4.2.2.1.

4.2.2.1 O módulo *Voice Recognition Module* V2 e sua integração com a Beaglebone Black

O *Voice Recognition Module V2*, ilustrado na Figura 17, se constitui de um módulo de reconhecimento de comandos por voz dependente de locutor, de baixo custo, e que pode ser utilizado para controlar os sistemas em um carro ou outros dispositivos eletroeletrônicos. Possui uma interface analógica monocanal integrada, com conector de 3.5mm para microfone (SHEN, 2013).

Figura 17 – *Voice Recognition Module V2*.



Fonte: Shen (2013)

A aplicação implementada nesta abordagem utiliza como entrada de áudio analógico um microfone conectado à referida interface analógica do módulo de reconhecimento de voz, tendo sido utilizado o microfone padrão disponibilizado juntamente com o referido módulo. Dessa forma, uma vez que tenha recebido o comando de voz, o módulo V2 processa o áudio de entrada e, então, é capaz de gerar um sinal digital que indica o resultado do reconhecimento da elocução pronunciada pelo usuário.

Esse módulo é capaz de armazenar até 15 comandos de voz, os quais são organizadas em 3 grupos. Cada grupo de comandos só pode ser habilitado individualmente para o reconhecimento, de modo que somente os 5 comandos do grupo atual ficam disponíveis

para serem reconhecidos. Para isso, é necessário realizar-se previamente o treinamento dos comandos de voz no referido módulo.

Os comandos de voz podem ser pronunciados no idioma português, atentando-se para a limitação na quantidade de memória interna do módulo, que somente permite elocuções de até 1300ms de duração. Assim, o sistema suporta apenas comandos de palavras isoladas.

A realização do treinamento dos comandos de voz, bem como a ativação e desativação dos modos de funcionamento do módulo, são controladas através do envio de comandos via interface serial (pinos *Transmit Data* – TXD e *Receive Data* – RXD). Além disso, através dessa mesma interface de comunicação, é selecionado o grupo de comandos de voz que deverá estar habilitado para reconhecimento em um dado momento.

A interface serial é do tipo TTL (*Transistor-Transistor Logic*) em 5V, sendo os dados enviados serialmente organizados em 8 bits, sem paridade, contento 1 bit de parada (*stop-bit*), aplicando-se por padrão a taxa de transmissão de 9600bps (bauds por segundo), podendo ser modificada via comando serial para 2400, 4800, 19200 ou 38400bps. Os comandos são transmitidos no formato hexadecimal na forma "*Head* + *Key*", contendo um cabeçalho (*Head*) seguido do código do comando (*Key*).

O resultado do reconhecimento (ID Comandos, conforme apresentado anteriormente na Figura 16) é indicado no módulo em um pino de saída correspondente (pinos de O1 a O5), de modo que a sequência dos comando treinados no grupo atualmente habilitado para reconhecimento está relacionada à sequência dos pinos de saída do módulo. Por exemplo: caso a primeira instrução gravada no grupo habilitado seja reconhecida, será gerado um sinal digital na saída O1, e, assim, sucessivamente.

Esses pinos de saída do módulo podem ser ajustados em variados modos, dentre eles: modo pulso, modo de inversão, modo baixo e modo alto. Maiores detalhes sobre códigos de comando serial e modos de operação podem ser encontrados em Shen (2013).

Cada um dos pinos de saída foram então conectados a pinos GPIO da Beaglebone Black, conforme a Tabela 6.

Tabela 6 – Mapa de pinos *Voice Recognition Module* V2 x Beagleboard.

Módulo V2		**Beaglebone Black**	
Pino	**Função**	**Pino**	**Função**
O1	Saída digital	P9_11	Entrada digital
O2	Saída digital	P9_12	Entrada digital
O3	Saída digital	P9_13	Entrada digital
O4	Saída digital	P9_15	Entrada digital
O5	Saída digital	P9_16	Entrada digital
VCC	Alimentação	P9_6	5V
RXD	Receptor serial	P9_21	TXD
TXD	Transmissor serial	P9_22	RXD
GND	*Ground*	P8_2	*Ground*

Fonte: Autoria própria.

Por questões de diferença entre o nível de tensão gerado nas saídas do módulo de reconhecimento de voz (5V) e o nível suportado pelos pinos de entrada da BeagleBone (3.3V), foram utilizados divisores de tensão, evitando-se a queima das portas na plataforma. A Equação 4.1 foi empregada para o cálculo de dimensionamento dos divisores de tensão, tendo sido utilizados resistores comerciais de 330Ω e $1k\Omega$.

$$R_2 = \frac{-V_2 R_1}{V_2 - V_1} \tag{4.1}$$

Estando a comunicação serial entre a BBB e o *Voice Recognition Module* V2 configurada e funcional, pôde-se dar sequência ao treinamento dos comandos de voz no referido módulo.

4.2.2.2 Treinamento dos comandos de voz no módulo de *hardware* e programação da plataforma

O procedimento de treinamento dos comandos de voz no módulo V2 consiste inicialmente em configurá-lo para iniciar a gravação das amostras de treinamento. Para isso, utilizou-se diretamente o terminal de comando do ambiente de desenvolvimento em Python já configurado com as bibliotecas necessárias.

Em seguida, o *Voice Recognition Module* V2 inicia uma sequência bem definida de acendimento de seus LEDs indicadores, conforme apresentado em (SHEN, 2013), possibilitando ao usuário determinar o momento exato em que se deve pronunciar os comandos de voz que se deseja treinar, sendo realizada, em sequência, a gravação de duas amostras para cada um dos cinco comandos num determinado grupo. A indicação do grupo a ser gravado é determinada pelo código de comando serial enviado.

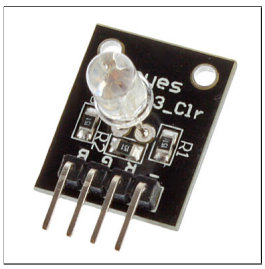
Com o intuito de se verificar o funcionamento dos comandos treinados, foi realizado um pré-teste diretamente através do interpretador do Python.

Após isso, foi desenvolvido o código apresentado no Anexo A, que é capaz de receber o sinal digital da saída do módulo de reconhecimento de voz, identificando o comando falado pelo usuário, e utilizar essa informação para tomar a decisão de alternância no estado dos pinos de saída da Beagleboard que produzirão o acendimento das luzes do LED RGB de acordo com o comando de voz recebido.

O módulo LED RGB utilizado no cenário de aplicação definido, mostrado na Figura 18, apresenta quatro pinos: um para acionar a luz vermelha (R), outro para a luz verde (G), um terceiro para a luz azul (B), e um quarto pino que funciona como GND (*ground*). Esses pinos estão conectados a pinos de expansão da Beagleboard configurados como saída digital, conforme apresenta a Tabela 7. Então, uma vez que a Beagleboard tenha recebido o resultado do reconhecimento, a saída equivalente irá desligar ou acionar as luzes do LED RGB.

Um vídeo demonstrativo do funcionamento da interface de comandos por voz implementada foi produzido e está disponível para visualização na internet, conforme referenciado na Figura 19.

Figura 18 – Módulo LED RGB.



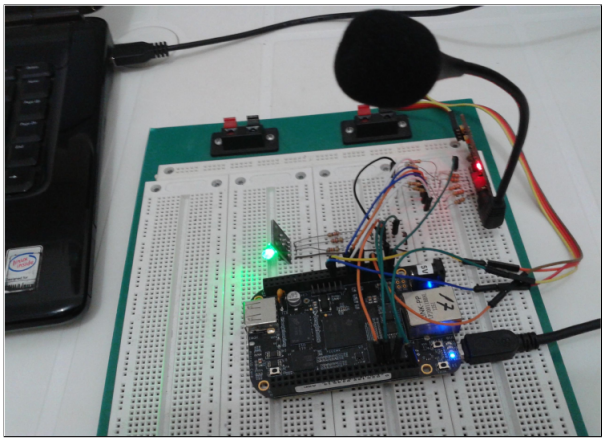
Fonte: Shenzhen Keyes Robot (2014)

Tabela 7 – Mapa de pinos Beagleboard x LED RGB.

Beaglebone Black		**LED RGB**	
Pino	**Função**	**Pino**	**Função**
P8_7	Saída digital	R	Luz vermelha
P8_9	Saída digital	G	Luz verde
P8_11	Saída digital	B	Luz azul
P9_1	*Ground*	–	*Ground*

Fonte: Autoria própria.

Figura 19 – *Frame* do vídeo demonstrativo do funcionamento da interface de comandos por voz baseada em módulo de *hardware*.



Fonte: Pimentel (2014)

4.2.3 Testes realizados com a interface baseada em módulo de *hardware*

Foram convidados 10 locutores para realizar o treinamento descrito na subseção 4.2.2.2, bem como os testes de avaliação do desempenho da interface. Esses procedimentos foram realizados para cada participante em sessões individuais, sendo cinco locutores do sexo masculino e cinco do sexo feminino.

Cada um dos locutores realizou o treinamento do módulo de reconhecimento de voz

com os comandos listados anteriormente na Tabela 5.

Conforme o manual de utilização do módulo V2, sua acurácia quanto ao reconhecimento de comandos em um ambiente ideal alcança a taxa de 99%. No entanto, para o propósito deste trabalho, é necessária sua verificação em ambiente ruidoso, onde a interface será naturalmente utilizada.

Para tanto, o referido treinamento, como também os testes, dos comandos foram realizados em ambiente com nível de ruído na faixa de 45 a 65dB, que equivale a um ambiente de escritório com circulação de pessoas abrindo e fechando portas, conversações, toque de telefone e ruído de aparelho condicionador de ar acontecendo aleatoriamente durante a utilização da interface.

O nível de ruído ambiente foi aferido utilizando-se o aplicativo decibelímetro[11] instalado em *smartphone* rodando o sistema operacional Android[12] . A calibração desse aplicativo é feita com base em níveis de referência apresentados no *software*, os quais são definidos em comparação com as medições apresentadas por outros aparelhos específicos para este fim.

Logo após a respectiva seção de treinamento, a interface de comandos por voz foi testada por cada locutor convidado, que forneceu 20 repetições para cada um dos comandos de voz através do microfone, observando-se o comportamento do sistema.

Foram, então registradas as ocorrências em que a aplicação respondeu corretamente ao comando, em que não apresentou resposta, e também os casos em que ocorreram falsos positivos, em que a pronúncia de um comando apresentou uma saída inesperada (por exemplo, o acendimento da luz errada no LED RGB). Esses dados foram registrados em planilhas, apresentadas no Anexo B.

Ao final deste capítulo, é realizada uma discussão acerca da metodologia aqui descrita. Os resultados dos testes de avaliação da interface desenvolvida na primeira abordagem são apresentados no Capítulo 5

4.3 Abordagem 2: Implementação do RAF com Julius integrado ao servidor Asterisk, embarcados na Beaglebone Black

Com relação à segunda abordagem de desenvolvimento foram utilizadas ferramentas de *software* para o reconhecimento automático da fala, integradas a um servidor de comunicação, embarcados na Beagleboard, para realizar a implementação da interface comandos por voz proposta no objetivo geral desse projeto.

A motivação para a escolha dessa abordagem reside no fato de que ela está inserida no paradigma das atuais tendências de convergência tecnológica, observadas na integração entre

[11] GOOGLE PLAY STORE. *Decibelímetro*: Sound Meter. Disponível em: <https://play.google.com/store/apps/details?id=kr.sira.sound&hl=pt_BR>. Acesso em: 25 jan. 2016.

[12] GOOGLE. *Android*. Disponível em: <https://www.android.com/>. Acesso em: 18 jun. 2015.

diferentes dispositivos e plataformas de comunicação para a implementação de aplicações de comando e controle nos mais diversos cenários, conforme discutido no Capítulo 2.

A sinergia envolvendo os componentes constituintes dessa segunda interface se alinha com os requisitos das modernas arquiteturas de computação ubíqua e pervasiva (SAHA; MUKHERJEE, 2003). Isso é proporcionado pelo servidor de comunicação empregado, que oferece suporte à integração entre diversos protocolos de comunicação, favorecendo a implantação da interface em um contexto de redes convergentes, de modo a acompanhar a presente evolução impulsionada pelo surgimento de novos conceitos como a Internet das Coisas (TAN; WANG, 2010; COETZEE; EKSTEEN, 2011).

A subseção subseção 4.3.1 descreve as atividades desenvolvidas durante a fase de concepção dessa segunda abordagem. Na subseção 4.3.2 serão detalhados aspectos relacionados aos procedimentos executados para sua implementação. Já na subseção 4.3.3 é descrita a metodologia de testes de avaliação do desempenho da plataforma de reconhecimento de comandos por voz para o respectivo cenário de aplicação definido.

4.3.1 Concepção da interface baseada em ferramentas de RAF integradas a servidor de comunicação

O propósito de prover uma interface de comandos por voz para tecnologias assistivas que auxiliem pessoas com mobilidade reduzida no controle do ambiente doméstico direcionou a definição do cenário de aplicação para esta segunda abordagem.

Assim, o cenário se constitui de um protótipo que ilustra a geração de sinais de controle para a iluminação, televisão e acesso (portas), em um ambiente doméstico hipotético constituído de sala, cozinha, quarto, banheiro e área externa, através do acendimento de LEDs, conforme ilustra a Figura 20.

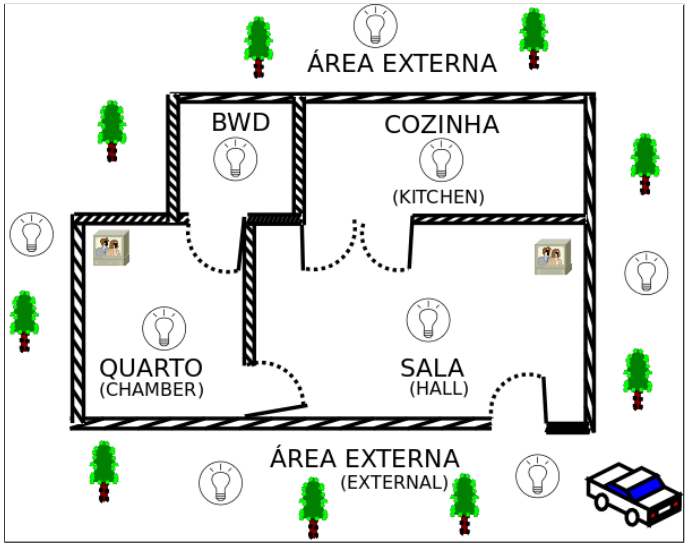
Com o intuito de representar a inserção da interface de comandos por voz no cenário de aplicação definido, bem como a interação entre seu usuário e seus componentes de *hardware* e *software*, foi elaborado um diagrama apresentando sua arquitetura, conforme ilustra a Figura 21.

Como forma de possibilitar a entrada de voz inicialmente a partir de um *smartphone*, e visando o desenvolvimento rápido, optou-se, então, por uma configuração envolvendo a instalação de um *softphone* em *smartphone* rodando o sistema operacional Android[13] . O *software* Zoiper[14] em sua versão 1.30 para Android foi instalado.

Essa escolha se justifica por ser o Zoiper um *software* gratuito, que implementa comunicação VoIP através do protocolo SIP (*Session Initiation Protocol*), compatível com a maioria dos provedores de serviço VoIP e PBX's. Além disso, é flexível, podendo ser executado em diversos sistemas operacionais, como: MacOS, Linux, Windows, iPhone, Android, ou até mesmo ser acessado a partir de um *web browser*.

[13] GOOGLE. *Android*. Disponível em: <https://www.android.com/>. Acesso em: 18 jun. 2015.
[14] ZOIPER. Disponível em: <http://www.zoiper.com/en>. Acesso em: 18 jun. 2015.

Figura 20 – Cenário de aplicação da plataforma baseada em ferramentas de RAF integradas a servidor de comunicação.



Fonte: Autoria própria.

Por questões de mobilidade e portabilidade, optou-se por uma conexão sem fio WiFi 802.11[15] para a disponibilização de acesso à plataforma. Um *access point* utilizando o adaptador Interbras WBN900[16] conectado à porta USB da placa Beaglebone foi configurado. Esse adaptador foi escolhido por acessibilidade, uma vez que estava dentre os recursos disponíveis, minimizando os custos do projeto.

O servidor de comunicação utilizado, conhecido como Asterisk, foi escolhido por ser um *software* livre e de código aberto, mundialmente reconhecido e recomendado, sendo atualmente empregado em mais de um milhão de servidores ao redor do mundo, e que conta uma grande comunidade de suporte composta por cerca de oitenta e seis mil membros registrados, com desenvolvedores em mais de 170 países [17] . Com o intuito de minimizar o tempo de desenvolvimento do projeto, optou-se pela instalação da versão disponibilizada nos repositórios de pacotes do Debian, que consiste no Asterisk 1.8.13.1.

Objetivando-se prover o recurso de reconhecimento automático da fala à plataforma, a ferramenta Julius, em sua versão 4.3.1, foi integrada ao servidor Asterisk. O funcionamento dessa ferramenta de RAF, apresentado no Capítulo 3, exige que sejam fornecidos em sua configuração os modelos acústicos (AM) e de linguagem (LM), bem como o dicionário fonético (DIC) específicos para o idioma que se deseja realizar o reconhecimento.

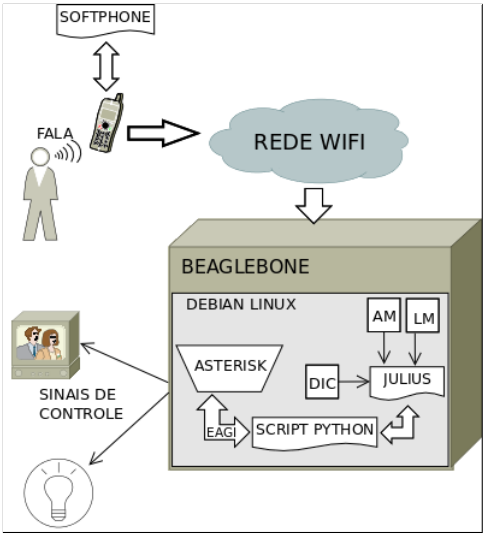
Conforme a discussão descrita no Capítulo 3, é possível se alcançar desempenhos

15 IEEE 802.11^{TM} Wireless Local Area Networks. Disponível em: <http://www.ieee802.org/11/>. Acesso em: 18 jun. 2015.

16 INTERBRAS. *Adaptador USB Wireless N 150 Mbps*. Disponível em: <http://www.intelbras.com.br/residencial/wireless/adaptadores-usb/wbn-900>. Acesso em: 18 jun. 2015.

17 DIGIUM. *Asterisk*. Disponível em: <http://www.asterisk.org/>. Acesso em: 27 jan. 2016.

Figura 21 – Arquitetura da aplicação baseada em ferramentas de RAF integradas a servidor de comunicação.



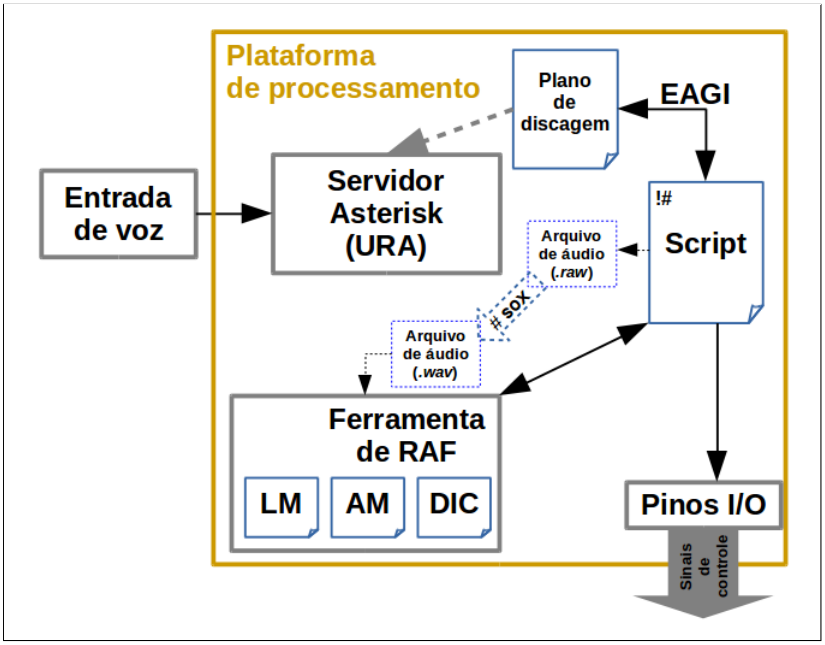
Fonte: Autoria própria.

similares quanto ao reconhecimento automático da fala com o emprego das duas ferramentas estudadas. A preferência pela utilização do Julius se fundamenta por, além de ser uma ferramenta de código aberto e adequada para aplicações embarcadas, conforme comenta Batista (2013), o *engine* utiliza um arquivo de configuração mais intuitivo, que permite melhor manipulação de seus parâmetros, como: entrada de áudio, definição dos modelos a serem utilizados, formato do resultado do reconhecimento, dentre outros. Com isso, criou-se maior afinidade pela utilização desse *software*.

No caso do *pocketsphinx*, as configurações devem ser manipuladas diretamente a partir do código, sendo fornecida uma aplicação modelo escrita na linguagem C (*pocketsphinx_continuous*), o que torna essa ferramenta menos intuitiva (GRIMMETT, 2014; CMU SPHINX, 2015a). Além disso, a documentação do *pocketsphinx* encontra-se dispersa na página de desenvolvimento do projeto, o que acaba requerendo maior tempo de busca sobre seu funcionamento, enquanto que a documentação do Julius se condensa em um livro disponibilizado livremente (LEE, 2010).

A arquitetura definida, ilustrada anteriormente na Figura 21, apresenta a entrada dos comandos de voz, os quais são enviados à plataforma por meio de uma conexão VoIP estabelecida entre o *softphone* instalado no *smartphone* e o servidor Asterisk. Essa conexão permite que o *softphone* devidamente configurado registre-se no servidor de comunicação, possibilitando ao usuário realizar uma chamada ao ramal do servidor onde está configurada uma URA (Unidade de Resposta Audível) integrada à ferramenta de reconhecimento automático da fala. O diagrama de blocos mostrado na Figura 22 detalha os componentes de *software* internos à interface, bem como a integração entre eles.

Figura 22 – Diagrama de blocos da interface baseada em ferramentas de RAF integradas a servidor de comunicação.



Fonte: Autoria própria.

A busca por soluções de RAF em URAs com o servidor Asterisk levou ao conhecimento de três diferentes métodos para sua implementação. O primeiro, baseado na utilização de uma API de reconhecimento de voz genérica[18] interna ao próprio Asterisk, no entanto, não lida com a comunicação entre o servidor e a ferramenta de reconhecimento automático da fala.

Dessa forma, a utilização da API supracitada requer a disponibilidade de um módulo que proveja essa comunicação. Entretanto, para esse primeiro método, foram encontrados somente módulos que proveem a comunicação com *engines* proprietários[19] . Além disso, o desenvolvimento de novos módulos demandaria grande quantidade de tempo e esforço, recursos escassos no caso desse projeto.

A segunda solução encontrada para o RAF integrado ao Asterisk, baseia-se na utilização do módulo *open-source* UniMRCP[20] . Esse módulo utiliza o protocolo MRCP (*Media Resource Control Protocol*) para prover a comunicação necessária entre o Asterisk e a ferramenta de reconhecimento automático da fala.

No entanto, ainda não existem outros módulos disponíveis livremente para a integra-

[18] DAVENPORT, M. *The Asterisk Speech Recognition API*. 2012. Disponível em: <https://wiki.asterisk.org/wiki/display/AST/Speech+Recognition+API>. Acesso em: 17 jun. 2015.

[19] LUMENVOX. *Pizza Demo Application*. Disponível em: <http://www.lumenvox.com/partners/digium/applicationzone/projects/originalPizza.aspx>. Acesso em: 30 jan. 2016.

[20] UNIVERSAL SPEECH SOLUTIONS LLC. *UniMRCP Open Source Project*. Disponível em: <http://www.unimrcp.org/>. Acesso em: 17 jun. 2015.

ção desse segundo método com o Julius, sendo mencionada pelo desenvolvedor do UniMRCP a existência de uma solução comercial[21] . O desenvolvimento dessa solução também exige alta disponibilidade de tempo e esforço.

O UniMCRP é citado em fóruns da Voxforge como a maneira mais acessível para a utilização de reconhecimento de voz com o Asterisk (VOXFORGE FORUM, 2012). Esse mesmo fórum, no entanto, traz outra opção, indicada como alternativa rápida e atrativa por ter sido utilizada com sucesso para um variado número de linguagens, gramáticas, modelos de áudio, etc. Essas características determinaram a motivação pela escolha dessa terceira solução no desenvolvimento desse projeto.

Utilizada em Batista (2013), essa última solução permite a integração da ferramenta de RAF Julius com o servidor Asterisk diretamente por meio do processo de EAGI (*Enhanced Asterisk Gateway Interface*) *scripting*. Desse modo é possível se ter acesso ao canal de áudio provindo do servidor PBX, através do descritor de arquivo 3[22] do sistema operacional Linux.

Com isso, o fluxo digital de áudio recebido pode ser gravado em um arquivo de extensão *.raw* ou redirecionado diretamente à ferramenta de reconhecimento, que procederá com a transcrição do comando falado em texto (MADSEN; MEGGELEN; BRYANT, 2011; BATISTA, 2013).

As dificuldades inerentes à implementação da entrega do áudio ao Julius, apresentadas na subseção 4.3.2, determinaram os rumos da solução desenvolvida neste projeto, que realiza a recepção do fluxo de áudio através do descritor de arquivo 3, sendo esse fluxo posteriormente gravado em um arquivo de extensão *.raw*, com dados brutos do áudio, e convertido, em seguida, para o formato *.wav*, conforme ilustrado no diagrama da Figura 22.

A metodologia dessa segunda abordagem abarca ainda procedimentos específicos quanto ao treinamento dos modelos da fala. Nesse sentido, ainda há escassez de recursos disponíveis livremente para a criação de modelos no idioma português brasileiro (BATISTA, 2013). Em virtude de problemas enfrentados com relação a isso, detalhados na subseção 4.3.2.1, optou-se por desenvolver a plataforma para o reconhecimento de comandos no idioma inglês.

O cenário de aplicação especificado gerou, portanto, condições para a criação de um vocabulário de comandos, conforme apresentado na Tabela 8.

É importante observar que o vocabulário de comandos criado para essa nova aplicação exige a definição de uma gramática, que foi constituída pela composição dos comandos em três componentes organizados numa ordem bem definida: Ambiente, Dispositivo, Ação.

Um maior detalhamento quanto às motivações que levaram ao desenvolvimento dessa concepção estão apresentados na subseção 4.3.2, tendo sido orientados por informa-

[21] GOOGLE GROUPS. *UniMRCP Discussion Group*: How to integrate UNIMRCP with Julius? Disponível em: <https://groups.google.com/forum/#!searchin/unimrcp/julius/unimrcp/ekb8VwfGd1U/h3TWST9sfnAJ>. Acesso em: 30 jan. 2016.

[22] WIKIPÉDIA. *File Descriptor*. Disponível em: <https://en.wikipedia.org/wiki/File_descriptor>. Acesso em: 17 jun. 2015.

Tabela 8 – Diagrama de comandos no idioma inglês para interface baseada em ferramentas de RAF integradas a servidor de comunicação.

Ambiente	Dispositivo	Ação
Bathroom	Light	On Off
	Door	Open Close
Kitchen	Light	On Off
	Door	Open Close
External	Light	On Off
Chamber	Light	On Off
	Door	Open Close
	TV	On Off
Hall	Light	On Off
	Door	Open Close
	TV	On Off

Fonte: Autoria própria.

ções da fase de implementação que realimentaram aqui a fase de concepção da plataforma.

4.3.2 Implementação da interface baseada em ferramentas de RAF integradas a servidor de comunicação

Os artefatos produzidos na fase de concepção da plataforma direcionaram os procedimentos descritos nessa seção, que aborda a fase de implementação. Similarmente ao caso da primeira abordagem, partimos do ponto em que já estão finalizados aspectos de configuração relacionados a preparação do ambiente de desenvolvimento com relação ao sistema operacional e bibliotecas de suporte à linguagem Python, conforme descrito na seção 4.1.

Nessa nova fase, inicialmente foi instalado e configurado o pacote Julius. A seção A.4 do Apêndice A detalha os procedimentos realizados para tanto.

De acordo com o estudo apresentado no Capítulo 3, esse *engine* suporta modelos acústicos no formato do HTK. Portanto, a execução do projeto envolveu ainda a instalação e configuração do pacote HTK, bem como a criação dos modelos acústicos utilizando-se esse pacote de ferramentas configurado em PC com o sistema Ubuntu 14.04 LTS Linux, conforme delineado no Apêndice B.

Dando-se sequência à preparação do ambiente de desenvolvimento, foi realizada

a instalação e configuração do servidor Asterisk, detalhadas na seção A.5 do Apêndice A. Para isso, criou-se um plano de discagem utilizando-se suas aplicações para o atendimento e resposta automáticos das chamadas, tendo sido realizados testes com o intuito de se averiguar o correto funcionamento do servidor quanto ao registro de clientes (*softphone*), ao estabelecimento de chamadas e à interação proporcionada pela URA implementada.

Em seguida, buscou-se prover conectividade sem fio à plataforma através da instalação e configuração do *access point*. Conexão esta disponibilizada empregando-se o adaptador WiFi USB Intelbrás WBN900. Os procedimentos de instalação e configuração desse adaptador estão descritos na seção seção A.6 do Apêndice A.

Providos os recursos anteriormente mencionados, instalou-se e configurou-se o *softphone* Zoiper em um *smartphone* com o sistema operacional Android instalado. Daí, foi possível a conexão do *smartphone* à plataforma via *access point* configurado, bem como realizando-se, assim, o registro do referido *softphone* no servidor Asterisk, sendo possível a realização de chamadas para a URA.

O desenvolvimento dos modelos acústicos e outras questões envolvidas com esse aspecto são apresentadas na subseção 4.3.2.1. Em sequência, a subseção 4.3.2.2 apresenta uma visão geral do funcionamento do Asterisk e seus principais componentes na medida em que são descritos ao longo do texto os desafios enfrentados e as soluções encontradas para a configuração da URA definitiva no servidor Asterisk e sua integração com o Julius através de uma aplicação EAGI.

4.3.2.1 Treinamento dos comandos de voz para a interface baseada em ferramentas de RAF integradas a servidor de comunicação

O treinamento dos comandos de voz concernentes à implementação dessa interface baseada em ferramentas de *software* integradas a servidor de comunicação está condicionado à disponibilidade de modelo acústicos e de linguagem para o Julius. Uma vez que se tenha tais modelos, o *engine* será capaz de receber uma sequência de palavras pronunciadas e retornar sua representação textual.

Um modelo acústico no formato do HTK para o idioma inglês foi inicialmente criado, pautando-se nas informações e tutoriais encontrados no *blog* Aonsquared[23] , mantido pelo engenheiro aeroespacial Arthur Amarra[24] , bem como na página do projeto Voxforge. Utilizando-se de um dicionário fonético fornecido pelo referido projeto, adaptou-se experimentos no sentido de se adquirir o *know-how* fundamental para a criação de modelos acústicos com o HTK.

Assim, durante a realização desse experimento, foi elaborada a gramática de estados finitos e o dicionário de pronúncias responsável por relacionar as gravações de áudio a texto

[23] AONSQUARED BLOG. *Raspberry Pi Speech Recognition*. 2015. Disponível em: <http://www.aonsquared.co.uk/node/9>. Acesso em: 18 jun. 2015.

[24] AON2. *About Us*. Disponível em: <http://www.aon2.co.uk/about>. Acesso em: 18 jun. 2015.

escrito para a criação dos modelos acústicos.

Tomou-se o cuidado para que as palavras contidas nesse dicionário de pronúncias apresentem uma distribuição fonética balanceada, de modo que a quantidade de vezes em que aparecem os fonemas dos comandos definidos na Tabela 8 seja o mais equilibrada possível. Esses componentes do projeto estão disponibilizados no Apêndice B juntamente com os procedimentos de instalação e um tutorial para a criação de modelos acústicos com o HTK.

A ideia inicial do projeto era de que a interface pudesse ser construída para a execução de comandos de voz no idioma português. No entanto, os recursos de base de dados (*corpora*) para esse idioma ainda são escassos (SAMPAIO NETO, 2006; SANTOS JÚNIOR, 2009; SILVA; NELSON NETO; KLAUTAU, 2009; NELSON NETO et al., 2010; NELSON NETO et al., 2011; OLIVEIRA et al., 2011; GOMÉZ, 2011; BATISTA, 2013). Dentre os recursos disponíveis pode-se citar a base de dados *Spoltech*, desenvolvida pelo *Oregon Graduate Institute* (OGI) em conjunto com a Universidade Federal do Rio Grande do Sul (UFRGS). Essa base de dados, no entanto, não é gratuita(SILVA; NELSON NETO; KLAUTAU, 2009).

Outra base de dados de áudio com a fala no idioma português brasileiro que deve ser mencionada foi desenvolvida por Ynoguti e Violaro (2008 apud SILVA, 2011) e utilizada em Silva (2011). Entretanto, essas bases não atendem aos requisitos da aplicação desenvolvida neste trabalho, que necessita de arquivos gravados com as características (taxa de amostragem, bits por amostra, quantidade de canais, nível de ruído, dispositivo de entrada de áudio) compatíveis com as do canal de áudio VoIP através do Asterisk, como também que se constituam de um vocabulário específico voltado para o controle de dispositivos num ambiente doméstico.

Uma alternativa pensada foi a utilização dos modelos disponibilizados pelo projeto Fala Brasil[25] , do Laboratório de Processamento Digital de Sinais (LaPS)[26] da Universidade Federal do Pará (UFPA). Os modelos acústicos do Fala Brasil, contudo, também não apresentam as características compatíveis com a aplicação aqui desenvolvida.

Então, a outra possibilidade seria a criação das bases de áudio no idioma português para, a partir delas, serem gerados os modelos acústicos através da ferramenta HTK. Esse procedimento, todavia, requeria a disponibilidade de um dicionário fonético para esse idioma que pudesse ser aplicado e adaptado ao processo descrito no Apêndice B.

A criação ou adaptação desses dicionários não é uma tarefa trivial e, em geral, é dispendiosa, uma vez que são arquivos constituídos de milhares de palavras de um determinado idioma com suas respectivas transcrições fonéticas. Existe um número bem menor de estudos dedicados ao português brasileiro nessa área quando comparado à língua inglesa, por exemplo (SILVA; NELSON NETO; KLAUTAU, 2009).

[25] FALABRASIL: Reconhecimento de Voz para o Português Brasileiro. Disponível em: <http://www.laps.ufpa.br/falabrasil/>. Acesso em: 12 jun. 2015.

[26] UFPA. *Laboratório de Processamento de Sinais*. Disponível em: <http://laps.ufpa.br/>. Acesso em: 28 jan. 2016.

As dificuldades enfrentadas durante as investidas para a utilização do idioma português acabaram consumindo excessivo tempo de desenvolvimento, tendo sido tomada a decisão de garantir a validação da plataforma utilizando-se o idioma inglês, de modo que sua adaptação para o português passa a ser sugerida dentre os possíveis caminhos de evolução da interface.

No caso do idioma inglês, se optou pela criação de novos modelos pois, apesar da disponibilidade de modelos nesse idioma (como nos projetos CMUSphinx e Voxforge), eles, em sua maioria, também não são compatíveis com as características específicas do cenário de aplicação aqui definido. Quando apresentam certa similaridade, são geralmente compostos de vocabulário com palavras de uso geral, de modo que o treinamento de novos modelos acústicos é recomendado nos casos em que há necessidade de um modelo especializado para aplicação com vocabulário reduzido (BATISTA, 2013; CMU SPHINX, 2015b), como é o caso desse projeto.

Assim, realizou-se a gravação das amostras de treinamento no idioma inglês, seguindo-se as especificações referidas na seção B.2 do Apêndice B, inicialmente para a voz de um locutor do sexo masculino, com idade entre 18 e 60 anos. A gravação dessas amostras, compostas em quarenta arquivos de áudio com as palavras constantes numa lista de *prompts*, ilustrada no Apêndice B (Figura 43), ocorreu em ambiente natural com médio nível de ruído, oscilando na faixa dos 50dB.

A aferição do nível de ruído do ambiente foi realizada empregando-se o mesmo método apresentado na subseção 4.2.3.

As gravações foram realizadas com entrada de áudio a partir de um *headset* conectado ao *smartphone*, através de uma chamada originada do *softphone* para uma extensão da URA, apresentada na subseção A.5.1.2 do Apêndice A e configurada no servidor Asterisk com o intuito de gravar o canal de áudio da chamada. Assim, as amostras de treinamento passam a apresentar as características do canal VoIP, com taxa de amostragem de $8kHz$, 16 bits por amostra, monocanal, sendo armazenadas em arquivos no formato *.wav*.

A cada amostra gravada, o *software* de gravação e edição de áudio Audacity[27] foi utilizado para se averiguar os níveis de energia do áudio, como também para realizar a exclusão de períodos muito longos de silêncio no início e no final de cada arquivo de áudio.

De posse dessas amostras, foi treinado o modelo acústico para a fala do primeiro locutor participante da pesquisa.

O sistema é dependente do locutor e capaz de reconhecer os comandos com três palavras, utilizando modelo de linguagem de estados finitos. O dados de entrada do processo de treinamento foram parametrizados em sequências de vetores característicos utilizando-se a configuração padrão dos parâmetros recomendada por Young et al. (2009), como também pelos tutoriais do projeto Voxforge[28] .

[27] AUDACITY. Disponível em: <http://audacityteam.org/>. Acesso em: 04 jul. 2015.

[28] VOXFORGE. *How-to*: Create acoustic model - with script. Disponível em: <http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/how-to>. Acesso em: 03 jul. 2015.; VOXFORGE. *Tutorial*: Create

Assim, a parametrização dos dados de entrada foi configurada empregando-se a janela de *Hamming* com tamanho de $25ms$ e deslocamento de $10ms$, sendo aplicada para as amostras em cada janela um coeficiente de pré-ênfase de 0.97. Para a análise espectral dos dados utilizou-se um banco de filtros com 26 canais, baseado na transformada rápida de Fourier (FFT), com filtros triangulares igualmente espaçados ao logo da escala Mel.

Em seguida, aplica-se a DCT ao banco de filtros para a extração dos 12 primeiros coeficientes MFCCs, os quais foram reescalonados com um deslocamento cepstral de 22 amostras. A normalização da média cepstral foi calculada e subtraída dos MFCCs para compensação dos efeitos do canal de áudio.

Parâmetros adicionais como a utilização do MFCC C_0 e coeficientes delta foram agregados. O qualificador para supressão da energia absoluta foi mantido ativado. A Tabela 9 apresenta os parâmetros de configuração do HTK utilizados.

Tabela 9 – Parâmetros de configuração utilizados no HTK.

Parâmetro	**Função**	**Valor**
USEHAMMING	Habilitar uso da janela de Hamming	TRUE
WINDOWSIZE	Tamanho da janela	250000
TARGETRATE	Deslocamento da janela	100000
PREEMCOEF	Coeficiente de pré-ênfase	0.97
NUMCHANS	Quantidade de canais do banco de filtros	26
NUMCEPS	Quantidade de coeficientes MFCC extraídos	12
CEPLIFTER	Deslocamento cepstral	22
TARGETKIND	Tipo do vetor de características alvo	MFCC_0_D_N_Z

Fonte: Autoria própria.

Os vetores de características extraídos são posteriormente utilizados para gerar os modelos monofones que são re-estimados para aperfeiçoar os modelos de silencio e de pausas curtas. Esse modelos são clonados e re-estimados utilizando-se as ferramentas do HTK.

Para o treino das HMMs foi utilizado o algoritmo de Baum-Welch (YOUNG et al., 2009). O dicionário fonético disponibilizado pelo projeto Voxforge representa as palavras utilizando 38 fonemas e um modelo de silêncio gerados com HMMs de 3 estados utilizando a estrutura *left-to-right*.

Obtido o conjunto de HMM monofones, o *script*, então, inicia o estágio final da modelagem, que gera os modelos trifones dependentes de contexto. Nessa etapa, inicialmente as transcrições monofones são convertidas em transcrições trifones e um modelo trifone é criado a partir da cópia e re-estimação dos modelos monofones. Em seguida, estados similares dos modelos trifones gerados são vinculados para garantir que a estimação de todas as distribuições de estado seja robusta.

Acoustic Model - Manually. 2016. Disponível em: <http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/tutorial>. Acesso em: 03 jul. 2015.

Os arquivos *hmmdefs* e *tiedlist* gerados pelo HTK com as informações do modelo foram, então, disponibilizados na Beaglebone Black para serem utilizados pelo *engine* Julius.

Esse mesmo processo de gravação das amostras de treinamento foi realizado para 10 locutores, em sessões individuais, com o intuito de se aplicar um conjunto de testes representativo para avaliação do desempenho da interface de comandos por voz proposta.

É, importante ressaltar que, por questões de praticidade, as gravações foram realizadas a partir de um PC, mencionado na subseção 4.3.2, com a configuração do ambiente de gravação descrito nessa seção. Isso, contudo, não afeta o resultado do reconhecimento, uma vez que são mantidas as características do áudio das amostras, pois continuam sendo gravadas através do canal VoIP e registradas em arquivos no formato *.wav*.

4.3.2.2 O servidor Asterisk e sua integração com o Julius

Inicialmente concebido com o intuito de implementar um "PBX (*Private Branch Exchange*) de código aberto", o Asterisk vem desde a década em que foi originalmente lançado, se transformando em uma ferramenta universal para aplicações de comunicação em edifícios.

No entanto, caracterizá-lo simplesmente como um PBX pode ser tanto uma forma de subestimá-lo (uma vez que ele pode ser bem mais que isto), como também de fazê-lo parecer exageradamente complexo (uma vez que ele pode ser bem menos) (ASTERISK DEVELOPMENT TEAM, 2013).

Compatível com as mais variadas linguagens de programação, o Asterisk é capaz de gerenciar o tráfego de áudio em canais digitais, analógicos e em redes TCP/IP. Desenvolvido sob licença de livre distribuição e código aberto, utiliza as principais tecnologias de comunicação existentes no mercado, dentre elas: linhas telefônicas analógicas, *links* TDM (*Time-Division Multiplexing*) de telefonia digital, VoIP (protocolos SIP, H.323, IAX2, MGCP, Skinny, GoogleTalk, Skype, dentre outros)(KELLER, 2009).

A arquitetura desse *software*, capaz de integrar diferentes tipos de canais e tecnologias, destacando-se no quesito convergência tecnológica, característica de relevante impacto para o desenvolvimento de aplicações no paradigma da IoT (KELLER, 2009; MADSEN; MEGGELEN; BRYANT, 2011; COETZEE; EKSTEEN, 2011; DIAS; LUCENA; SANTOS, 2014; PERERA et al., 2014).

As mais recentes versões do Asterisk não suportam somente a função de PBX, mas também implementam funcionalidades como URAs, correio de voz, conferência, distribuição automática de chamadas e todos os tipos de outras aplicações que envolvam comunicação em tempo real. Permitem agregar outras funções através de seu próprio plano de discagem, de módulos adicionais escritos em linguagem C, ou ainda utilizando-se de AGI *scripting* (MADSEN; MEGGELEN; BRYANT, 2011).

O Asterisk é dividido em módulos, de modo que uma vez carregado no sistema cada módulo é responsável por dar suporte a uma funcionalidade específica do *software*. Esse módulos são carregados de acordo com o conteúdo do arquivo */etc/asterisk/modules.conf*

(MADSEN; MEGGELEN; BRYANT, 2011). A instalação padrão realizada para o desenvolvimento desse projeto, descrita na seção A.5 do Apêndice A, já está configurada por padrão para carregamento dos módulos necessários ao desenvolvimento da interface de comandos por voz.

O comportamento do Asterisk é definido com base em diversos arquivos de configuração distribuídos em diretórios do sistema Linux. As definições para a implementação da URA utilizada na plataforma desse projeto concentram-se no conteúdo dos arquivos *sip.conf* e *extensions.conf*, que localizam-se no diretório */etc/asterisk*, como também no código desenvolvido em linguagem Python para integração com o Julius e controle das GPIO da Beagleboard, estando este último localizado na pasta */usr/share/asterisk/agi-bin*.

O coração do Asterisk é seu plano de discagem. Todos os canais que chegam ao sistema passam através do plano de discagem, que contém um *script* com o fluxo das chamadas, determinando como elas serão tratadas. Esse plano de discagem é escrito no arquivo */etc/asterisk/extensions.conf*. O plano que integra a capacidade de reconhecimento dos comandos de voz à interface desenvolvida é apresentado na subseção A.5.1.3 do Apêndice A.

Para que uma chamada realizada à URA seja direcionada ao referido plano de discagem é necessária a configuração de um canal, para cada dispositivo do usuário, indicando-se o contexto onde o ramal (ou extensão) de destino está configurado no plano de discagem.

A criação dos canais é determinada no arquivo */etc/sip.conf*, uma vez que foi escolhido o protocolo SIP para comunicação VoIP entre o dispositivo cliente e o servidor. A escolha desse protocolo se justifica em função de sua simplicidade e versatilidade, o qual se configura em um padrão aberto especificado pela *Internet Engineering TaskForce* (IETF)[29] . O arquivo de criação de canais configurado é apresentado na subseção A.5.1 do Apêndice A.

Conforme observado no plano de discagem criado, a aplicação EAGI é utilizada para executar o *script* Python desenvolvido, disponibilizado no Anexo C.

Uma vez que a integração entre os componentes da interface foi realizada, os comandos de voz provindos de uma chamada discada ao ramal da referida URA podem ser reconhecidos. O Julius, então, retorna a representação textual do comando pronunciado, e essa saída é posteriormente comparada com os comandos previamente definidos no cenário de aplicação da plataforma, levando o sistema a tomar decisão e gerar os sinais de controle dos dispositivos correspondentes no ambiente doméstico hipotético representado no protótipo desenvolvido.

De início, o propósito de desenvolvimento foi o de integrar o RAF ao Asterisk através do direcionamento do fluxo de áudio do canal de uma chamada diretamente à entrada do Julius. Esse caminho, contudo, se apresentou mais demorado, uma vez que exige a implementação de um *plugin* para o *engine* (BATISTA, 2013).

Decidiu-se, então, por direcionar o fluxo de áudio proveniente do descritor de arquivos

[29] IETF. *SIP*: Session initiation protocol. 2002. Disponível em: <https://www.ietf.org/rfc/rfc3261.txt>. Acesso em: 31 jan. 2016.

3 para um arquivo de dados brutos de áudio, no formato *.raw* e fornecer esse arquivo à entrada do Julius para o reconhecimento. Essa solução, entretanto, apresentou problemas relacionados à compatibilidade de formatos de áudio, uma vez que o arquivo *.raw* não estava sendo lido corretamente pelo Julius.

O esforço empreendido para solucionar esse problema acabou trazendo resposta a alguns questionamentos que surgiram. Suspeitou-se de que a incompatibilidade poderia estar sendo causada em virtude do formato de áudio determinado pelo *codec* (Codificador/-Decodificador) utilizado do canal do Asterisk. Entretanto, o projeto Voxforge afirma em um documento sobre a criação de modelos acústicos que:

> Para Voz sobre IP, o *codec* utilizado usualmente determina a taxa de amostragem e bits por amostra na transmissão da voz. O modelo acústico deve ser treinado com dados de áudio que tenham as mesmas características utilizadas no canal de voz (taxa de amostragem e bits por amostra). No caso específico do Asterisk, o áudio é novamente amostrado internamente para 8kHz, com 16 bits por amostra, independentemente da taxa e bits do *codec* utilizado. Portanto, o Asterisk necessita de um modelo acústico treinado com dados de áudio amostrado a 8kHz e 16 bits por amostra[30] .

Assim, no caso do servidor Asterisk, o *codec* não influencia o formato do áudio entregue no referido descritor de arquivo.

Outra suspeita, relacionada à ordem de organização dos *bits* de dados (*endianness*) no arquivo *.raw*, acabou sendo confirmada. Na seção 3.2 do Capítulo 3 é informado, de acordo com LEE (2010), que o Julius somente realiza a leitura de arquivos com extensão *.raw* no formato *big endian*.

Por outro lado, diversas fontes que trazem informações sobre o processador ARM Cortex-A8 AM335x Sitara da fabricante Texas, que é a CPU (*Central Processing Unit*) utilizada na Beagleboard, apontam para o fato de que, apesar de essa unidade central de processamento oficialmente suportar ambos formatos *big* e *little endian*, sua constituição é definida, por padrão, nesse último formato citado[31] .

Essa característica, exige que sejam utilizados módulos conversores encapsulando os *drivers* do processador, uma vez que o suporte oferecido por este aos sistemas Linux embarcados somente trabalha com o formato *little endian*[32] .

A alternativa encontrada para essa questão foi o emprego do utilitário *sox* do Linux embutido no código do *script* EAGI desenvolvido em linguagem Python, através da chamada *subprocess.call*, para realizar a conversão do arquivo de áudio no formato *.raw* para o formato *.wav*.

---

30 VOXFORGE. *Acoustic Model Creation*. Disponível em: <http://www.voxforge.org/home/docs/acoustic-model-creation>. Acesso em: 30 jan. 2016.

31 TEXAS INSTRUMENTS. *Sitara Processors Forum*: Big Endian support feasibility on AM335x chip from silicon design point view. Disponível em: <https://e2e.ti.com/support/arm/sitara_arm/f/791/p/431713/1543339>. Acesso em: 30 jan. 2016.; SIGROK. *Embedded*. Disponível em: <https://sigrok.org/wiki/Embedded>. Acesso em: 30 jan. 2016.

32 TEXAS INSTRUMENTS. *Sitara Processors Forum*: the big-endian and little-endian settings for am335x. Disponível em: <https://e2e.ti.com/support/arm/sitara_arm/f/791/t/274171>. Acesso em: 30 jan. 2016.

Consequentemente, o arquivo salvo nesse último formato pode ser adequadamente fornecido à entrada do Julius para execução do reconhecimento do comando de voz. Esse direcionamento do arquivo de áudio à entrada do *engine* foi implementado utilizando-se o construtor *Popen* da função *subprocess* do Python, sendo passado como parâmetro para o Julius um arquivo de configuração, disponibilizado no Anexo D, com as opções de entrada configuradas para receber um arquivo de áudio.

Os comandos reconhecidos pela plataforma se caracterizam em traduções do áudio em texto (*Speech-to-Text*). Então, foi criado um laço *while* que inicialmente lê a saída do Julius com o resultado do reconhecimento e armazena essa saída em uma variável.

Como a saída do Julius apresenta uma série de informações além do texto referente ao comando reconhecido, posteriormente é aplicado um código para a detecção de expressões regulares que identifica as *strings* correspondentes ao comando pronunciado e armazena em uma variável somente o texto com as palavras do comando.

Dando continuidade ao código, são executados testes condicionais (*if/else*) comparando-se o valor dessa última variável mencionada com o texto dos comandos preestabelecidos no vocabulário da aplicação. Dessa forma, o sistema identifica os dispositivos a serem controlados pela plataforma de comandos por voz, realizando o controle das GPIO correspondentes.

Uma consideração importante a ser feita reside no fato de que, diferentemente da primeira abordagem, o controle das portas GPIO da BBB para o caso da interface baseada em ferramentas de *software* integradas com servidor de comunicação não pôde ser realizado através da inclusão das bibliotecas da Adafruit.

Essa impossibilidade se deu em função de essa biblioteca ter sido desenvolvida e testada usando somente o acesso às GPIO a partir das permissões de usuário *root* no Linux, conforme explica o desenvolvedor da biblioteca no fórum do portal *github* de desenvolvimento da mesma (COOPER, 2013).

Isso se tornou um problema pois, como o *script* Python é executado a partir da aplicação EAGI do asterisk, sua execução é realizada pelo usuário do servidor no sistema (o usuário *asterisk*). Então, mesmo após se configurar todas as permissões necessárias para o controle das GPIO com o referido usuário, essas permissões não tomaram efeito, ocorrendo o mesmo problema apresentado por outros desenvolvedores no fórum supracitado.

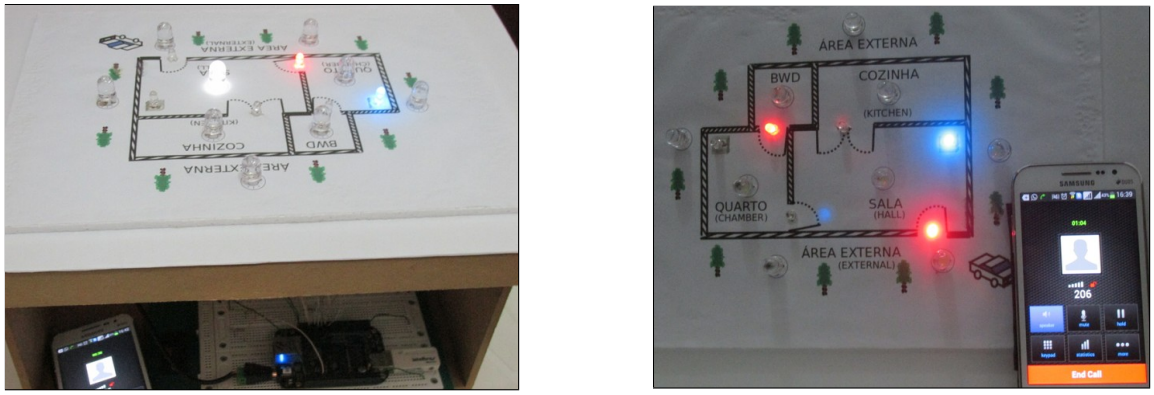
A solução utilizada para resolver esse problema foi encontrada nesse mesmo fórum, a partir da execução de um *script* que altera as permissões das GPIO diretamente nos arquivos através dos quais elas são manipuladas, conforme o procedimento descrito no seção A.7.

Dessa forma, pode-se utilizar comandos em linguagem *shell* para o controle das GPIO na Beagleboard. Assim, esses comandos são passados a partir do *script* Python para a URA através da interface EAGI. Daí, foi empregada a aplicação *System()* do plano de discagem do Asterisk para passar os comandos *shell* ao sistema Linux, controlando-se os pinos GPIO.

Com isso, a interface de comandos por voz para o auxílio de pessoas com mobilidade reduzida passou a cumprir com seu propósito respondendo de forma coerente aos comandos

de voz fornecidos à sua entrada. Visando ilustrar o cenário de aplicação definido, foi construído um protótipo para o acendimento das luzes de LEDs dispostos no referido cenário de modo a representar os dispositivos domésticos a serem controlados, conforme apresentado na Figura 23.

Figura 23 – Protótipo desenvolvido.



Fonte: Autoria própria

Os pinos GPIO utilizados e o respectivo dispositivo a ser controlado estão apresentados na Tabela 10.

Tabela 10 – Mapa de pinos para o controle dos dispositivos do ambiente doméstico.

Ambiente	Dispositivo	Pino na Beagleboard
Bathroom	Light	P8_17 (27)
	Door	P8_14 (26)
Kitchen	Light	P8_18 (65)
	Door	P8_13 (23)
External	Luz 0	P8_8 (67)
	Luz 1	P8_7 (66)
	Luz 2	P8_16 (46)
	Luz 3	P9_12 (60)
	Luz 4	P8_26 (61)
Chamber	Light	P8_11 (45)
	Door	P8_10 (68)
	TV	P8_15 (47)
Hall	Light	P8_12 (44)
	Door	P8_9 (69)
	TV	P8_19 (22)

Fonte: Autoria própria.

A estrutura do protótipo foi montada para a realização dos testes de avaliação do desempenho da interface, com a vantagem de ser facilmente transportado uma vez que foi construído utilizando-se de uma caixa constituída do material MDF (*Medium-Density Fiberboard*), com as seguintes dimensões: 36,5cm (largura) x 25,5cm (altura) x 26,5cm (profundidade).

4.3.3 Testes realizados com a interface baseada em ferramentas de RAF integradas a servidor de comunicação

A realização dos testes de avaliação do desempenho da interface de comandos por voz desenvolvida nessa segunda abordagem contou com a participação de 10 sujeitos amostrais, dentre cidadãos brasileiros comuns, sendo 5 do sexo masculino e 5 do sexo feminino, com boa capacidade das habilidades da fala preservada, de idade maior ou igual a 18 anos, escolhidos por acessibilidade.

No caso dessa segunda abordagem, foi elaborado e submetido um projeto ao Comitê de Ética em Pesquisa (CEP) da Pró-Reitoria de Pesquisa, Inovação e Pós-Graduação do IFPB, com o intuito de se obter aprovação quanto à observação dos interesses dos participantes da pesquisa em sua integridade e dignidade e para contribuir no desenvolvimento da pesquisa dentro de padrões éticos (PRPIPG-IFPB, 2016).

O projeto submetido, como também o modelo do termo de consentimento livre e esclarecido (TCLE) para participação na pesquisa, estão disponíveis no Anexo E e no Anexo F, respectivamente. O projeto foi aprovado conforme o parecer consubstanciado do CEP, de nº 1.350.868, constante no Anexo G.

Dessa forma, foram agendadas as sessões para gravação das amostras de treinamento, de acordo com a disponibilidade de cada participante, sendo fornecidas todas as informações quanto aos objetivos da pesquisa e procedimentos aos quais os mesmos seriam submetidos, bem como os possíveis riscos decorrentes de sua participação na pesquisa. Todos os participantes assinaram o Termo de Consentimento Livre e Esclarecido (TCLE).

Os testes foram realizados em um ambiente mais próximo a uma situação real de utilização da interface de comandos por voz, com um nível de ruído na faixa de 50 a 78 dB, conforme medição realizada através do aplicativo decibelímetro, devidamente calibrado, a partir de um *smartphone*. Tanto para as sessões de gravação, como para as de testes, foi registrado um tempo médio de duas horas de duração.

Para a realização dos testes, a plataforma foi treinada com a voz de cada indivíduo participante da pesquisa, em sessões individuais, observando-se a metodologia apresentada na subseção 4.3.2.1.

Uma vez gravadas as amostras de treinamento, os modelos acústicos foram gerados utilizando-se as ferramentas do pacote HTK. Então, deu-se sequência ao agendamento das sessões para a realização dos testes de avaliação do desempenho da plataforma.

Durante os testes, cada locutor forneceu à entrada do sistema 20 repetições para cada um dos comandos definidos na Tabela 8, tendo-se observado o comportamento do sistema, registrado-se esses dados em planilhas. Dessa forma, realizou-se a execução de 4400 comandos com a plataforma.

Aqui, diferentemente dos testes realizados para o caso da primeira abordagem, não houve a necessidade de registro das ocorrências do tipo “Sem resposta”, uma vez que o sistema

apresentou uma resposta para todas as vezes em que um comando de voz é fornecido à sua entrada.

Posteriormente, foi então realizada uma análise dos dados coletados, tendo sido registrados os casos em que o sistema comportou-se conforme esperado, em relação à taxa de acerto, como também observando-se a baixa ocorrência de falsos positivos[33].

Com o intuito de observar o compromisso de não identificar o nome dos participantes na divulgação dos resultados da pesquisa, foi atribuída uma nomenclatura referindo-se a cada um como sendo do Locutor "X", em que "X" é um número que varia de 1 a 10. A análise dos resultados é apresentada no Capítulo 5.

Apesar de se propor um sistema pensado para ser aplicado no controle de dispositivos do ambiente doméstico para o auxílio de pessoas com mobilidade reduzida, deu-se preferência pela participação de locutores que não apresentam esta limitação física, uma vez que o principal requisito relacionado aos testes de reconhecimento consiste na participação de pessoas com boas habilidades da capacidade da fala preservadas.

Esse direcionamento foi tomado com base nas orientações do CEP do IFPB, com o intuito de não expor indivíduos já em situação de limitação física aos riscos inerentes à pesquisa, os quais estão descritos no projeto submetido ao referido comitê, conforme o Anexo E.

4.4 Discussão

O desenvolvimento da interface de comandos por voz proposta nesse trabalho envolve a integração entre diversas tecnologias, de modo que a aplicação de uma metodologia de desenvolvimento se apresenta como uma boa prática quanto ao acompanhamento da evolução do projeto. Nesse sentido, constituiu-se um processo em três fases: concepção, implementação e testes.

Assim, os procedimentos inerentes à concepção, construção e avaliação do desempenho da plataforma puderam ser gradualmente executados e avaliados a partir da geração de artefatos produzidos a cada fase do projeto. De modo geral, a fase de concepção proporcionou a organização das informações quanto ao cenário de aplicação da interface, bem como sua constituição em função das tecnologias disponíveis para sua implementação, e ainda fundamentando o estabelecimento de um vocabulário de comandos de voz a serem utilizados.

Quanto à fase de implementação, esta foi orientada pelos produtos gerados na fase de concepção, que proveram as informações necessárias à preparação do ambiente de desenvolvimento, além de nortear o processo de treinamento dos comandos de voz e também a programação da interface. Por fim, a fase de testes estabelece uma metodologia de avaliação do desempenho do protótipo final desenvolvido.

[33] Quando o resultado da sentença reconhecida divergia da sentença pronunciada pelo participante.

As tecnologias empregadas durante o processo foram especificadas, detalhando-se seus componentes e seu emprego no contexto da pesquisa. A integração entre tais tecnologias foi apresentada juntamente com diagramas de blocos que dão uma visão geral dos componentes de cada interface desenvolvida, bem como do relacionamento entre eles.

O vocabulário e gramática dos comandos em cada caso foi representado em tabelas, tomadas como base no processo de treinamento das interfaces e/ou de criação dos modelos acústicos a serem empregados.

A primeira abordagem desenvolvida, que se constituiu de um experimento piloto para elucidação de alguns fundamentos práticos e metodológicos quanto ao desenvolvimento de aplicações de comandos por voz, observou uma linha de desenvolvimento voltada para a utilização de um módulo específico de *hardware* no processamento concernente ao reconhecimento dos comandos.

A implementação dessa abordagem teve seu início naturalmente direcionado para a integração entre o módulo de *hardware* utilizado e a plataforma ARM Beaglebone Black. Uma vez que essa integração, que envolve uma programação inicial para a comunicação entre os componentes da interface, foi realizada, pôde-se treinar os comandos de voz no *Voice Recognition Module* V2.

O manual do módulo de reconhecimento de comandos por voz utilizado aponta uma taxa de reconhecimento de 99% num ambiente ideal. No entanto, pensando-se no potencial de integração dessa plataforma em aplicações que auxiliem pessoas com mobilidade reduzida, os testes foram realizados num ambiente próximo às condições reais de operação, com nível de ruído oscilando na faixa dos $50dB$.

Então, observando-se essas condições, o treinamento do módulo de *hardware* foi realizado por 10 locutores, que posteriormente prestaram-se a testar a interface desenvolvida fornecendo a pronúncia de comandos à plataforma repetidamente, permitindo-se avaliar seu desempenho. Os resultados são apresentados no Capítulo 5.

O baixo custo dessa primeira interface, descriminado na Tabela 11, viabiliza sua utilização no desenvolvimento de soluções acessíveis de tecnologias assistivas de comandos por voz.

Tabela 11 – Custo dos componentes utilizados no projeto da primeira abordagem.

Item	**Qtd.**	**Valor (*R*$) em 03/02/2016**
Beaglebone Black	1	250,89
Voice Recognition Module V2	1	88,19
Resistor 220Ω	3	0,30
Resistor 330Ω	5	0,50
Resistor $1K\Omega$	5	0,50
	Total	340,38

Fonte: Autoria própria.

Outra questão interessante a ser também discutida é o fato de que o módulo de

hardware é baseado no processador digital de sinais SPCE061A, que é um controlador de som com arquitetura de 16 bits, desenvolvido pela SUNPLUS Tecnologia, que inclui memória *flash* interna com palavras de 32Kbits, bem como uma memória RAM estática (*Static Random Access Memory* – SRAM) com palavras de 2Kbits(SUNPLUS TECHNOLOGY, 2002).

Essa estrutura pode ser acessada e reprogramada através do módulo V2 aplicando-se um processo de engenharia reversa. Entretanto, com a limitação de *hardware* da placa, que somente disponibiliza acesso aos pinos de comunicação serial UART (*Universal asynchronous receiver/transmitter*) TTL, aos pinos de entrada analógica de áudio para microfone, e a somente 8 dos 32 pinos configuráveis de entrada e saída digital.

A placa *Voice Recognition Module* já possui uma nova versão disponível, conhecida como *Voice Recognition Module* V3. Nessa evolução da placa, os comandos de voz agora são armazenados num único grupo, que funciona como uma biblioteca, capaz de armazenar até 80 comandos. O fabricante fornece as bibliotecas para a utilização do módulo com a placa Arduino[34] .

O novo módulo permite que 7 comandos sejam importados por vez e disponibilizados para reconhecimento ao mesmo tempo. A nova versão do módulo é de baixo custo e pode ser comprada por um preço a partir de $R\$83,00$[35] .

Apesar de ter a limitação de que os comandos gravados só podem ter, no máximo $1500ms$, constituindo-se de palavras isoladas ou de até duas palavras curtas, a quantidade de comandos disponibilizados por essa interface de *hardware* pode ser considerada suficiente para o emprego em aplicações no controle de dispositivos do ambiente doméstico. Especialmente, para o caso do cenário de aplicação definido na segunda abordagem de desenvolvimento dessa pesquisa, em que o vocabulário de comandos se constitui de 22 comandos com 3 palavras cada, apresentando um total de 66 palavras.

Entretanto, na medida em que a quantidade de dispositivos a serem controlados aumenta, essa limitação do módulo se constitui numa desvantagem para sua utilização. Além disso, a possibilidade de se prover a entrada de comandos de voz para o referido módulo a partir de um canal telefônico VoIP, por exemplo, não é uma tarefa trivial, uma vez que seu manual somente menciona a entrada de áudio através de suas interfaces analógicas para microfone.

De toda forma, sua aplicação em contextos de uso não remoto, com a entrada de áudio fornecida a partir do microfone, de modo que a plataforma de comandos por voz deva estar em posse do usuário para ser utilizada é uma solução rápida e de baixo custo para proporcionar o reconhecimento em qualquer idioma.

Por outro lado, a utilização de ferramentas de *software* para o reconhecimento automático da fala tem apresentado uma rápida evolução ao longo dos anos. Essa evolução contribuiu para a redução dos custos dos *softwares* desenvolvidos com esse intuito, ao ponto

[34] ARDUINO. *Arduino*. Disponível em: <https://www.arduino.cc/>. Acesso em: 03 jan. 2016.
[35] ALIEXPRESS. *AliExpress*. Disponível em: <http://pt.aliexpress.com/>. Acesso em: 03 jan. 2016.

de já existirem hoje em dia, inclusive soluções livres, disponibilizadas gratuitamente.

Esse último caso é observado na segunda abordagem, que constitui o objetivo principal desse projeto. Sua linha de desenvolvimento foi voltada para a utilização de ferramentas de*software* integradas a um servidor de comunicação. Nesse caso, o cenário de aplicação definido teve como base o contexto das tecnologias assistivas para o controle dos dispositivos num ambiente doméstico.

Na arquitetura da plataforma de RAF baseada em *software* foi apresentada a inserção dessa interface no cenário de aplicação definido, bem como a interação entre seu usuário e seus componentes de *hardware* e *software*. O servidor de comunicação, bem como as ferramentas de reconhecimento automático da fala empregados foram apresentados e tiveram sua utilização justificada aos longo da descrição dos procedimento executados nas fases de desenvolvimento da interface.

A metodologia envolvendo essa segunda abordagem, diferentemente da primeira, direcionou os rumos de desenvolvimento em função da necessidade de disponibilidade dos modelos acústicos necessários para o reconhecimento da fala. Assim, inicialmente foram realizados os procedimentos de treinamento dos modelos utilizando-se o pacote HTK. Limitações relacionadas à indisponibilidade de recursos para a criação de modelos no idioma português determinaram a utilização do idioma inglês com fins de validar o funcionamento da interface.

Para tanto, o cenário se consistiu de um protótipo que simula um ambiente doméstico hipotético, constituído de 4 cômodos, sendo representado o controle de acesso (portas), iluminação e televisores, a partir do acendimento de LEDs dispostos no protótipo conforme apresentado na Figura 23.

O que chama a atenção no caso dessa segunda interface é sua flexibilidade e liberdade de utilização. Desenvolvida a partir de um servidor de comunicação Asterisk integrado à ferramenta de reconhecimento automático da fala Julius proporciona a entrada dos comandos de voz a partir dos mais variados dispositivos, como: PCs, *tablets*, telefones celulares convencionais, telefones fixos convencionais, telefones VoIP fixos, *softphones* e *smartphones*.

O servidor Asterisk hoje em dia é considerado um *framework*, com suporte a inúmeros serviços e protocolos, provê ainda os meios necessários para a integração entre as diferentes tecnologias de comunicação. Essa característica do *software* representa um enorme valor agregado à interface desenvolvida, uma vez que assim ela passa a dispor de importantes requisitos de convergência tecnológica alinhados à presente revolução da Internet em que a sociedade da informação se encontra, no paradigma da IoT.

Todas as ferramentas de *software* empregadas no desenvolvimento desse projeto são gratuitas e distribuídas livremente. Essa é uma importante característica da interface implementada, que proporciona, então uma maior redução nos custos em comparação com a interface desenvolvida na primeira abordagem. Uma vez que essa segunda necessita unicamente da plataforma ARM, no caso a Beaglebone Black, para ser implementada. Os

custos relacionados aos componentes da interface são apresentados na Tabela 12.

Tabela 12 – Custo dos componentes utilizados no projeto da segunda abordagem.

Item	Qtd.	Valor (*R*$) em 03/02/2016
Beaglebone Black	1	250,89
Adaptador Wireless USB Intelbrás WBN900	1	55,00
Resistor de 330Ω	15	3,00
	Total	308,89

Fonte: Autoria própria.

A utilização do adaptador USB de rede sem fio juntamente com a interface implicaria em um considerável aumento nos custos de um produto final que pudesse vir a ser desenvolvido com a interface proposta. No entanto, sua utilização não é obrigatória, tendo sido escolhida em função de proporcionar versatilidade no desenvolvimento e realização de testes com o protótipo, já que o acesso ao serviço fica disponível diretamente a partir do *access point* configurado, facilitando ainda o transporte do dispositivo.

A redução nos custos das tecnologias assistivas que venham a empregar as interfaces propostas neste trabalho é visível quando comparado aos custos do sistema apresentado por Hall e Molloy (2003), em que, somente a interface de reconhecimento de voz, gera custos da ordem de $329,98 dólares (*R*$1.214,60 em 09 de Março de 2016), o que sinaliza uma redução de gastos de cerca de 71,98% em comparação com a interface desenvolvida na primeira abordagem, e de 74,82% com relação a segunda interface desenvolvida.

Já comparando-se aos custos do dispositivo apresentado em Jiang et al. (2000), que tem um valor previsto da ordem de $200 dólares (*R*$736,17 em 09 de Março de 2016), o dispositivo aqui proposto apresenta-se 53,76% mais barato que a primeira interface desenvolvida, enquanto demonstra-se 58,45% mais barato comparativamente à interface desenvolvida na segunda abordagem.

Como possui a característica de ser um servidor de comunicação, a interface pode ser simplesmente conectada junto à infraestrutura de rede convencional de uma residência, que, em geral, se constitui de um roteador WiFi conectado a um provedor de Internet, podendo, do mesmo modo, ser acessada a partir de uma conexão sem fio, podendo contar, inclusive, com acesso a partir da Internet.

Como foi desenvolvida com o propósito de uso pessoal no auxílio a indivíduos que apresentam dificuldades com relação à sua mobilidade, a plataforma foi concebida utilizando-se de modelos acústicos dependentes de locutor. Nesse aspecto, o treinamento dos modelos ainda é um processo lento e cansativo, em que se exige a gravação de amostras de treinamento com a fala do usuário para que os modelos possam ser gerados.

Esse problema, contudo, pode ser gradualmente eliminado com o tempo, uma vez que a geração de novos modelos para cada usuário contribuirá para o constante aumento da base de dados da fala (*corpora*), dando condições para que sejam criados posteriormente novos modelos com características de independência ao locutor.

Um outro fator interessante observado com o emprego do servidor Asterisk para a implementação da plataforma é que, apesar de no caso dessa segunda abordagem as amostras de treinamento terem sido coletadas em um ambiente com médio nível de ruído, as características de supressão de ruído do *codec* utilizado no canal VoIP contribuíram para uma redução da influência desse aspecto no desempenho do sistema.

As técnicas e parâmetros de configuração utilizadas para o processamento dos sinais da fala no treinamento dos modelos acústicos com o HTK foram apresentadas, bem como foram discutidos e justificados diversos aspectos relacionados à escolha dos *softwares* utilizados. A resolução de problemas enfrentados para se alcançar a integração entre o Julius e o Asterisk na Beaglebone Black foi também descrita.

Os procedimentos de instalação e configuração dos componentes da interface, bem como para o treinamento dos modelos acústicos utilizados foram descritos e são apresentados no Apêndice A e no Apêndice B.

O código-fonte desenvolvido para cada interface, bem como o arquivo de configuração do Julius empregado, também foram disponibilizados no Apêndice A, no Anexo C e no Anexo D, respectivamente.

A Tabela 13 apresenta um resumo com as características inerentes às interfaces de comandos por voz desenvolvidas na primeira e segunda abordagens.

Tabela 13 – Características das interfaces desenvolvidas.

Característica	**Abordagem baseada em módulo da *hardware* específico**	**Abordagem baseada em ferramentas de *software* integradas a servidor de comunicação**
Plataforma ARM	Beaglebone Black	Beaglebone Black
Módulo de hardware específico para RAF	*Voice Recognition Module* V2	Não
Servidor de comunicação VoIP	Não	Asterisk™
Possibilidade de conexão com a Internet	Sim	Sim
Comandos de voz	Palavras isoladas com até 1300ms	Frases com três palavras, podendo-se ampliar a gramática de comandos
Independência / Dependência do locutor	Dependente de locutor	Depende de locutor, podendo-se utilizar um modelo independente de locutor quando disponível
Idioma dos comandos	Pode ser treinada em qualquer idioma	Pode ser configurada para qualquer idioma, exigindo a disponibilidade dos modelos acústicos no idioma desejado
Entrada dos comandos	Microfone diretamente conectado ao módulo de *hardware*	Microfone conectado ao PC, telefone fixo, telefone VoIP, telefone celular GSM, *smartphone*, *softphone*, *tablets*, dentre outros
Mobilidade	A interface deve estar próxima ao usuário	Funciona como servidor central, permitindo que a entrada do comando seja realizada remotamente através de diversos dispositivos

Fonte: Autoria própria.

A realização do estudo, configurações, treinamento de modelos e dos testes necessários utilizou a infraestrutura disponível no laboratório de telefonia e redes convergentes (LTCON) do IFPB.

5 Resultados

Neste capítulo serão apresentados os resultados dos testes de avaliação do desempenho realizados com os protótipos desenvolvidos para as interfaces de reconhecimento automático da fala propostas neste trabalho.

5.1 Resultados dos testes realizados com a interface baseada no módulo *Voice Recognition Module* V2 associado à Beaglebone Black

Os resultados aqui descritos foram alcançados aplicando-se metodologia de testes descrita na subseção 4.2.2.2 do Capítulo 4. Com os comandos de voz devidamente treinados no módulo *Voice Recognition Module* V2, foi executado o procedimento avaliativo da plataforma, de modo que cada locutor participante da pesquisa forneceu à entrada de áudio do sistema, em sessões individuais, 20 repetições para cada um dos comandos estabelecidos no vocabulário da Tabela 5, apresentada no Capítulo 4.

Assim, foram executados 1000 comandos por diferentes indivíduos com a interface, e o comportamento do sistema foi observado, tendo sido registradas as ocasiões em que a plataforma comportou-se conforme o esperado, não gerou resposta, ou ainda, em que houve a ocorrência de falsos positivos. As planilhas com os registros individuais para os comandos fornecidos para cada locutor estão disponíveis no Anexo B.

Os resultados da validação apresentaram uma excelente taxa de acerto para a execução de comandos no idioma português brasileiro, conforme apresentado na Tabela 14. Os testes envolveram tanto vozes masculinas como femininas.

Tabela 14 – Resultados dos testes realizados para a interface baseada em módulo de *hardware*

Locutor (sexo)	**Taxa de acertos (%)**
Locutor 1 (F)	96%
Locutor 2 (F)	97%
Locutor 3 (F)	98%
Locutor 4 (F)	90%
Locutor 5 (M)	97%
Locutor 6 (M)	92%
Locutor 7 (M)	97%
Locutor 8 (M)	95%
Locutor 9 (F)	97%
Locutor 10 (M)	100%
Taxa média geral de acertos	95,9%

Fonte: Autoria própria.

Esses resultados deram base para a produção de um artigo completo, disponível no Anexo H, apresentado no $XXIV$ Congresso Brasileiro de Engenharia Biomédica (CBEB) e publicado nos anais do evento.

5.1.1 Discussão dos resultados

Os resultados permitem concluir que, em ambientes ruidosos, a taxa média global de acertos (de 95,9%) é bastante alta, garantindo a validação do sistema. No entanto, durante a realização dos testes por cada locutor, ocorreram momentos isolados em que o comportamento desviou-se do esperado. Por exemplo, o acendimento do LED verde para o Locutor 5, não funcionou como esperado em 25% das vezes. Além disso, certa quantidade de vezes, o sistema permaneceu inoperante após ter sido fornecido o comando, representando 3,5% das vezes.

A escolha de comandos homófonos, com sonoridade semelhante, aumenta a ocorrência de acionamento indevido dos LEDs, podendo dois comandos resultar na mesma resposta. Há ainda relação entre a ocorrência de erros e o cansaço vocal do locutor, tendo sido as repetições dos comandos realizadas em sequência.

Em algumas vezes observou-se que o sistema foi acionado acidentalmente, influenciado pelo ruído provindo de conversas de terceiros no ambiente de teste (situação próxima da real). Assim, as aplicações desenvolvidas utilizando o dispositivo proposto devem apresentar mecanismos que tratem questões de segurança, fornecendo modalidades de entrada diversas para ativar ou desativar a ferramenta comandado por voz (LU; CHEN, 2012).

A possibilidade de treinamento utilizando a língua português brasileiro permite sua ampla aceitação no país, uma vez que os usuários com mobilidade reduzida, como idosos ou indivíduos com lesão na coluna vertebral, paraplégicos ou tetraplégicos, tendem a preferir utilizar sua própria língua no controle por voz de dispositivos (QIDWAI; SHAKIR, 2012).

A utilização de microcontrolador é uma prática recorrente para esses dispositivos, conforme observado também em Jiang et al. (2000). No entanto, a utilização de unidades de controle baseadas em computador, tendo um módulo externo específico responsável pela tarefa de realizar o reconhecimento de comandos por voz também vem sendo observada na literatura (AGUILERA et al., 1992), onde a unidade de reconhecimento de voz se constitui de uma placa que funciona como *plug-in* para o PC.

Essa prática tem o intuito de retirar o pesado processamento de áudio do computador (Beaglebone Black), de modo que os comandos de voz sejam utilizados somente para sinalizar tarefas, liberando a capacidade de processamento da plataforma para a utilização de recursos avançados que podem exigir alta capacidade de processamento. Um exemplo, seria a associação de diversas outras modalidades de entrada, permitindo o desenvolvimento de aplicações com interfaces multimodais (VALLÉS et al., 1996; HUI; MENG, 2014), de modo a aperfeiçoar a experiência dos usuários com mobilidade reduzida.

Além disso, pode-se também implementar o controle de diversos tipos de dispositivos utilizando protocolos de comunicação avançados, característica concernente ao caráter de uso multidisciplinar e estrutura heterogênea dos sistemas inteligentes de automação, podendo ainda envolver questões de segurança (UZUNAI; BICAKCI, 2007).

Nesse sentido, a grande flexibilidade e compatibilidade com as mais variadas tecnologias proporcionadas com a utilização da plataforma apresentada permitem que sua alta capacidade de processamento esteja disponível para as aplicações que demandem recursos avançados, fazendo com que os sinais enviados pela plataforma para o controle do acendimento e desligamento das luzes do LED RGB possam ser adaptados para comandar os mais diversos tipos de dispositivos. Além disso, o baixo custo viabiliza ainda mais sua utilização no desenvolvimento de soluções acessíveis.

5.2 Resultados dos testes realizados com a interface baseada em RAF com Julius integrado ao servidor Asterisk, embarcados na Beaglebone Black

Com o Julius devidamente configurado para ser utilizado no RAF durante a execução do procedimento avaliativo da plataforma, os procedimentos metodológicos apresentados na subseção 4.3.3 foram realizados para aplicação dos testes de avaliação do desempenho.

Assim, cada participante da pesquisa forneceu à entrada de áudio do sistema 20 repetições para cada um dos comandos da Tabela 8, apresentada no Capítulo 4, tendo sido observado o comportamento do sistema. Foram registradas as ocasiões em que o sistema comportou-se conforme o esperado ou em que houve a ocorrência de falsos positivos.

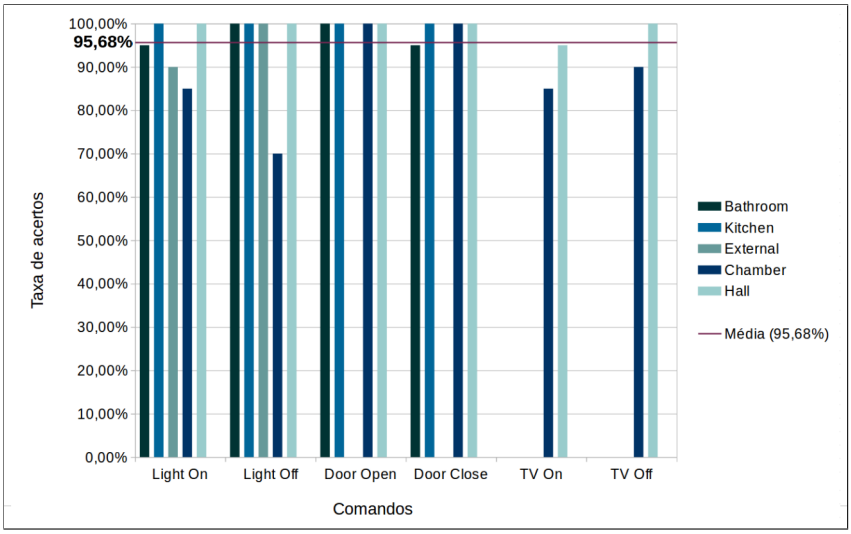
Durante os testes observou-se não ser necessário o registro de ocorrências do tipo "sem resposta", uma vez que o sistema apresentou uma resposta para todas as vezes em que um comando de voz é fornecido à sua entrada. Essas informações foram registradas em planilhas, disponíveis no Anexo I.

Por ser essa abordagem diretamente relacionada ao objetivo principal do projeto, foi realizada uma análise mais criteriosa dos dados, representando-os em gráficos, organizados por locutor, com as taxas de acertos alcançadas para cada comando.

O Locutor 1, que representa uma voz masculina, alcançou a taxa média de acertos de 95,68%. O gráfico da Figura 24 detalha a taxa de acertos para as repetições dos comandos testados por esse locutor. A legenda do gráfico apresenta uma cor para cada ambiente, simulando os cômodos de uma residência. Os comandos referentes aos dispositivos de um cômodo estão distribuídos ao longo do eixo horizontal.

Na Figura 24 já é possível perceber de início uma redução no nível de reconhecimento para alguns comandos específicos, como observa-se para o caso do cômodo *chamber*, em que o comando *light off* chegou a apresentar 70% de taxa de acerto.

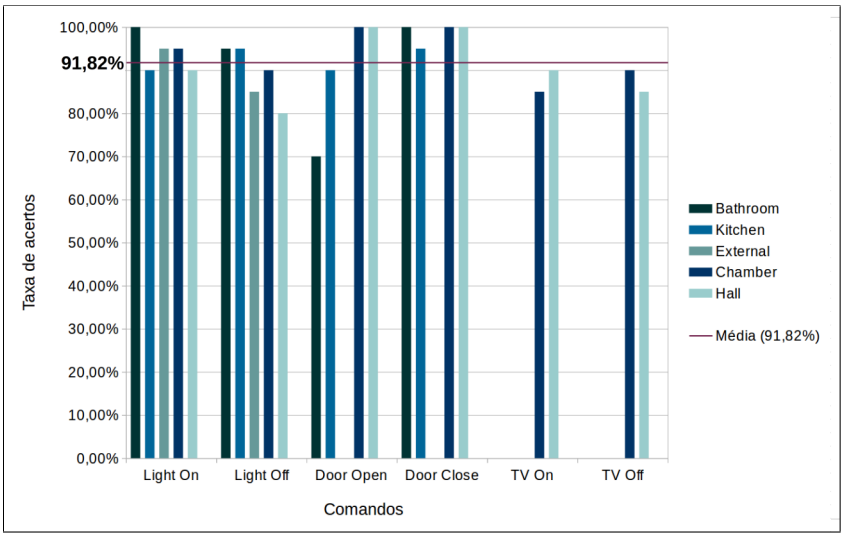
Figura 24 – Taxa de acertos por comando para o Locutor 1.



Fonte: Autoria própria.

O Locutor 2, que representa uma voz masculina, obteve a taxa média de acertos de 91,82%, conforme o gráfico da Figura 25. Nota-se uma variação na taxa de acertos de comandos distintos com relação aos diferentes locutores. Para o caso desse segundo locutor, o comando *door open* do cômodo *bathroom* apresentou a taxa de acertos de 70%.

Figura 25 – Taxa de acertos por comando para o Locutor 2.

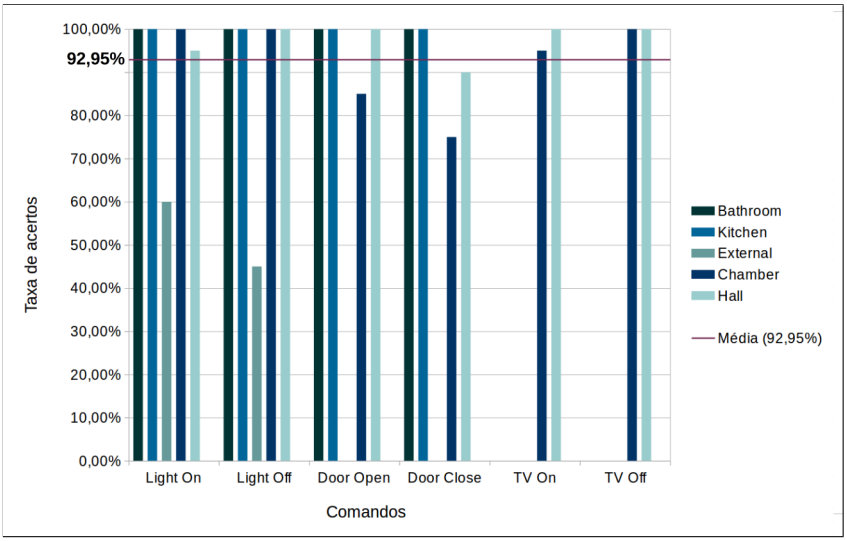


Fonte: Autoria própria.

O gráfico da Figura 26 apresenta as taxas de acertos para os comandos do Locutor 3, que representa uma voz feminina, tendo alcançado a taxa média de acertos de 92,95%. Para

esse terceiro locutor, nota-se uma redução nas taxas de acerto referentes aos comandos da área *external*, com 60% para o comando *light on* e 45% para *light off*.

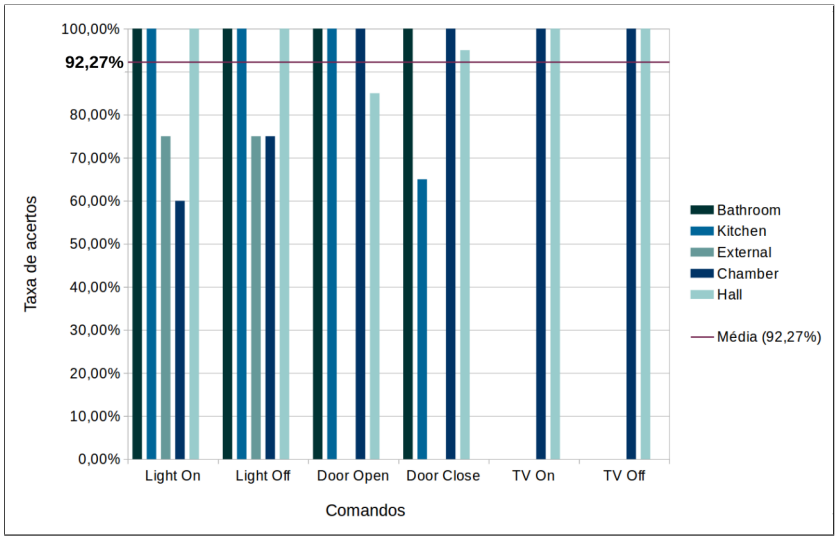
Figura 26 – Taxa de acertos por comando para o Locutor 3.



Fonte: Autoria própria.

A taxa média de acertos de 92,27% no gráfico da Figura 27 detalha o resultado para o Locutor 4, com voz feminina. Neste caso, 77,27% dos comandos apresentaram 100% de acerto, mas houve casos com taxa de acerto de 75% ou menos, como os comandos *light on* e *light off* das áreas *external* e *chamber*, e o comando *door close* do cômodo *kitchen*.

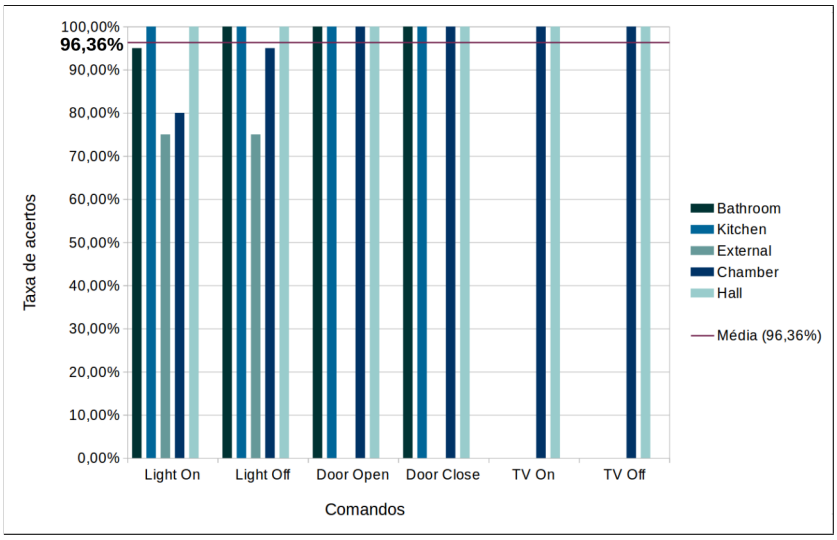
Figura 27 – Taxa de acertos por comando para o Locutor 4.



Fonte: Autoria própria.

Na Figura 28, observamos a taxa de acertos para o Locutor 5, que representa uma voz feminina, com uma taxa média de acertos de 96,36%. Para esse locutor, as taxas de reconhecimento mantiveram-se com níveis a partir de 75%.

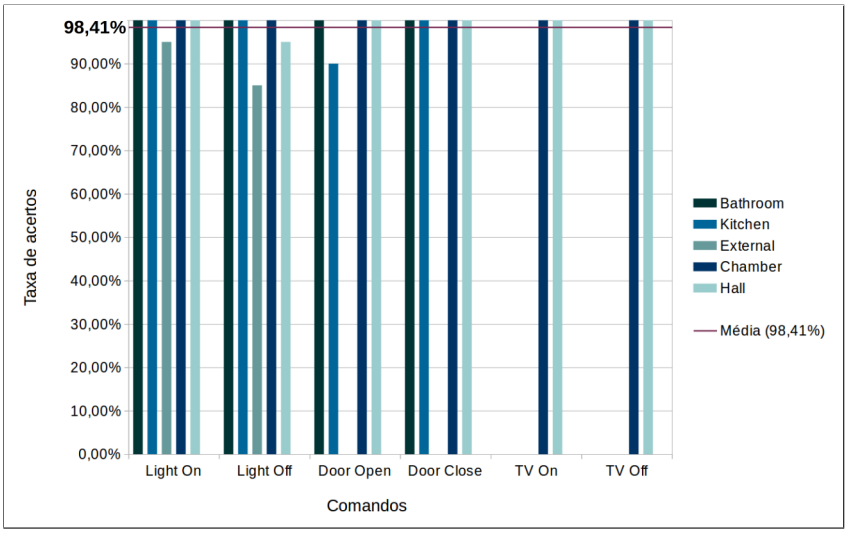
Figura 28 – Taxa de acertos por comando para o Locutor 5.



Fonte: Autoria própria.

O resultado apresentado na Figura 29 refere-se às taxas de acertos alcançadas pelo Locutor 6, que representa uma voz masculina. Apresenta a mais alta taxa média de acertos dentre todos os testes realizados, com 98,41%.

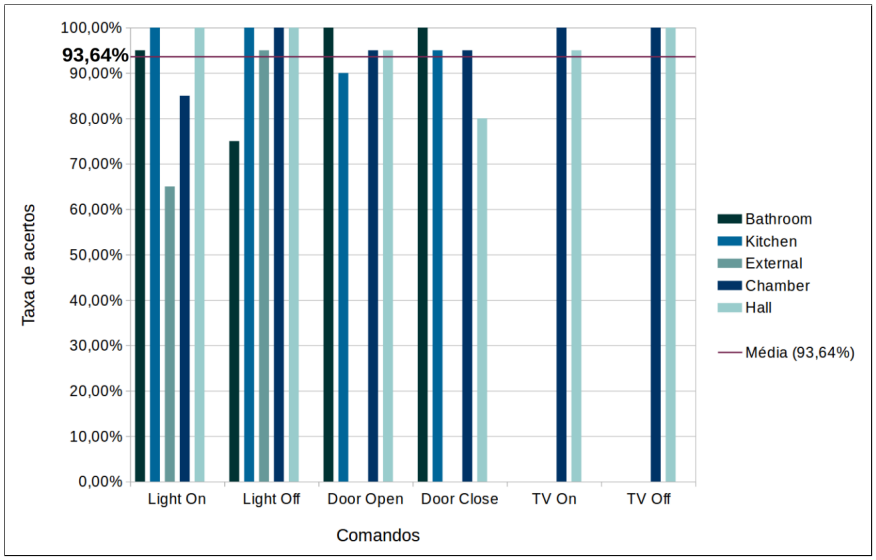
Figura 29 – Taxa de acertos por comando para o Locutor 6.



Fonte: Autoria própria.

A Figura 30 apresenta as taxas de acerto do Locutor 7, que representa uma voz feminina. Sua taxa média de acertos foi de 93,64%. Observa-se outro caso isolado em que um comando referente à área *external* apresenta baixa taxa de acerto, de 65% para *light on*.

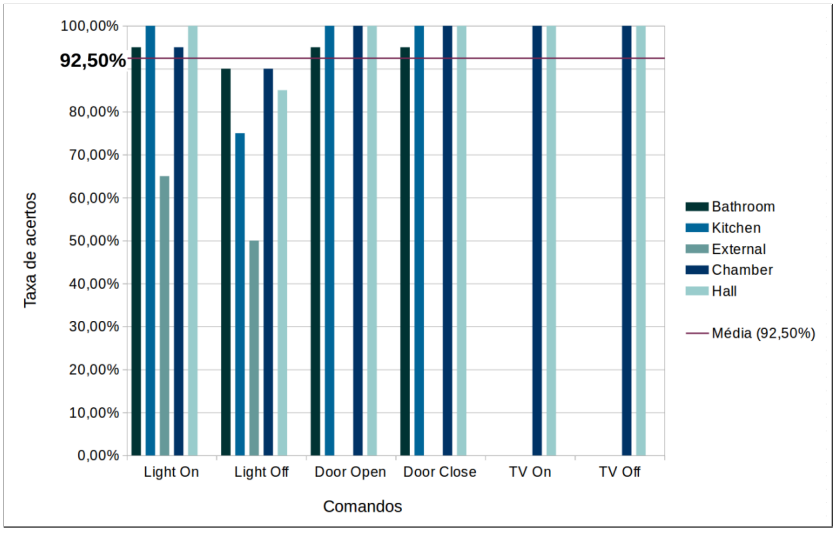
Figura 30 – Taxa de acertos por comando para o Locutor 7.



Fonte: Autoria própria.

A Figura 31 apresenta as taxas de acerto do Locutor 8, que representa uma voz masculina. Sua taxa média de acertos foi de 92,50%. Esses resultados corroboram a recorrente redução no reconhecimento para os comandos relacionados à área *external*.

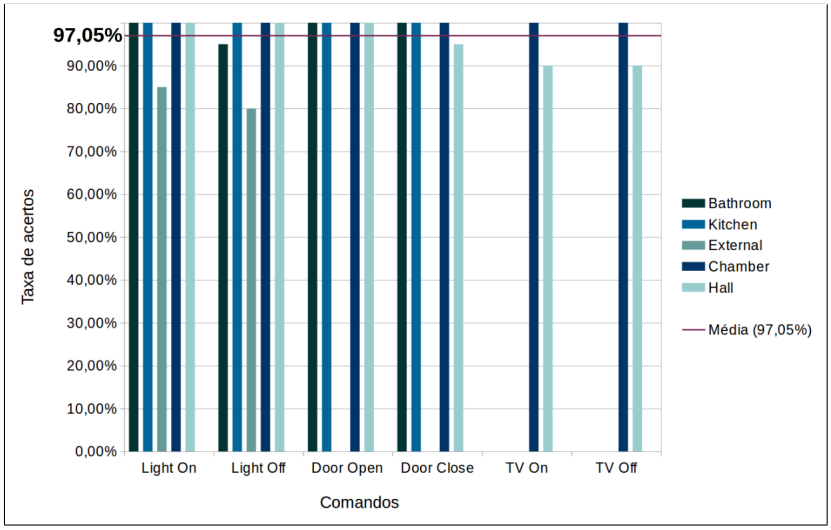
Figura 31 – Taxa de acertos por comando para o Locutor 8.



Fonte: Autoria própria.

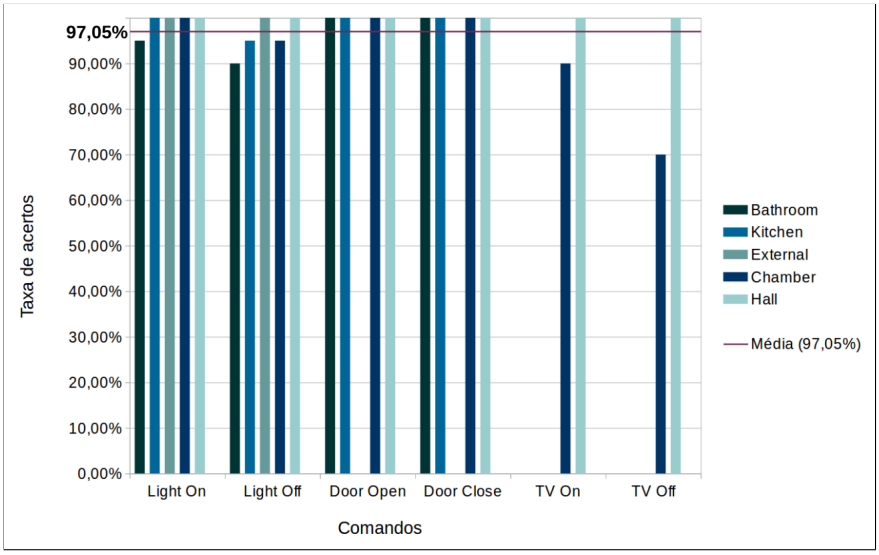
A Figura 32 e a Figura 33 apresentam as taxas de acerto obtidas para os testes realizados pelos locutores 9 e 10, que representam uma voz feminina e uma voz masculina, respectivamente, com taxas medias de acerto de 97,05% em ambos os casos.

Figura 32 – Taxa de acertos por comando para o Locutor 9.



Fonte: Autoria própria.

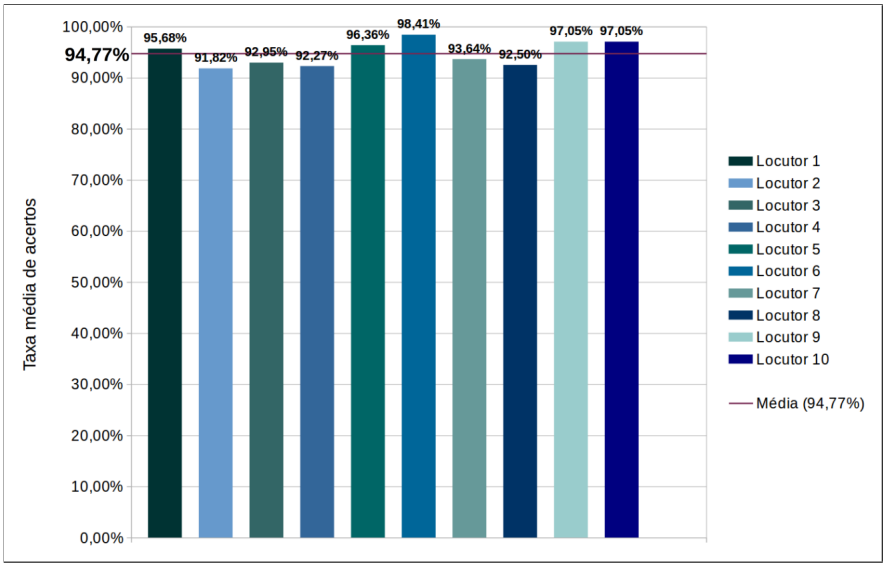
Figura 33 – Taxa de acertos por comando para o Locutor 10.



Fonte: Autoria própria.

Por fim, a Figura 34 apresenta o gráfico com a taxa média global de acertos dos testes de avaliação do desempenho realizados para essa interface de comandos por voz, que foi de 94,77%.

Figura 34 – Taxa média de acertos por locutor.



Fonte: Autoria própria.

5.2.1 Discussão dos resultados

A análise dos dados coletados permitiu concluir que não há diferenças significativas com relação às taxas médias de acerto em função do sexo do locutor, tendo-se alcançado níveis de reconhecimento com valores bem próximos para vozes masculinas e femininas. O estudo realizado por Batista (2013) para a validação de um sistema de atendimento automático de chamadas utilizando reconhecimento de voz também apontam nessa direção, de modo que sua análise exalta aspectos mais relacionados ao canal de comunicação.

A partir dos resultados obtidos, observa-se uma variação característica da taxa de acertos em sensível relação à forma como são pronunciados os distintos comandos pelos diferentes locutores, bem como às características próprias da voz de cada um, mesmo no âmbito de um sistema dependente de locutor.

Isso pode ter sido ocasionado em virtude das diferentes situações de ruído ambiente em cada momento específico da execução dos testes, dado o impacto que o contexto do ambiente onde o reconhecimento é realizado imprime no desempenho dos sistemas de RAF. A influência desse fator no desempenho do reconhecimento vem direcionando diversos estudos na área (SANTOS JÚNIOR, 2009; SCHULLER et al., 2009; LI et al., 2014; SHARMA; MAHESHKAR; MISHRA, 2015).

Além disso, as características de articulação do idioma inglês para cada locutor, bem como outras características de sua fala, também se apresentaram como fatores determinantes no desempenho do reconhecimento das palavras que constituem os diferentes comandos, de acordo com o cômodo, o dispositivo e a ação que se deseja comandar. A criação de modelos acústicos baseados em um dicionário de pronúncias com a ocorrência de fonemas

em quantidade adequada para proporcionar o devido balanceamento fonético minimiza esses efeitos (VOXFORGE, 2016; YOUNG et al., 2009).

A redução do nível de reconhecimento observada em certos casos pode estar relacionada a características de rugosidade (voz rouca) e astenia (pouca intensidade) (FREITAS, 2010) da fala de alguns locutores participantes da pesquisa, em virtude da influência que esses fatores possam ter quanto à qualidade dos modelos acústicos. Com, por exemplo, no caso do Locutor 3, que apresentava uma voz rouca e, consequentemente, de baixa intensidade, tendo sido observado reduzido nível nas taxas de acerto para os comandos relativos à área *external*, e para o comando *door open* do cômodo *chamber*.

Observou-se também que a similaridade fonética entre algumas palavras, como também a entonação imposta pelo locutor para sua pronúncia, acabava por aumentar a ocorrência de falsos positivos. Em alguns casos, o cansaço vocal dos participantes também teve certa influência com relação a apresentação de respostas incoerentes pelo sistema, por exemplo, quanto o participante acabava por diminuir a intensidade de sua fala, ou quando apresentava uma ênfase na pronúncia de determinado fonema ao longo da palavra.

É bastante disseminado na literatura da área o emprego de técnicas de adaptação ao locutor (SILVA; NELSON NETO; KLAUTAU, 2009; BATISTA, 2013; WRIGHT et al., 2013) visando a minimização desses efeitos no desempenho do reconhecimento. Assim, uma melhor compreensão da influência que os fatores de instabilidade da voz (FREITAS, 2010) imprimem ao desempenho do reconhecimento pode configurar-se em uma interessante linha de estudos, fazendo-se, por exemplo, sua correlação com a análise de quantificação de recorrência apresentada em Vieira (2014).

De modo geral, houve uma redução das taxas de acertos quando o comando apresentava palavras que continham fonemas com sons oclusivos (ou plosivos) (FECHINE, 2000; SEARA; NUNES; LAZZAROTTO, 2011) em sua constituição, como: *external*, *chamber* e *kitchen*. Nota-se, por exemplo, uma recorrente inferioridade na taxa de acertos para os comandos do ambiente *external* [ɪkstˈɜrnəl]. Há uma tendência à produção de baixo volume de voz para os fonemas iniciais dessa palavra "[ɪkstˈɜ]"devido à articulação necessária à sua pronúncia.

As transcrições fonéticas aqui apresentadas foram obtidas através do conversor de transcrições fonéticas para o inglês, no portal *EasyPronunciation.com*[1] .

Além disso, notou-se que a redução na taxa de acertos relacionada aos comandos executados para os referidos cômodos tem uma relação com a similaridade fonética entre eles: *chamber* [tʃˈeɪmbər], *kitchen* [kˈɪtʃən] e *external* [ɪkstˈɜrnəl]. Na maioria dos casos de falsos positivos observados, a falha no reconhecimento relacionava-se a errada indicação de uma dessas palavras pelo sistema quando outra delas era pronunciada. Por exemplo, em considerável parte das vezes, a pronúncia de *external* ou de *chamber* levava a um resultado de reconhecimento indicando *kitchen*, e assim sucessivamente.

[1] EASYPRONUNCIATION.COM. *English Phonetic Transcription Converter*. Disponível em: <http://easypronunciation.com/en/english-phonetic-transcription-converter#result>. Acesso em: 05 jan. 2016.

Isso foi observado, inclusive, no caso de locutores que já apresentavam fluência no idioma inglês como, por exemplo, dois professores de língua inglesa que participaram como locutores na pesquisa. Pode, então, se configurar em outra interessante iniciativa, a realização de estudos mais aprofundados que avaliem a influência das características articulatórias dos fonemas sobre o desempenho do reconhecimento.

Em outros casos, foram observadas algumas situações em que a terceira palavra dos comandos, referente à ação, também apresentou uma alta ocorrência de falsos positivos. Num primeiro momento, verificou-se uma relação desse comportamento com o fato de a interface ter sido programada para receber comandos num tempo de 5 segundos, conforme observa-se no Anexo C. Apesar de os participantes da pesquisa terem sido avisados sobre esse tempo, isso pode ter causado problemas em poucas situações, quando o usuário demorou um tempo maior para pronunciar o comando completo.

É importante considerar para os trabalhos de evolução da plataforma a implementação de detecção da fala para que o sistema continue recebendo o áudio de entrada até que o locutor termine completamente a frase, independente das variações de duração entre diferentes locutores ou situações. Esse último caso, no entanto, não foi tão comum, tendo sido os testes repetidos quando se percebeu a ocorrência desse problema, o que pôde ser averiguado a partir da escuta do arquivo de áudio gravado a partir do descritor de arquivos 3.

Em outros casos ainda, percebeu-se que o sistema tendeu a confundir os comandos terminados com as ações *on* [ˈɑn] e *off* [ˈɔf] nos casos em que o locutor realizava pronúncias com baixa intensidade na fala. Notou-se, então, a importância de uma boa articulação do fonema fricativo (SEARA; NUNES; LAZZAROTTO, 2011) da palavra *off* para que o sistema fizesse uma boa diferenciação desses termos.

Um exemplo em destaque, foi o caso do Locutor 6, que possui naturalmente uma fala de baixa intensidade (astenia). Com o intuito de não constranger o participante da pesquisa, as amostras de treinamento foram gravadas uma primeira vez, chamando-se a atenção somente em certos momentos para a necessidade de aumentar a intensidade da voz.

Com isso, durante os testes com o referido locutor, observou-se, logo de início, uma baixa taxa de acerto relacionada aos termos *on* e *off*. Este resultado não havia sido observado até então para os demais locutores, tendo sido registradas taxas de acerto de apenas 65% para o comando *bathroom light on*, como taxas bem menores, de 25% e 15%, para os comandos *kitchen light on* e *kitchen light off*, respectivamente.

Em virtude disso, os testes foram interrompidos com esse locutor para a realização de um melhoramento do modelo acústico gerado a partir da sua voz. Para tanto, as amostras de treinamento foram analisadas, tendo sido regravadas aquelas que continham em sua lista os comandos do cenário da aplicação, e em que a primeira gravação havia ficado com intensidade muito baixa, sinalizando para o cuidado de se obter níveis adequados de intensidade na voz do participante visando observar o impacto no comportamento do sistema.

Esse procedimento indicou uma grande influência existente entre a qualidade da

gravação das amostras de treinamento com relação ao desempenho do sistema. Pôde-se observar que, ainda que não haja uma preocupação com a intensidade das pronúncias durante a realização dos testes, um modelo gerado a partir de amostras de treinamento com intensidade e articulação das palavras adequadas tem um impacto consideravelmente positivo no desempenho do sistema, uma vez que a taxa média de reconhecimento para esse locutor subiu para a faixa de 98,41%.

Com o intuito de realizar mais observações sob esse aspecto, foi aplicada, propositadamente, uma estratégia de orientar somente o essencial para a gravação das amostras de treinamento dos locutores 7 e 8, sem uma observação criteriosa quanto aos níveis de intensidade da voz e clara articulação das palavras, que determinam a qualidade das amostras.

Observou-se que a voz desses locutores apresentava naturalmente uma intensidade maior que a do locutor 6. Com isso, pôde-se averiguar que, apesar de a gravação das amostras de treinamento ter sido realizada a partir de um microfone de *headset* comum, ainda assim obteve-se uma taxa de acertos acima dos 90%, concluindo-se, então, o alto potencial do sistema, que pode alcançar ainda melhores resultados a partir da utilização de modelos acústicos gerados com amostras de treinamento de maior qualidade.

Outra verificação foi realizada durante a participação do Locutor 7, que apresentou em uma primeira sessão taxas de acerto muito baixas com relação aos comandos referentes à área *external*, de 60% e 65%. Então, simplesmente repetiu-se os testes referentes a esses comandos, mantendo os modelos acústicos. No caso do comando *light off*, houve um considerável aumento na taxa de acerto, que subiu para 95%. No entanto, para o *light on*, permaneceu a baixa taxa de acerto.

Nesse sentido, a gravação das amostras de treinamento para os locutores 9 e 10, apesar de ter sido realizada com o mesmo microfone, levou em consideração um maior cuidado com a qualidade da gravação com relação à articulação dos comandos pronunciados, como também com relação ao volume da fala, observando-se altas taxas de acertos para ambos que, inclusive, apresentaram o mesmo valor, de 97,95%, apesar de as ocasiões de erro acontecerem em diferentes comandos.

Assim, apesar da dificuldade apresentada pelo locutor 9 com relação a fala no idioma inglês, uma maior atenção em vista da qualidade das pronúncias realizadas na gravação das amostras de treinamento, determinando um modelo acústico de mais qualidade, proporcionou o alcance de uma alta taxa de reconhecimento.

De modo geral, a taxa média de acertos desempenhada pelo sistema para cada locutor apresentou níveis aproximados, de modo que a taxa média geral de acertos, de 94,77%, sinaliza um coerente funcionamento, em um contexto dependente de locutor, em meio a diversidade de características da fala de diferentes locutores, em uma situação de ruído ambiente natural (não controlada).

6 Conclusão e sugestões para trabalhos futuros

Diante do atual cenário mundial de acelerado envelhecimento populacional (FLANDORFER, 2012; SECRETARIA DE DIREITOS HUMANOS, 2012), a demanda por assistência às pessoas idosas e com mobilidade reduzida quanto aos cuidados com sua saúde e ao auxílio para a execução de tarefas cotidianas tem aumentado consideravelmente. As tecnologias assistivas surgem como um importante recurso para o provimento de maior autonomia a esses indivíduos, representando, assim, uma ampla variedade de aplicações com elevado potencial e interesse por parte da indústria de *healthcare* (PERERA et al., 2014).

Os idosos e as pessoas com mobilidade reduzida usualmente apresentam uma voz estável até mesmo nos casos de tetraplegia e idade muito avançada (QIDWAI; SHAKIR, 2012). A utilização da entrada de comandos por voz para o controle dessas tecnologias se apresenta como uma atrativa solução para que esses indivíduos realizem o controle do computador e outros dispositivos no ambiente doméstico, por exemplo.

Motivado por esse contexto, o trabalho desenvolvido nessa pesquisa apresenta uma interessante alternativa para o desenvolvimento de tecnologias assistivas de baixo custo controladas pela fala. O pequeno tamanho dos componentes utilizados nas soluções apresentadas permite que seja desenvolvido um encapsulamento que possa ser adaptável à fixação em locais como uma cabeceira de cama ou uma cadeira de rodas.

A metodologia adotada ao longo do trabalho, constituída em três fases, permitiu um melhor acompanhamento de suas etapas de desenvolvimento, gerando a cada fase do processo uma documentação consistente sobre os componentes e procedimentos utilizados, as dificuldades encontradas e as soluções desenvolvidas. Com a observação desse preceito espera-se também proporcionar a outros pesquisadores a oportunidade de terem uma via mais rápida de poder contribuir com a evolução da plataforma, ou mesmo para a criação de novas aplicações.

O emprego de tecnologias de *software* de código aberto e *hardware* aberto na concepção das interfaces de reconhecimento de comandos por voz implementadas contribui para reduzir o preço final das aplicações que venham a ser desenvolvidas.

Apesar de ter sido desenvolvida com o intuito de se adquirir uma experiência inicial no âmbito das tecnologias de comandos por voz, a primeira abordagem, baseada em um módulo de *hardware* específico, apresentou resultados que indicaram a confiabilidade necessária para o emprego da plataforma em diversas aplicações.

Embora a solução dessa primeira interface tenha sido concebida para um cenário de aplicação experimental, envolvendo o acendimento das luzes de um LED RGB, seu desempenho em ambientes ruidosos, com uma taxa média global de acertos de 95,9%, indica um alto

potencial de empregabilidade em soluções de tecnologias assistivas.

Aliado a isso, a flexibilidade e potencial de integração proporcionados com a utilização da placa Beaglebone Black, permitem que sua alta capacidade de processamento esteja disponível para aplicações que demandem recursos avançados. Um exemplo seria a associação com outras modalidades de entrada, dentre elas a entrada de vídeo, permitindo o desenvolvimento de aplicações com interfaces multimodais (HUI; MENG, 2014), de modo a aperfeiçoar a experiência dos usuários com mobilidade reduzida.

O vocabulário de comandos de voz, bem como os sinais enviados pela plataforma para o controle do acendimento e desligamento das luzes do LED RGB, podem ser devidamente adaptados com o intuito de se estender a utilização dessa solução para o controle de iluminação do ambiente, controle de aparelhos eletrodomésticos, integrando-os com mecanismos de segurança, bem como dispositivos domésticos de cuidados com a saúde (*Home Health Care*), dentre outros.

Com relação a segunda abordagem, que concentrou o objetivo principal desse projeto, foi desenvolvida uma interface de reconhecimento da fala, livre e de baixo custo, dependente de locutor, devidamente configurada com modelos acústicos específicos para aplicações de comando e controle em domótica. A solução empregada envolveu a utilização de ferramentas de *software* integradas a um servidor de comunicação.

Os modelos acústicos são componentes fundamentais dos sistemas de RAF. No entanto, uma grande dificuldade encontrada esteve relacionada a escassez de modelos disponíveis livremente, ainda mais quando voltados para o propósito de uma aplicação específica. Assim, houve a necessidade de se gerar tais modelos através da aplicação de um processamento estatístico, cuja entrada de dados se constituiu de amostras de áudio da fala de 10 diferentes locutores, associadas a uma lista de palavras foneticamente balanceada e a um dicionário fonético do idioma utilizado.

O processo de gravação das amostras de treinamento se apresentou como um desafio, uma vez que demanda muito tempo. Além disso, trata-se do envolvimento de pessoas que dispuseram-se a doar seu tempo para contribuir com a pesquisa realizando registros de sua voz para a formação da base de dados necessária à criação dos modelos acústicos.

Assim, a submissão do projeto à apreciação do Comitê de Ética em Pesquisa do IFPB foi um importante passo quanto a observação dos padrões éticos em pesquisa envolvendo seres humanos. Pôde-se com isso, assegurar as garantias quanto aos interesses dos participantes da pesquisa, bem como de seus desenvolvedores. O relatório final do projeto junto ao Comitê de Ética em Pesquisa foi aprovado, conforme indica o parecer de nº 1.397.087, disponível no Anexo J.

Apesar de requerer um maior tempo de desenvolvimento, essa segunda interface também apresentou resultados que permitem concluir que o sistema proposto possui elevado potencial de aplicabilidade. O resultado da avaliação do seu desempenho apresentou uma taxa média geral de acertos de 94,77%, quando aplicada num ambiente ruidoso, com nível de

ruído já considerado relativamente elevado, de acordo com os critérios de avaliação do ruído em áreas habitadas observados na norma ABNT NBR 10151 (ABNT, 2000).

Além disso, o tempo dedicado à gravação de amostras de treinamento e realização dos testes com essa plataforma implicam em ganhos a longo prazo, com a construção de uma base de dados cada vez mais sólida e consistente para ser utilizada na aplicação pretendida. Isso permite ainda que se possa chegar à geração de modelos independentes de locutor.

O arquivo contendo o texto referente ao processo de gravação das amostras de treinamento (arquivo *prompts*), juntamente com as próprias amostras obtidas, foram disponibilizadas, de acordo com o consentimento dos locutores, no *site* do projeto Voxforge.

Como acontece com relação à interface baseada em módulo de *hardware*, o emprego da plataforma Beaglebone Black proporciona uma configuração portável e flexível à interface, que pode ser facilmente integrada a outras tecnologias, por ser baseada em recursos de *hardware* aberto e *software* de código aberto. No caso da plataforma baseada em ferramentas de *software* para o RAF associadas ao servidor de comunicação Asterisk, o potencial de integração da plataforma é ampliado consideravelmente, uma vez que amplia as possibilidades de entrada de voz através por diferentes dispositivos através do protocolo da Internet (IP).

Essa característica, enquadra o produto da segunda abordagem no contexto das redes convergentes, compatível com diversos protocolos e as mais variadas tecnologias de comunicação, em uma arquitetura de sistemas embarcados conectada à IoT.

Os dados obtidos com a pesquisa realizada fornecem base para justificar maior investimento na interface desenvolvida, uma vez que a qualidade do seu funcionamento pode ser ainda melhorada a partir de soluções simples, como a observação de um maior cuidado com questões relacionadas somente ao volume das gravações de amostras de treinamento, buscando-se agora empregar o conhecimento produzido para adaptá-la à utilização de comandos no idioma português, tendo sido demonstrada sua viabilidade.

6.1 Contribuições do trabalho

A partir dos resultados obtidos nesse trabalho, pode-se citar as seguintes contribuições:

1. Produção de material teórico e prático de referência, no idioma português, para o desenvolvimento de aplicações de reconhecimento automático da fala.

2. Indicação de uma possível metodologia a ser empregada para o desenvolvimento de aplicações de reconhecimento automático da fala.

3. Implementação de dois protótipos de interfaces de comandos por voz, consistentemente documentados, para serem empregados no desenvolvimento de tecnologias assistivas.

4. Desenvolvimento e livre disponibilização de recursos de base de texto e base de áudio no idioma inglês, com sotaque brasileiro, para a criação de modelos acústicos específicos para aplicações de comando e controle em domótica.

6.2 Sugestões para trabalhos futuros

Conforme exposto nesse trabalho, a agregação de capacidades de reconhecimento automático da fala a dispositivos de tecnologias assistivas se torna iminente para que os indivíduos com limitações quanto a redução de sua mobilidade interajam com tais tecnologias de maneira mais natural e espontânea. Assim, a seguir são apresentadas algumas sugestões para continuidade do trabalho desenvolvido.

1. Adaptar o vocabulário de comandos utilizado na criação do dicionário de pronúncias para trabalhar na geração dos modelos acústicos que implementem o reconhecimento da fala no idioma português. Isso requer a disponibilização de um dicionário de pronúncias no referido idioma, bem como a adaptação dos fonemas para a língua portuguesa.

2. Averiguar o comportamento da interface quanto ao emprego de um modelo acústico geral criado a partir das amostras de treinamento dos 10 locutores, com o intuito de avaliar seu desempenho quanto à independência ao locutor.

3. Realizar um estudo da influência que os fatores de instabilidade da fala originados na fonte glótica podem causar no desempenho do RAF correlacionando-se, por exemplo, essa medida de desempenho com uma avaliação da presença dos referidos fatores na fala a partir de análise de quantificação de recorrência.

4. Realizar um estudo para observar a influência dos parâmetros relacionados ao processamento e análise espectral do sinal de voz no desempenho do sistema e implementar um ajuste automático desses parâmetros de modo que sejam adaptados conforme as características de cada locutor.

5. Implementar otimizações para alcançar melhores taxas de acerto a partir da verificação dos índices de confiança do reconhecimento fornecidos na saída do Julius.

6. Realização de testes com a plataforma a partir de outros dispositivos, como: telefones fixos convencionais, chamada a partir da rede de telefonia móvel, através da Internet, dentre outros.

7. Implementar a interface de reconhecimento automático da fala embarcada utilizando outras tecnologias, como FPGA (*Field Programmable Gate Array*) ou DSP (*Digital Signal Processor*) e realizar um estudo comparativo com os resultados apresentados neste trabalho.

8. Implementar a entrada de áudio sensível ao tempo de pronúncia característico de cada usuário.

9. Implementar o ajuste automático da intensidade vocal para a aquisição das amostras de treinamento e dos dados de áudio na entrada dos comandos de voz.

10. Implementar questões relacionadas à segurança do sistema, evitando que os dispositivos sejam acionados por outras vozes presentes no ruído de fundo do ambiente, ou por indivíduos estranhos, que não o usuário. Implementar autenticação do usuário a partir da fala.

11. Realizar um estudo para avaliação do desempenho da rede WiFi IEEE 802.11 na transmissão dos comandos de voz e sua influência no desempenho do RAF. Verificar a influência relacionada ao alcance da rede sem fio.

12. Implementar a comunicação dos comandos de voz com a plataforma, bem como o envio dos sinais de controle para os dispositivos domésticos, utilizando redes com o padrão IEEE 802.15.4. E verificar o desempenho dessas redes em comparação com as redes WiFi IEEE 802.11.

Referências bibliográficas

ABNT. *Acústica - Avaliação do ruído em áreas habitadas, visando o conforto da comunidade*. [S.l.], 2000. Citado na página 125.

ADAFRUIT. *Setting up IO Python Library on BeagleBone Black*. 2013. Disponível em: <https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/overview>. Acesso em: 21 abr. 2014. Citado na página 81.

ADAFRUIT. *SSH to BeagleBone Black over USB*. 2013. Disponível em: <https://learn.adafruit.com/ssh-to-beaglebone-black-over-usb/overview>. Acesso em: 24 fev. 2015. Citado na página 147.

AGUILERA, F. et al. A Personal Computer Based Environmental Control System For The Disabled. In: INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY, 14., 1992, França. *Proceedings...* Paris: IEEE, 1992. Citado 3 vezes nas páginas 45, 56 e 112.

ALIEXPRESS. *AliExpress*. Disponível em: <http://pt.aliexpress.com/>. Acesso em: 03 jan. 2016. Citado na página 106.

ALSHU'EILI, H.; GUPTA, S. M. G. S. Voice Recognition Based Wireless Home Automation System. In: INTERNATIONAL CONFERENCE ON MECHATRONICS, 4., 2011, Malaysia. *Proceedings...* Kuala Lumpur: IEEE, 2011. Citado 3 vezes nas páginas 37, 51 e 57.

AON2. *About Us*. Disponível em: <http://www.aon2.co.uk/about>. Acesso em: 18 jun. 2015. Citado na página 94.

AONSQUARED BLOG. *Raspberry Pi Speech Recognition*. 2015. Disponível em: <http://www.aonsquared.co.uk/node/9>. Acesso em: 18 jun. 2015. Citado na página 94.

APACHE Open Office. Disponível em: <http://www.openoffice.org/pt-br/>. Acesso em: 12 jun. 2015. Citado na página 50.

ARDUINO. *Arduino*. Disponível em: <https://www.arduino.cc/>. Acesso em: 03 jan. 2016. Citado na página 106.

ASAHI KASEI. *VORERO*. 2000. Disponível em: <http://www.asahi-kasei.co.jp/asahi/en/services_products/other/>. Citado 2 vezes nas páginas 46 e 56.

ASTERISK DEVELOPMENT TEAM. *Asterisk Administrator Guide*. 2013. Disponível em: <https://wiki.asterisk.org/wiki/display/AST/Asterisk+Exported+Documentation>. Acesso em: 26 jun. 2015. Citado na página 98.

ATMEGA8. Disponível em: <http://www.atmel.com/devices/atmega8.aspx>. Acesso em: 12 jun. 2015. Citado na página 51.

AUDACITY. Disponível em: <http://audacityteam.org/>. Acesso em: 04 jul. 2015. Citado na página 96.

AYLOR, J. et al. Versatile Wheelchair Control System. *Medical and Biologycal Engineering and Computing*, v. 17, n. 1, p. 110–114, 1979. Citado na página 40.

BAINBRIDGE ISLAND REVIEW. *Quadriplegic Todd Stabelfelt wants to help others overcome their disabilities through technology*: A model for independent living. Bainbridge Island, 2008. Disponível em: <http://www.bainbridgereview.com/news/18288499.html>. Acesso em: 27 mar. 2014. Citado 2 vezes nas páginas 32 e 36.

BATISTA, P. dos S. *Avanços em Reconhecimento de Fala para Português Brasileiro e Aplicações*: Ditado no libreoffice e unidade de resposta audível com asterisk. 95 f. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal do Pará, Belém, 2013. Citado 15 vezes nas páginas 31, 37, 52, 57, 59, 60, 70, 72, 90, 92, 95, 96, 99, 119 e 120.

BEAGLEBOARD. *Debian*. Disponível em: <http://beagleboard.org/project/debian/>. Acesso em: 21 jan. 2016. Citado na página 81.

BEAGLEBOARD. *Latest Firmware Images*. 2015. Disponível em: <http://beagleboard.org/latest-images>. Acesso em: 24 out. 2015. Citado na página 145.

BEAGLEBONE Black. 2015. Disponível em: <http://beagleboard.org/black>. Acesso em: 12 jun. 2015. Citado 3 vezes nas páginas 52, 74 e 78.

BENTHIN, F. Linux Controlado por Voz. *Tutorial*, Revista Linux Magazine, n. 101, p. 56–61, Abr. 2013. Citado 2 vezes nas páginas 37 e 52.

BETTELHEIM, R.; STEELE, D. *Speech and Command Recognition*. Freescale, 2010. Disponível em: <http://www.arcturusnetworks.com/VoiceCtrl_WP.pdf>. Acesso em: 08 jun. 2015. Citado na página 44.

BRASIL. *Tecnologias Assistivas*. Secretaria Nacional de Direitos Humanos da Presidência da República (SDH/PR) – Comitê de Ajudas Técnicas (CTA), 2009. Disponível em: <http://www.pessoacomdeficiencia.gov.br/app/sites/default/files/publicacoes/livro-tecnologia-assistiva.pdf>. Acesso em: 26 mar. 2014. Citado 2 vezes nas páginas 32 e 35.

BRASIL. *Viver sem Limite*: Um plano para todo o brasil. Secretaria Nacional de Promoção dos Direitos da Pessoa com Deficiência (SNPD), 2011. Disponível em: <http://www.pessoacomdeficiencia.gov.br/app/viver-sem-limite-0>. Acesso em: 26 mar. 2014. Citado na página 35.

BRASIL. *Centro Nacional de Referência em Tecnologia Assistiva (CNRT)*. 2012. Disponível em: <http://www.cti.gov.br/cnrta>. Acesso em: 27 mar. 2014. Citado na página 35.

BRASIL. *Catálogo Nacional de Produtos de Tecnologia Assistiva*. Ministério da Ciência e Tecnologia, 2014. Disponível em: <http://assistiva.mct.gov.br/>. Acesso em: 27 mar. 2014. Citado na página 35.

BROADBENT, E.; STAFFORD, R.; MACDONALD, B. Acceptance of healthcare robots for the older population: review and future directions. *International Journal of Social Robotics*, v. 1, p. 319–330, 2009. Citado na página 34.

BURDA, R.; WIETFELD, C. Multimedia over 802.15.4 and ZigBee Networks for Ambient Environment Control. In: VEHICULAR TECHNOLOGY CONFERENCE, 65., 2007, Irlanda. *Proceedings...* Dublin: IEEE, 2007. Citado na página 51.

CAMBRIDGE UNIVERSITY ENGINEERING DEPARTMENT. *HTK FAQ*. Cambridge, 2015. Disponível em: <http://htk.eng.cam.ac.uk/docs/faq.shtml>. Citado na página 166.

CHANDRAMOULI, C.; AGARWAL, V. Speech Recognition based Computer Keyboard Replacement for the Quadriplegics, Paraplegics, Paralytics and Amputees. In: INTERNATIONAL WORKSHOP ON MEDICAL MEASUREMENTS AND APPLICATIONS, 2009, Cetraro. *Proceedings...* Cetraro: IEEE, 2009. p. 241–245. Citado na página 36.

CMU ROBUST GROUP. *Tutorial*: Learning to use the CMU SPHINX Automatic Speech Recognition System. 2008. Disponível em: <http://www.speech.cs.cmu.edu/sphinx/tutorial.html>. Acesso em: 17 jan. 2016. Citado na página 73.

CMU SPHINX. *Tutorial*: Building application with pocketsphinx. 2015. Disponível em: <http://cmusphinx.sourceforge.net/wiki/tutorialpocketsphinx>. Acesso em: 18 jan. 2016. Citado 3 vezes nas páginas 73, 74 e 90.

CMU SPHINX. *Tutorial*: Training Acoustic Model For CMUSphinx. 2015. Disponível em: <http://cmusphinx.sourceforge.net/wiki/tutorialam>. Acesso em: 18 jun. 2015. Citado 2 vezes nas páginas 73 e 96.

CMU SPHINX FAQ. *Frequently Asked Questions*. 2016. Disponível em: <http://cmusphinx.sourceforge.net/wiki/faq>. Acesso em: 19 jan. 2016. Citado na página 74.

COETZEE, L.; EKSTEEN, J. The Internet of Things – Promise for the Future? An Introduction. In: IST-AFRICA CONFERENCE, 2011, Botswana. *Proceedings...* Gaborone: IEEE, 2011. Citado 2 vezes nas páginas 88 e 98.

COOPER, J. *GitHub Adafruit Forum*: GPIO access as user even with rigth permissions to */sys* doesnt́ work. 2013. Disponível em: <https://github.com/adafruit/adafruit-beaglebone-io-python/issues/36>. Acesso em: 31 jan. 2016. Citado na página 101.

COOPER, J. *Setting up IO Python Library on BeagleBone Black*. 2015. Disponível em: <https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/installation>. Acesso em: 02 jul. 2015. Citado na página 83.

CORNELL, J. et al. *Comprehensive scoping study on the use of assistive technology by frail older people living in the community*. [S.l.], 2008. 92 p. Citado 2 vezes nas páginas 32 e 34.

DAVENPORT, M. *The Asterisk Speech Recognition API*. 2012. Disponível em: <https://wiki.asterisk.org/wiki/display/AST/Speech+Recognition+API>. Acesso em: 17 jun. 2015. Citado na página 91.

DELLER JR, J. R.; HANSEN, J. H. L.; PROAKIS, J. G. *Discrete-Time Processing of Speech Signals*. New York: Macmillan, 1993. 918 p. ISBN 0-7803-5386-2. Citado 5 vezes nas páginas 63, 64, 65, 66 e 67.

DESENVOLVIMENTO ÁGIL. *Scrum*. 2014. Disponível em: <http://www.desenvolvimentoagil.com.br/scrum/>. Acesso em: 16 jan. 2016. Citado na página 77.

DIAS, M. C.; LUCENA, D. C.; SANTOS, E. P. O Uso do Asterisk para o Controle Remoto de Sistemas de Automação. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, XX., 2014, MG. *Anais...* Belo horizonte: Universidade Federal de Minas Gerais, 2014. Citado 4 vezes nas páginas 31, 37, 43 e 98.

DIGIUM. *Asterisk*. Disponível em: <http://www.asterisk.org/>. Acesso em: 27 jan. 2016. Citado na página 89.

DOHR, A. et al. The Internet of Things for Ambient Assisted Living. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY, 7., 2010, Estados Unidos. *Proceedings...* Las Vegas: IEEE, 2010. Citado 2 vezes nas páginas 31 e 37.

DSP Starter Kit (DSK) for TMS320C6711. Disponível em: <http://www.kanecomputing.co.uk/dsk_c6711.htm>. Acesso em: 12 jun. 2015. Citado na página 48.

EASYPRONUNCIATION.COM. *English Phonetic Transcription Converter*. Disponível em: <http://easypronunciation.com/en/english-phonetic-transcription-converter#result>. Acesso em: 05 jan. 2016. Citado na página 120.

EASYVR 3. Disponível em: <http://www.veear.eu/products/easyvr3/>. Acesso em: 12 jun. 2015. Citado na página 51.

ENGINEERING AND TECHNOLOGY HISTORY WIKI. *First-Hand: The Hidden Markov Model*. 2015. Disponível em: <http://ethw.org/First-Hand:The_Hidden_Markov_Model>. Acesso em: 09 mar. 2015. Citado na página 44.

EUROPEAN COMISSION - INFORMATION SOCIETY AND MEDIA. *AsTeRICS Deliverable D2.4*: Report on state-of-the-art. [S.l.], 2010. 182 p. Citado 6 vezes nas páginas 34, 36, 39, 40, 41 e 43.

FALABRASIL: Reconhecimento de Voz para o Português Brasileiro. Disponível em: <http://www.laps.ufpa.br/falabrasil/>. Acesso em: 12 jun. 2015. Citado 2 vezes nas páginas 50 e 95.

FECHINE, J. M. *Reconhecimento Automático de Identidade Vocal Utilizando Modelagem Híbrida: Paramétrica e Estatística*. 237 f. Tese (Doutorado em Ciências) — Universidade Federal de Campina Grande, Campina Grande, 2000. Citado na página 120.

FLANDORFER, P. Population Ageing and Socially Assistive Robots for Elderly Persons: The Importance of Sociodemographic Factors for User Acceptance. *International Journal of Population Research*, v. 2012, p. 14, Fev. 2012. Citado 4 vezes nas páginas 32, 34, 39 e 123.

FREITAS, S. V. de. *Correlação entre a avaliação acústica e perceptual na caracterização de vozes patológicas*: Relatório do Estado da Arte. [S.l.], 2010. 142 f. Citado na página 120.

FURUI, S. *Digital Speech Processing, Syntesis and Recognition*. 2. ed. [S.l.]: CRC Press, 1993. 476 p. ISBN 978-0824704520. Citado 3 vezes nas páginas 64, 65 e 67.

GALAN, F. et al. A brain-actuated wheelchair: Asynchronous and non-invasive Brain-computer interfaces for continuous control of robots. *Clinical Neurophysiology*, v. 119, n. 9, p. 2159–2169, Set. 2008. Citado na página 42.

GIGADNS. *GigaDNS*. Disponível em: <http://gigadns.web1407.kinghost.net//>. Acesso em: 24 jan. 2016. Citado na página 149.

GITHUB. *RobotControl*: 80-gpio.rules. 2014. Disponível em: <https://github.com/cnobile2012/RobotControl/tree/master/contrib>. Acesso em: 31 jan. 2016. Citado na página 161.

GOMÉZ, E. M. T. *Reconhecimento de fala para navegação em aplicativos móveis para português brasileiro*. 126 f. Dissertação (Mestrado em Ciência da Computação) — Universidade de São Paulo, São Paulo, 2011. Citado 4 vezes nas páginas 59, 61, 73 e 95.

GOOGLE. *Android*. Disponível em: <https://www.android.com/>. Acesso em: 18 jun. 2015. Citado 2 vezes nas páginas 87 e 88.

GOOGLE. *Google Now*. 2016. Disponível em: <https://www.google.com/landing/now/>. Acesso em: 05 jan. 2016. Citado 2 vezes nas páginas 53 e 57.

GOOGLE. *Natural Language Processing Group*. 2016. Disponível em: <http://research.google.com/pubs/NaturalLanguageProcessing.html>. Acesso em: 05 jan. 2016. Citado na página 53.

GOOGLE. *Speech Processing Group*. 2016. Disponível em: <http://research.google.com/pubs/SpeechProcessing.html>. Acesso em: 05 jan. 2016. Citado na página 53.

GOOGLE GROUPS. *UniMRCP Discussion Group*: How to integrate UNIMRCP with Julius? Disponível em: <https://groups.google.com/forum/#!searchin/unimrcp/julius/unimrcp/ekb8VwfGd1U/h3TWST9sfnAJ>. Acesso em: 30 jan. 2016. Citado na página 92.

GOOGLE PLAY STORE. *Decibelímetro*: Sound Meter. Disponível em: <https://play.google.com/store/apps/details?id=kr.sira.sound&hl=pt_BR>. Acesso em: 25 jan. 2016. Citado na página 87.

GRIMMETT, R. *Mastering Beaglebone Robotics*. Birmingham: Packt Publishing Ltd., 2014. 234 p. ISBN 978-1-78398-890-7. Citado 3 vezes nas páginas 73, 74 e 90.

HALL, B.; MOLLOY, J. Installing a Voice Activated Environmental Control Unit for Under 500 Dollars. In: INTERNATIONAL ANUAL CONFERENCE OF REHABILITATION ENGINEERING AND ASSISTIVE TECHNOLOGY SOCIETY OF NORTH AMERICA, 26., 2003, Geórgia. Atlanta: RESNA, 2003. Citado 3 vezes nas páginas 46, 56 e 108.

HAYASHI, T.; KAWAMOTO, H.; SANKAI, Y. Control method of robot suit HAL working as operator's muscle using biological and dynamical information. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 2005, Canada. *Proceedings...* Alberta: IEEE, 2005. Citado na página 42.

HINTON, G. et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Processing Magazine*, p. 82–97, Nov. 2012. Citado na página 70.

HTK3. *Registration*. 2015. Disponível em: <http://htk.eng.cam.ac.uk/register.shtml>. Acesso em: 22 jan. 2015. Citado na página 61.

HUANG, X. et al. An overview of the SPHINX-II speech recognition system. In: WORKSHOP ON HUMAN LANGUAGE TECHNOLOGY, 1993, Estados Unidos. *Proceedings...* Morristown, NJ, 1993. Citado na página 73.

HUGGINS-DAINES, D. et al. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In: INTERNAITONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, 2006, Estados Unidos. *Proceedings...* Pittsburgh, PA: IEEE, 2006. Citado na página 73.

HUI, P.; MENG, H. Latent Semantic Analisys for Multimodal User Input With Speech and Gestures. *IEEE/ACM Transactions On Audio, Speech and Language Processing*, v. 22, n. 2, p. 417–429, Feb. 2014. Citado 2 vezes nas páginas 112 e 124.

IBGE. *Censo demográfico 2010*: Resultados gerais da amostra. Instituto Brasileiro de Geografia e Estatística, 2012. Disponível em: <http://www.ibge.gov.br/home/presidencia/noticias/imprensa/ppts/000000847310412201231572748 3985.pdf>. Acesso em: 29 mai. 2015. Citado na página 32.

IEEE 802.11^{TM} Wireless Local Area Networks. Disponível em: <http://www.ieee802.org/11/>. Acesso em: 18 jun. 2015. Citado na página 89.

IETF. *SIP*: Session initiation protocol. 2002. Disponível em: <https://www.ietf.org/rfc/rfc3261.txt>. Acesso em: 31 jan. 2016. Citado na página 99.

INTELBRÁS. *Guia de instalação*: WBN900. Disponível em: <http://www.intelbras.com.br/sites/default/files/downloads/guia_de_instalacao_-_adaptador_usb_wireless_n_150_mbps_wbn_900.pdf>. Acesso em: 29 jan. 2016. Citado na página 157.

INTERBRAS. *Adaptador USB Wireless N 150 Mbps*. Disponível em: <http://www.intelbras.com.br/residencial/wireless/adaptadores-usb/wbn-900>. Acesso em: 18 jun. 2015. Citado na página 89.

JIANG, H. et al. Voice-Activated Environmental Control System for Persons with Disabilities. In: ANUAL NORTHEAST BIOENGINEERING CONFERENCE, 26., 2000, Connecticut. *Proceedings...* Storrs: IEEE, 2000. Citado 5 vezes nas páginas 45, 46, 56, 108 e 112.

JULIUS: Open-source large vocabulary CSR engine. 2014. Disponível em: <http://julius.osdn.jp/en_index.php?q=index-en.html>. Citado 4 vezes nas páginas 45, 69, 70 e 151.

JULIUS: Open-source large vocabulary CSR engine. 2015. Disponível em: <https://github.com/julius-speech/julius>. Citado na página 70.

JULIUS DEVELOPMENT TEAM. *How to write a recognition grammar for Julius*. Disponível em: <http://julius.osdn.jp/en_index.php?q=en_grammar.html>. Acesso em: 17 jul. 2015. Citado na página 171.

KELLER, A. *Asterisk na prática*. São Paulo: Novatec, 2009. 333 p. ISBN 978-85-7522-183-9. Citado na página 98.

LABORATÓRIO DE PROCESSAMENTO DE SINAIS. *Download de softwares*. Disponível em: <http://www.laps.ufpa.br/falabrasil/downloads.php>. Acesso em: 18 jun. 2015. Citado na página 49.

LEE, A. *The Julius Book*. [S.l.: s.n.], 2010. 67 p. Citado 9 vezes nas páginas 45, 60, 70, 71, 72, 90, 100, 151 e 153.

LEE, A.; KAWAHARA, T. Recent Development of Open-Source Speech Recognition Engine Julius. In: 2009 ANNUAL SUMMIT AND CONFERENCE OF ASIA-PACIFIC SIGNAL AND INFORMATION PROCESSING ASSOCIATION, 2009, Japão. *Proceedings...* Hokkaido: Hokkaido University, 2009. p. 131–137. Citado 3 vezes nas páginas 70, 71 e 72.

LEGO Mindstorms NTX 2.0. Disponível em: <http://shop.lego.com/en-US/LEGO-MINDSTORMS-NXT-2-0-8547>. Acesso em: 12 jun. 2015. Citado na página 51.

LENZO, K. *The CMU Pronouncing Dictionary*. Carnegie Mellon University. Disponível em: <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>. Acesso em: 17 jul. 2015. Citado na página 170.

LI, J. et al. An Overview of Noise-Robust Automatic Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, v. 22, n. 4, p. 745–777, Abr. 2014. Citado na página 119.

LU, Y.; CHEN, Y. Prototyping Potential Control Systems to Assist Complete Quadriplegics. In: BIOMEDICAL ENGINEERING INTERNATIONAL CONFERENCE, 2012, Tailândia. *Proceedings...* Ubon Ratchathani: IEEE, 2012. Citado 4 vezes nas páginas 40, 42, 51 e 112.

LUMENVOX. *Pizza Demo Application*. Disponível em: <http://www.lumenvox.com/partners/digium/applicationzone/projects/originalPizza.aspx>. Acesso em: 30 jan. 2016. Citado na página 91.

MADRIGAL, A. *Tongue Drive System Controls Wheelchair*. 2008. Disponível em: <http://www.wired.com/2008/07/tongue-drive-sy/>. Acesso em: 03 jun. 2015. Citado na página 40.

MADSEN, L.; MEGGELEN, J. V.; BRYANT, R. *Asterisk^{TM}: The Definitive Guide*. 3. ed. Sebastopol: O'Reilly Media, Inc., 2011. 734 p. ISBN 978-0-569-51734-2. Citado 5 vezes nas páginas 37, 92, 98, 99 e 155.

MATHWORKS. *Configure Network Connection with BeagleBoard Hardware*. 2015. Disponível em: <http://www.mathworks.com/help/supportpkg/beagleboard/ug/getting-the-beagleboard-ip-address.html>. Acesso em: 24 jan. 2016. Citado na página 148.

MC68HC11 Family Datasheet. Disponível em: <http://www.freescale.com/files/microcontrollers/doc/data_sheet/M68HC11E.pdf>. Acesso em: 12 jun. 2015. Citado na página 46.

MICROSOFT. *Configurar reconhecimento de fala*. 2015. Disponível em: <http://windows.microsoft.com/pt-br/windows/set-speech-recognition#1TC=windows-7>. Acesso em: 03 jun. 2015. Citado na página 43.

MICROSOFT Office Power Point. Disponível em: <https://products.office.com/pt-br/powerpoint>. Acesso em: 12 jun. 2015. Citado na página 50.

MIORI, V.; RUSSO, D. Anticipating health hazards through an ontology-based, IoT domotic environment. In: INTERNAITONAL CONFERENCE ON INNOVATIVE MOBILE AND INTERNET SERVICES IN UBIQUITOUS COMPUTING, 6., 2012, Itália. *Proceedings...* Palermo: IEEE, 2012. Citado 2 vezes nas páginas 34 e 37.

NATURAL POINT. *SmartNav*. 2015. Disponível em: <http://www.naturalpoint.com/smartnav/>. Acesso em: 03 jun. 2015. Citado na página 41.

NELSON NETO et al. *Um Reconhecedor de Voz Livre para Português Brasileiro com Interface de Programação*. Laboratório de Processamento de Sinais, 2010. Disponível em: <http://www.laps.ufpa.br/falabrasil/files/coruja-doc-pt.pdf>. Acesso em: 11 jun. 2015. Citado 3 vezes nas páginas 49, 56 e 95.

NELSON NETO et al. Free tools and resources for Brazilian Portuguese speech recognition. *Journal of the Brazilian Computer Society (Online)*, v. 17, n. 1, p. 53–68, Mar. 2011. Citado 6 vezes nas páginas 50, 56, 66, 67, 68 e 95.

NTP.BR. *NTP.br*. Disponível em: <http://ntp.br/>. Acesso em: 24 jan. 2016. Citado na página 150.

NUANCE COMMUNICATIONS. *Dragon Speech Recognition Software*. Disponível em: <http://www.nuance.co.uk/dragon/index.htm>. Citado na página 46.

OBER CONSULTING. *Blink-It*. 2015. Disponível em: <http://www.ober-consulting.com/product/blink-it/>. Acesso em: 03 jun. 2015. Citado na página 41.

OLIVEIRA, R. et al. Recursos para Desenvolvimento de Aplicativos com Suporte a Reconhecimento de Voz para Desktop e Sistemas Embarcados. In: WORKSHOP DE SOFTWARE LIVRE, 2011, Brasil. *Proceedings...* Porto Alegre, 2011. Disponível em: <http://wsl.softwarelivre.org/2011/0016/85180_1.pdf>. Acesso em: 11 jun. 2015. Citado 3 vezes nas páginas 50, 56 e 95.

OPEN SOURCE INITIATIVE. Disponível em: <http://opensource.org/>. Acesso em: 04 jul. 2015. Citado na página 81.

ORIGIN INSTRUMENTS. *HeadMouse® Extreme*. 2015. Disponível em: <http://www.naturalpoint.com/smartnav/>. Acesso em: 03 jun. 2015. Citado na página 41.

PELLEGRINI, N. et al. Optimization of power wheelchair control for patients with severe Duchenne muscular dystrophy. *Neuromuscular disorders*, Elsevier, v. 14, n. 5, p. 297–300, 2004. Citado na página 40.

PERERA, C. et al. A Survey on Internet of Things From Industrial Market Perspective. *The Journal for rapid open access publishing*, v. 2, p. 1660–1679, Dec. 2014. Citado 5 vezes nas páginas 31, 34, 37, 98 e 123.

PHILIPS, J. et al. Adaptive Shared Control of a Brain-Actuated Simulated Wheelchair. In: INTERNATIONAL CONFERENCE ON REHABILITATION ROBOTICS, 10., 2007, Netherlands. *Proceedings...* Noordwijk: IEEE, 2007. Citado na página 42.

PIC16F876A. Disponível em: <http://www.microchip.com/wwwproducts/Devices.aspx?product=PIC16F876A>. Acesso em: 12 jun. 2015. Citado na página 49.

PIMENTEL, V. C. A. *Reconhecimento de Comandos por Voz*. 2014. Disponível em: <https://www.youtube.com/watch?v=53tq-M9TslQ>. Acesso em: 22 jan. 2016. Citado na página 86.

PIMENTEL, V. C. A. et al. Uma Plataforma de Baixo Custo Comandada por Voz para Tecnologias Assistivas com Programação em Python. In: CONGRESSO BRASILEIRO DE ENGENHARIA BIOMÉDICA, XXIV., 2014, MG. *Anais...* Uberlândia: Universidade Federal de Uberlândia, 2014. Citado 3 vezes nas páginas 37, 52 e 82.

PLAZA, M.; AVILÉS, O.; APERADOR, W. Technology in Locomotion and Domotic Control for Quadriplegic. In: SOUTHERN BIOMEDICAL ENGINEERING CONFERENCE, 29., 2013, Estados Unidos. *Proceedings...* Florida: IEEE, 2013. Citado na página 40.

POENARU, E.; POENARU, C. A Structured Approach of the Internet-of-Things eHealth Use Cases. In: INTERNAITONAL CONFERENCE ON E-HEALTH AND BIOENGINEERING, 4., 2013, Romênia. *Proceedings...* Laśi: IEEE, 2013. Citado 2 vezes nas páginas 34 e 37.

PRABHAKAR, O. P.; SAHU, N. K. A Survey On: Voice Command Recognition Technique. *International Journal of Advanced Research in Computer Science and Software Engineering*, v. 3, n. 5, p. 576–585, Mai. 2013. Citado 2 vezes nas páginas 36 e 44.

PROAKIS, J. G.; MANOLAKIS, D. G. *Digital Signal Processing*: Principles, Algorithms and Applications. 3. ed. New Jersey: Prentice Hall International, 1996. 1033 p. ISBN 0-13-373762-4. Citado 3 vezes nas páginas 62, 63 e 64.

PRPIPG-IFPB. *Comitê de Ética em Pesquisa*. 2016. Disponível em: <http://www.ifpb.edu.br/reitoria/pro-reitorias/prpipg/comite-de-etica-em-pesquisa>. Acesso em: 31 jan. 2016. Citado na página 103.

PYTHON. Disponível em: <https://www.python.org>. Acesso em: 12 jun. 2015. Citado na página 52.

QADRI, M. T.; AHMED, S. A. Voice Controlled Wheelchair Using DSK TMS320C6711. In: INTERNATIONAL CONFERENCE ON SIGNAL ACQUISITION AND PROCESSING, 2009, Malásia. *Proceedings...* Kuala Lumpur: IEEE, 2009. Citado 2 vezes nas páginas 48 e 56.

QIDWAI, U.; SHAKIR, M. Ubiquitous Arabic Voice Control Device To Assist People With Disabilities. In: INTERNATIONAL CONFERENCE ON INTELLIGENT AND ADVANCED SYSTEMS, 4., 2012, Kuala Lumpur. *Proceedings...* Kuala Lumpur: IEEE, 2012. p. 333–338. Citado 5 vezes nas páginas 36, 51, 57, 112 e 123.

RABINER, L. R.; SCHAFER, R. W. *Digital Processing of Speech Signals*. [S.l.]: Prentice Hall, 1978. 265 p. ISBN 0-13-213603-1. Citado 3 vezes nas páginas 62, 63 e 64.

RATIONAL Unified Process. 2003. Disponível em: <http://www.wthreex.com/rup/portugues/index.htm>. Acesso em: 15 jan. 2016. Citado na página 77.

RESCH, B. *Automatic Speech Recognition with HTK*: A Tutorial for the Course Computational Intelligence. Signal Processing and Speech Communication Laboratory, 2011. Disponível em: <https://www.spsc.tugraz.at/system/files/asr.pdf>. Acesso em: 16 jul. 2015. Citado 2 vezes nas páginas 68 e 69.

RICHARDSON, M. *Getting Started With BeagleBone*. 1. ed. Sebastopol: Maker Media, Inc., 2014. 143 p. ISBN 978-1-449-34537-2. Citado na página 79.

SAHA, D.; MUKHERJEE, A. Pervasive computing: a paradigm for the 21st century. *Computer Magazine*, v. 36, n. 3, p. 25–31, Mar. 2003. Citado na página 88.

SAMPAIO NETO, N. C. *Desenvolvimento de Aplicativos Usando Reconhecimento e Síntese de Voz*. 97 f. Dissertação (Mestrado em Ciências) — Universidade Federal do Pará, Belém, 2006. Citado 3 vezes nas páginas 47, 56 e 95.

SANTOS, F. et al. *ipProcess*: A Usage of an IP-core Development Process to Achieve Time-to-Market and Quality Assurance in a Multi Project Environment. Disponível em: <http://www.design-reuse.com/articles/21746/ip-core-development-process.html>. Acesso em: 16 jan. 2016. Citado na página 77.

SANTOS JÚNIOR, G. G. dos. *Redução de Ruído para Sistemas de Reconhecimento de Voz Utilizando Subespaços Vetoriais*. 88 f. Dissertação (Mestrado) — Universidade Federal de Campina Grande, Campina Grande, 2009. Citado 5 vezes nas páginas 48, 49, 56, 95 e 119.

SARTORETTO, M. L.; BERSCH, R. *Tecnologia Assistiva*. Assistiva®•Tecnologia e Educação, 2014. Disponível em: <http://www.assistiva.com.br/tassistiva.html>. Acesso em: 26 mar. 2014. Citado 3 vezes nas páginas 34, 35 e 36.

SCHULLER, B. et al. Recognition of Noisy Speech: A Comparative Survey of Robust Model Architecture and Feature Enhancement. *EURASIP Journal on Audio, Speech and Music Processing*, p. 17, 2009. Citado na página 119.

SEARA, I. C.; NUNES, V. G.; LAZZAROTTO, C. *Fonética e Fonologia do Português Brasileiro*. 2. ed. Florianópolis: Universidade Federal de Santa Catarina/LLV/CCE/UFSC, 2011. 119 p. ISBN 978-85-61482-38-1. Citado 2 vezes nas páginas 120 e 121.

SEBESTYEN, G. et al. eHealth Solutions in the Context of Internet of Things. In: INTERNAITONAL CONFERENCE ON AUTOMATION, QUALITY AND TESTING, ROBOTICS, 19., 2014, Romênia. *Proceedings...* Cluj-Napoca: IEEE, 2014. Citado 3 vezes nas páginas 31, 34 e 37.

SECRETARIA DE DIREITOS HUMANOS. *Dados sobre o envelhecimento no Brasil*. Coordenação Geral dos Direitos do Idoso, 2012. Disponível em: <http://www.sdh.gov.br/assuntos/pessoa-idosa/dados-estatisticos/DadossobreoenvelhecimentonoBrasil.pdf>. Acesso em: 29 mai. 2015. Citado 3 vezes nas páginas 32, 39 e 123.

SHARMA, U.; MAHESHKAR, S.; MISHRA, A. N. Study of Robust Feature Extraction Techniques for Speech Recognition System. In: INTERNATIONAL CONFERENCE ON FUTURISTIC TREND IN COMPUTATIONAL ANALYSIS AND KNOWLEDGE MANAGEMENT, 1., 2015, India. *Proceedings...* Noida: IEEE, 2015. p. 654–658. Citado na página 119.

SHEN, W. *Voice Recognition Module V2*: Manual. [S.l.], 2013. Disponível em: <http://www.elechouse.com/elechouse/images/product/Voice%20Recognition%20Module/Manual.pdf>. Acesso em: 03 jul. 2015. Citado 4 vezes nas páginas 82, 83, 84 e 85.

SHENZHEN KEYES ROBOT. *LED RGB Module*. 2014. Disponível em: <http://en.keyes-robot.com/>. Acesso em: 23 jan. 2016. Citado na página 86.

SHMYREV, N. *Decoders and Features*. 2011. Disponível em: <http://nshmyrev.blogspot.ru/2011/07/decoders-and-features.html>. Acesso em: 19 jan. 2016. Citado na página 74.

SHMYREV, N. *Is HTK better than CMU Sphinx regarding speech recognition, especially for non-English languages (Arabic, local dialects)? Why?* 2014. Disponível em: <https://www.quora.com/>. Acesso em: 19 jan. 2016. Citado na página 74.

SIGROK. *Embedded*. Disponível em: <https://sigrok.org/wiki/Embedded>. Acesso em: 30 jan. 2016. Citado na página 100.

SILVA, D. D. C. da. *Desenvolvimento de um IP Core de Pré-Processamento Digital de Sinais de Voz para Aplicações em Sistemas Embutidos*. 108 f. Dissertação (Mestrado em Informática) — Universidade Federal de Campina Grande, Campina Grande, 2006. Citado na página 50.

SILVA, D. D. C. da. *Reconhecimento de Fala Contínua para o Português Brasileiro em Sistemas Embarcados*. 194 f. Tese (Doutorado em Ciências) — Universidade Federal de Campina Grande, Campina Grande, 2011. Citado 5 vezes nas páginas 31, 50, 56, 59 e 95.

SILVA, D. D. e. *Contribuições ao reconhecimento automático de fala robusto*. 264 f. Tese (Doutorado em Engenharia de Automação e Sistemas) — Universidade Federal de Santa Catarina, Florianópolis, 2010. Citado 3 vezes nas páginas 61, 66 e 67.

SILVA, P.; NELSON NETO; KLAUTAU, A. Novos Recursos e Utilização de Adaptação de Locutor no Desenvolvimento de um Sistema de Reconhecimento de Voz para o Português Brasileiro. In: SIMPóSIO BRASILEIRO DE TELECOMUNICAçõES, XXVII., 2009, Santa Catarina. *Proceedings...* Blumenau, 2009. Citado 6 vezes nas páginas 49, 56, 68, 69, 95 e 120.

SIMON LISTENS. *Simon listens*: non-profit organization for research and apprenticeship. 2012. Disponível em: <http://simon-listens.org/index.php?id=122&L=1>. Citado 2 vezes nas páginas 52 e 57.

SIMONBR. Disponível em: <http://www.laps.ufpa.br/falabrasil/simonbr.php>. Acesso em: 12 jun. 2015. Citado na página 52.

SPAANS, M. A. *On Developing Acoustic Models Using HTK*. 113 f. Dissertação (*Master of Science*) — Delft University of Technology, Delft, 2004. Citado 12 vezes nas páginas 43, 44, 45, 47, 55, 59, 60, 62, 63, 64, 65 e 68.

SPEECH Recognition System HM2007. Disponível em: <http://www.sunrom.com/p/speech-recognition-system-hm2007>. Acesso em: 12 jun. 2015. Citado na página 45.

SPEECHOO. Disponível em: <http://www.laps.ufpa.br/falabrasil/speechoo.php>. Acesso em: 12 jun. 2015. Citado na página 52.

SSH to Beaglebone Black over USB. Disponível em: <https://learn.adafruit.com/ssh-to-beaglebone-black-over-usb/overview>. Acesso em: 03 jul. 2015. Citado na página 80.

STARR, M. *Using intentions to control a robotic prosthetic*: A neural implant on the area of the brain that controls the patient's intention to move could be the key to better robotic prosthetics. CNET, 2015. Disponível em: <http://www.cnet.com/news/using-intentions-to-control-a-robotic-prosthetic/>. Acesso em: 26 mai. 2015. Citado na página 42.

SUNPLUS TECHNOLOGY. *SPCE061A Datasheet*: 16-bit Soundo Controller with 32k x 16 Flash Memory. [S.l.], 2002. Citado na página 106.

TAN, L.; WANG, N. Future Internet: The Internet of Things. In: INTERNATIONAL CONFERENCE ON ADVANCED COMPUTER THEORY AND ENGINEERING, 2010, China. *Proceedings...* Chengdu: IEEE, 2010. Citado na página 88.

TEVAH, R. T. *Implementação de um Sistema de Reconhecimento de Fala Contínua com Amplo Vocabulário para o Português Brasileiro*. 102 f. Dissertação (Mestrado em Ciências) — Universidade Federal do Rio de Janeiro, COOPE, Rio de Janeiro, 2006. Citado 3 vezes nas páginas 37, 48 e 56.

TEXAS INSTRUMENTS. *Sitara Processors Forum*: Big Endian support feasibility on AM335x chip from silicon design point view. Disponível em: <https://e2e.ti.com/support/arm/sitara_arm/f/791/p/431713/1543339>. Acesso em: 30 jan. 2016. Citado na página 100.

TEXAS INSTRUMENTS. *Sitara Processors Forum*: the big-endian and little-endian settings for am335x. Disponível em: <https://e2e.ti.com/support/arm/sitara_arm/f/791/t/274171>. Acesso em: 30 jan. 2016. Citado na página 100.

UFPA. *Laboratório de Processamento de Sinais*. Disponível em: <http://laps.ufpa.br/>. Acesso em: 28 jan. 2016. Citado na página 95.

UNIVERSAL SPEECH SOLUTIONS LLC. *UniMRCP Open Source Project*. Disponível em: <http://www.unimrcp.org/>. Acesso em: 17 jun. 2015. Citado na página 91.

UNIVERSIDADE FEDERAL DE SANTA CATARINA. *Modelo PRODIP*. 2016. Disponível em: <http://emc5302.ogliari.prof.ufsc.br/artigo/modelo-prodip>. Acesso em: 16 jan. 2016. Citado na página 77.

UZUNAI, Y.; BICAKCI, K. SHA: A secure voice activated smart home for quadriplegia patients. In: INTERNATIONAL CONFERENCER ON BIOINFORMATICS AND BIOMEDICINE WORKSHOPS, 2007, Califórnia. *Proceedings...* Freemond: IEEE, 2007. Citado 3 vezes nas páginas 48, 56 e 113.

VALLÉS, M. et al. Multimodal environmental control system for elderly and disabled people. In: INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY, 18., 1996, Holanda. *Proceedings...* Amsterdam: IEEE, 1996. Citado 2 vezes nas páginas 45 e 112.

VIEIRA, V. J. D. *Avaliação de Distúrbios da Voz por meio de Análise de Quantificação de Recorrência*. 218 f. Dissertação (Mestrado em Ciências no domínio da Engenharia Elétrica) — Universidade Federal da Paraíba, João Pessoa, 2014. Citado na página 120.

VOICE Recognition Module V2. Disponível em: <http://www.elechouse.com/elechouse/index.php?main_page=product_info&cPath=&products_id=2151>. Acesso em: 12 jun. 2015. Citado na página 52.

VOXFORGE. *Acoustic Model Creation*. Disponível em: <http://www.voxforge.org/home/docs/acoustic-model-creation>. Acesso em: 30 jan. 2016. Citado na página 100.

VOXFORGE. *Codetrain.scp*. Disponível em: <https://raw.githubusercontent.com/VoxForge/develop/master/howto/codetrain.scp>. Acesso em: 18 jul. 2015. Citado na página 175.

VOXFORGE. Scripts. Disponível em: <http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/how-to/script>. Acesso em: 18 jul. 2015. Citado na página 175.

VOXFORGE. *How-to*: Create acoustic model - with script. Disponível em: <http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/how-to>. Acesso em: 03 jul. 2015. Citado 3 vezes nas páginas 96, 97 e 168.

VOXFORGE. *Running Julius Live*. Disponível em: <http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/tutorial/run-julius>. Citado na página 71.

VOXFORGE. Disponível em: <http://www.voxforge.org/>. Acesso em: 11 jun. 2015. Citado na página 48.

VOXFORGE. *VoxforgeDict.txt*. Disponível em: <http://www.voxforge.org/uploads/2D/Oe/2DOeE3hM6livfkY6yxM97g/VoxForgeDict.txt>. Acesso em: 18 jul. 2015. Citado na página 172.

VOXFORGE. *Tutorial*: Create Acoustic Model - Manually. 2016. Disponível em: <http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/tutorial>. Acesso em: 03 jul. 2015. Citado 7 vezes nas páginas 60, 96, 97, 120, 165, 166 e 168.

VOXFORGE FORUM. *Speech Recognition Engines*: Julius plug-in for asterisk. 2012. Disponível em: <http://www.voxforge.org/home/forums/message-boards/speech-recognition-engines/julius-plug-in-for-asteriks>. Acesso em: 18 jun. 2015. Citado na página 92.

WIKI DEBIAN. *rt2800usb*. Disponível em: <https://wiki.debian.org/rt2800usb>. Acesso em: 29 jan. 2016. Citado na página 157.

WIKIPÉDIA. *File Descriptor*. Disponível em: <https://en.wikipedia.org/wiki/File_descriptor>. Acesso em: 17 jun. 2015. Citado na página 92.

WIKIPÉDIA. *Rapid Application Development*. Disponível em: <https://pt.wikipedia.org/wiki/Rapid_Application_Development>. Acesso em: 16 jan. 2016. Citado na página 77.

WINSDOM KING. *Products Catalog*: Mouth sticks. 2015. Disponível em: <http://www.wisdomking.com/mouth-sticks>. Acesso em: 02 jun. 2015. Citado na página 40.

WISDOM KING. *Products Catalog*: Headpointers. 2015. Disponível em: <http://www.wisdomking.com/headpointers>. Acesso em: 02 jun. 2015. Citado na página 40.

WRIGHT, S. J. et al. Optimization Algorithms and Applications for Speech and Language Processing. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, v. 21, n. 11, p. 2231–2243, Nov. 2013. Citado na página 120.

WRITER. Disponível em: <https://www.libreoffice.org/discover/writer/>. Acesso em: 12 jun. 2015. Citado na página 52.

YAGHMOUR, K. et al. *Construindo Sistemas Linux Embarcados*: Conceitos, técnicas, truques e dicas. Rio de Janeiro: Alta Books, 2014. 377 p. ISBN 978-85-7608-343-6. Citado na página 81.

YANG, L. et al. A home mobile healthcare system for wheelchair users. In: INTERNAITONAL CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK IN DESIGN, 18., 2014, Taiwan. *Proceedings...* Hsinchu: IEEE, 2014. Citado 3 vezes nas páginas 31, 37 e 42.

YNOGUTI, C. A.; VIOLARO, F. A Brazilian Portuguese Speech Database. In: XXVI SIMPÓSIO BRASILEIRO DE TELECOMUNICAçõES, 2008, Brasil. *Anais...* Rio de Janeiro, RJ, 2008. Citado na página 95.

YOUNG, S. et al. *The HTK Book*. Cambridge: Cambridge Univertsity Engineering Department, 2009. 384 p. Citado 13 vezes nas páginas 61, 62, 63, 64, 65, 66, 67, 68, 96, 97, 120, 165 e 168.

ZIGBEE Alliance. Disponível em: <http://www.zigbee.org/>. Acesso em: 12 jun. 2015. Citado na página 51.

ZOIPER. Disponível em: <http://www.zoiper.com/en>. Acesso em: 18 jun. 2015. Citado na página 88.

Apêndices

APÊNDICE A – Tutorial de instalação e configuração do Debian Linux na Beagleboard

A instalação da imagem de um sistema operacional na memória *flash* interna (*Embedded MultiMedia Card* – eMMC) da Beaglebone Black está bem descrita, e com instruções claras, em documentos fornecidos por sua comunidade de suporte. Este tutorial, no entanto, apresenta algumas configurações adicionais necessárias ao desenvolvimento deste projeto.

A.1 Criando um cartão SD inicializável no Linux

Para a realização dessa etapa, é necessário que se disponha de um cartão de memória *microSD* com capacidade de 4GB ou superior.

Primeiramente, utilizando um PC com o sistema operacional Ubuntu Linux 14.04 instalado, deve-se realizar o *download* da imagem da distribuição Debian a ser instalada. Esse arquivo pode ser salvo em qualquer diretório do sistema. As versões de distribuições mais recentes podem ser encontradas no *site* da comunidade Beagleboard[1] .

É importante salientar que existem dois tipos de imagem que podem ser baixadas: uma que é executada diretamente do cartão *microSD*, quando inserido no leitor da Beagleboard; ou o tipo de imagem específica para ser escrita na memória *flash* interna da BBB, conhecida como "*flasher image*".

A instalação realizada no presente trabalho utilizou uma imagem para a gravação do sistema operacional na memória *flash* interna da Beagleboard, tendo sido realizado o *download* da imagem apropriada da distribuição Debian 7.8, na versão *console* (somente linha de comando, sem interface gráfica), através do seguinte comando no terminal do Linux:

```
# wget https://rcn-ee.com/rootfs/bb.org/release/2015-03-01/console/BBB-eMMC-flasher-debian-
7.8-console-armhf-2015-03-01-2gb.img.xz
```

Deve-se atentar para o fato de que o comando acima está apresentado em duas linhas somente devido ao pouco espaço disponível, devendo ser fornecido ao terminal em uma única linha.

Após isso, o arquivo descarregado deve ser descompactado, através do seguinte comando:

[1] BEAGLEBOARD. *Latest Firmware Images*. 2015. Disponível em: <http://beagleboard.org/latest-images>. Acesso em: 24 out. 2015.

```
# xz -d BBB-eMMC-flasher-debian-7.8-console-armhf-2015-03-01-2gb.img.xz
```

Com isso a imagem está pronta para ser escrita no cartão *microSD*, que será posteriormente inserido no leitor da Beagleboard para então realizar-se o procedimento de escrita na memória *flash* interna da BBB.

Então, agora deve-se inserir o cartão *microSD* no computador em que a imagem do Debian foi descarregada. Em seguida, deve-se verificar o dispositivo em que foi montado o referido cartão, através do seguinte comando:

```
# mount
```

O dispositivo de montagem do cartão de memória é identificado pela nomenclatura do tipo */dev/sdX*, em que "X"é uma letra específica para cada sistema. No caso dessa instalação, o cartão de memória foi definido pelo sistema no dispositivo */dev/sdb*.

Por precaução, é importante que o cartão de memória seja formatado utilizando-se o sistema de arquivos FAT (*Fat Allocation Table*).

Com o cartão de memória devidamente preparado, pode-se escrever nele a imagem do Debian, executando-se o seguinte comando:

```
# sudo dd if=BBB-eMMC-flasher-debian-7.8-console-armhf-2015-03-01-2gb.img of=/dev/sdX
```

Devendo o "X"ser substituído pelo dispositivo específico associado ao cartão *microSD* em cada caso.

A execução do comando acima demora cerca de 5 a 10 minutos para uma escrita completa da imagem do sistema operacional no cartão de memória.

A.2 Instalando a imagem na memória *flash* da Beagleboard

Nesse ponto, já com o cartão *microSD* configurado com a imagem "*flasher*"do Debian, segue os passos a serem realizados para sua instalação na memória *flash* interna da Beagleboard:

1. Para realizar esse procedimento, deve ser utilizada uma imagem *flasher* do sistema operacional a ser instalado.

2. O processo pode ser realizado utilizando um adaptador de 5V/2A ou mesmo fornecendo a energia pela USB. O cabo ethernet, *shields* e quaisquer periféricos USB devem ser removidos antes de se iniciar o procedimento.

3. Desligue a Beaglebone e desconecte o cabo de energia.

4. Conecte o cartão *microSD* no referido *slot* da BBB.

5. Segure o botão de *boot* (próximo ao *slot* SD) e conecte a energia à placa, aguardando que os 4 LEDs próximos ao conector *ethernet* acendam e, após um tempo se apaguem. Somente após isso, pode-se soltar o botão de *boot*. Então, os 4 LEDs próximos ao conector *ethernet* ficarão acendendo em sequência durante os próximos 5 a 25 minutos. Deve-se aguardar os LEDs pararem de piscar, ficando os 4 acesos de forma contínua (sem apagar). Caso os LEDs não pisquem ou fiquem piscando por mais de 45 minutos, deve se retirar a energia da placa e tentar reiniciá-la novamente com o botão de *boot* pressionado.

6. Em caso de sucesso, retira-se o cartão *microSD* e pressiona-se o botão *reset*.

A Beaglebone agora está com o novo sistema operacional instalado na memória *flash* interna.

A.3 Configurando o Debian pós-instalação na Beagleboard

A.3.1 Realizando o primeiro acesso ao terminal da Beagleboard

Para ter acesso ao terminal de comando do sistema operacional instalado na Beagleboard, é possível utilizar-se de dois modos:

1. Utilizar um monitor conectado à interface HDMI da Beagleboard, juntamente com um teclado conectado à sua porta USB. Caso seja necessária a utilização de mais dispositivos USB, pode-se utilizar um *hub* USB.

2. A partir de um computador com o Linux, o Windows, ou o MacOS, pode-se configurar o acesso à Beagleboard via interface *miniUSB*, através de uma conexão SSH[2] .

3. Utilizar uma conexão *ethernet* cabeada entre a Beagleboard e um PC para acesso via SSH.

Como neste tutorial a Beagleboard foi preparada com a distribuição Debian 7.8 Linux, o sistema já está configurado por padrão para realizar a configuração de IP automática através do protocolo DHCP (*Dynamic Host Configuration Protocol*). Então, adotou-se a terceira forma de acesso supracitada.

Caso se utilize outro sistema operacional, essa forma de acesso pode não funcionar, sendo necessário que o primeiro acesso seja realizado por um dos dois primeiros modos

[2] ADAFRUIT. *SSH to BeagleBone Black over USB*. 2013. Disponível em: <https://learn.adafruit.com/ssh-to-beaglebone-black-over-usb/overview>. Acesso em: 24 fev. 2015.

mencionados anteriormente para que as configurações de DHCP[3] sejam realizadas e, posteriormente, se possa acessar o terminal da Beagleboard utilizando-se o terceiro modo.

Esse último modo de acesso é interessante, pois a alimentação da BBB pode agora ser realizada através do adaptador fonte de 5V/1A, que fornece uma potência mais eficiente para o desenvolvimento de aplicações que exigirão a utilização das saídas GPIO para fornecimento de corrente, não sendo necessário o cabo USB para alimentação da placa.

Dando continuidade, utilizou-se um computador com o Ubuntu 14.04 Linux, tendo-se configurado sua interface *ethernet* para o modo "Compartilhado com outros computadores". Em seguida, conectou-se um cabo *ethernet* entre a interface *ethernet* do computador e a interface *ethernet* da Beagleboard. Com isso, a conexão de rede ponto a ponto deve estar configurada.

Observou-se, então, o endereço de rede da interface *ethernet* do computador através do seguinte comando:

```
1 # ifconfig
```

Observou-se que a faixa de endereços de rede utilizada na conexão foi 10.42.0.0/24. Então, para descobrir o endereço de rede configurado na Beagleboard, executou-se o seguinte comando:

```
1 # nmap -sP 10.42.0.0/24
```

Para que o comando acima funcione, é necessário que o pacote *nmap* esteja instalado no sistema. A saída desse comando retorna os endereços IP configurados na rede.

Então, de posse do endereço IP configurado na Beagleboard (no caso, 10.42.0.18/24), executa-se o acesso SSH através do seguinte comando:

```
1 # ssh debian@10.42.0.18
```

Por padrão, o usuário criado no Debian que instalamos na sessão anterior é configurado com o *username debian* e com o *password temppwd*.

A.3.2 Configurando servidores DNS (*Domain Name System*)

Tendo-se realizado o acesso ao terminal do sistema operacional instalado na Beaglebone Black, é interessante primeiramente realizarmos algumas configurações de rede. Como a BBB está conectada ao computador diretamente, estando este com a conexão configurada

[3] MATHWORKS. *Configure Network Connection with BeagleBoard Hardware*. 2015. Disponível em: <http://www.mathworks.com/help/supportpkg/beagleboard/ug/getting-the-beagleboard-ip-address.html>. Acesso em: 24 jan. 2016.

para compartilhamento com outros computadores, o acesso à internet a partir da Beagleboard é realizado através do computador, que deve estar conectado ao roteador da rede local. Assim, pode ser que os servidores DNS não tenham sido configurados corretamente.

Antes de iniciar a configuração DNS, é interessante saber que na distribuição Debian instalada o utilitário *ifconfig* do Linux não está disponibilizado para o usuário *debian*, sendo conveniente adicioná-lo à variável *PATH* desse usuário. Para tanto, executa-se o seguinte comando:

```
1 PATH=$PATH:/sbin
```

Pode-se, então, configurar servidores DNS abertos, acrescentando-se a seguinte linha ao arquivo */etc/resolv.conf/interfaces* nas configurações da interface ethernet:

```
1 dns-nameservers 189.38.95.96 189.38.95.95
```

Esses são os endereços dos servidores DSN da GigaDNS[4] , que fornece serviço de DNS aberto no Brasil.

A.3.3 Configurando local, data e hora

Para o ajuste da localidade, data e hora do sistema, primeiramente deve-se atualizar a lista de repositórios de pacotes do sistema, através do comando:

```
1 # apt-get install update
```

Então, instala-se o pacote *locale* com o seguinte comando:

```
1 # apt-get install locales
```

Em seguida, gera-se o locale para português brasileiro (pt-BR), com o comando a seguir:

```
1 # locale-gen pt_BR pt_BR.UTF-8
```

Depois, recarrega-se a configuração dos locais com o comando:

```
1 # dpkg-reconfigure locales
```

Para a configuração do fuso horário, executa-se o seguinte comando:

[4] GIGADNS. *GigaDNS*. Disponível em: <http://gigadns.web1407.kinghost.net//>. Acesso em: 24 jan. 2016.

```
# dpkg-reconfigure tzdata
```

Então, na tela apresentada, escolhe-se o fuso horário mais adequado. No caso deste tutorial, foi definida a opção "América/Recife".

Para que a data e hora sejam ajustadas automaticamente através dos servidores da internet, deve-se instalar o cliente NTP (*Network Time Protocol*), através do seguinte comando:

```
# apt-get install ntpdate
```

Estando o cliente NTP instalado, pode-se agora atualizar a data e hora a partir dos servidores *ntp.br*[5] através do comando:

```
# ntpdate pool.ntp.br
```

Com isso, as configurações de local, data e hora estão realizadas.

A.3.4 Instalando os pacotes de desenvolvimento necessários

Aqui apresentamos a instalação de alguns pacotes de desenvolvimento necessários para a implementação do projeto.

Com o intuito de configurar o suporte à linguagem Python, foram instalados os pacotes com as ferramentas e bibliotecas dessa linguagem através da execução do seguinte comando:

```
# apt-get install python python-setuptools python-dev python-pip python-smbus -y
```

Além disso, para prover o adequado funcionamento do *engine* Julius na plataforma, é necessário que sejam instalados alguns pacotes e bibliotecas, executando-se os seguintes comandos:

```
# apt-get install build-essential
# apt-get install flex zlib1g-dev
# apt-get install libasound2-dev libesd0-dev libsndfile1-dev
```

O utilitário *sox* de conversão entre formatos de arquivos de áudio também foi instalado através do comando:

```
# apt-get install sox
```

[5] NTP.BR. *NTP.br*. Disponível em: <http://ntp.br/>. Acesso em: 24 jan. 2016.

A.4 Instalando o Julius

Nessa seção serão apresentados os procedimentos para instalação do Julius rev. 4.3.1 em sistema operacional Debian 7.8 embarcado na plataforma Beaglebone Black rev. C. Pressupõe-se que todos os pacotes e bibliotecas necessárias já estão devidamente instaladas no sistema.

A.4.1 Obtendo e descompactando os pacotes do Julius

Inicialmente, deve-se abrir o terminal de comandos do Linux, acessar a pasta do usuário no sistema e criar uma subpasta de nome *bin*, caso ainda não exista, através dos comandos abaixo.

```
1 $ cd /home/debian
2 $ mkdir /home/debian/bin
```

A seguir, deve-se salvar os arquivos para instalação do Julius nessa pasta que acabamos de criar. Para isso, pode-se obter a versão mais atual do Julius, realizando-se o *download* dos arquivos fonte (*Source (tarball)*) na página do projeto (JULIUS, 2014).

Feito isso, é necessário agora extrair os arquivos, utilizando o comando relacionado abaixo. Certifique-se de estar com o *shell* Linux localizado na pasta onde o arquivo compactado fora salvo.

```
1 $ tar -xzvf julius-4.3.1.tar.gz
```

Tendo sido o comando executado com sucesso, foi criada um novo subdiretório na pasta */home/debian/bin*, com o nome *julius-4.3.1*.

A.4.2 Compilando o Julius a partir dos arquivos fonte

A compilação do Julius a partir de seus arquivos fonte nos permite determinar algumas configurações de instalação como, por exemplo, limitação do tamanho do vocabulário ou do comprimento da entrada, variações de algoritmos de busca, dentre outros aspectos (LEE, 2010).

Por padrão, ao compilarmos sem indicar opções de instalação, o comando *make install* realiza uma cópia das bibliotecas, executáveis, cabeçalhos e manuais para a pasta */usr/local* do sistema. Com o intuito de que os arquivos sejam instalados num diretório qualquer que se deseje, é possível iniciar o processo de compilação através do comando *./configure*, passando-lhe o parâmetro *prefix*, conforme o exemplo abaixo.

```
1 $ ./configure --prefix=/home/nomedousuario/diretoriodeinstalacao
```

No entanto, para o caso dessa instalação, não será passado esse parâmetro, de modo que o Julius será instalado no diretório padrão. É importante notar que, os termos *nomedosusario* e *diretoriodeinstalacao* devem ser substituídos pelos nomes dos respectivos usuário e diretório que se deseja.

Uma particularidade observada para o caso desse trabalho foi a necessidade de se estender o tamanho do vocabulário suportado pelo Julius. Para isso, estando com o *prompt* de comando no diretório */home/debian/bin/julius-4.3.1*, deve-se iniciar o processo de compilação executando-se o comando *./configure* com a opção *enable-words-int*, conforme apresentado a seguir:

```
1 $ ./configure --enable-words-int
```

Caso este comando tenha sido executado com sucesso, sua saída se apresentará sem mensagens de erro, finalizando com uma mensagem semelhante a ilustrada na Figura 35.

Figura 35 – Sucesso do comando *configure* na instalação do Julius.

```
************************************************************
Julius/Julian libsent library rev.4.3.1:

- Audio I/O
    primary mic device API   : alsa (Advanced Linux Sound Architecture)
    available mic device API : alsa oss esd
    supported audio format   : various formats by libsndfile ver.1
    NetAudio support         : no
- Language Modeling
    class N-gram support     : yes
- Libraries
    file decompression by    : zlib library
- Process management
    fork on adinnet input    : no

  Note: compilation time flags are now stored in "libsent-config".
        If you link this library, please add output of
        "libsent-config --cflags" to CFLAGS and
        "libsent-config --libs" to LIBS.
************************************************************
```

Fonte: Autoria própria

Então, para gerar as bibliotecas e binários, deve ser executado o seguinte comando:

```
1 $ make all
```

Não apresentando a saída mensagens de erro, devemos instalar as referidas bibliotecas e binários através do comando:

```
1 # make install
```

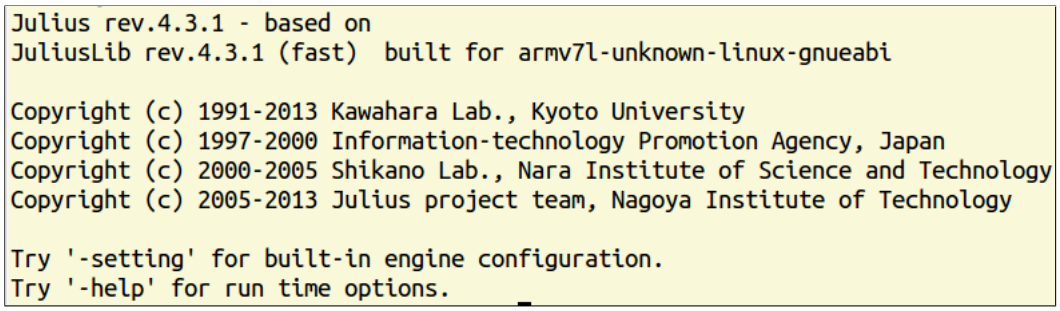
A saída deste último comando também não deve apresentar mensagens de erro. É importante estar atento para isto!

A instalação do Julius pode ser verificada através do seguinte comando:

```
1 $ julius
```

Caso a instalação tenha sido realizada com sucesso, a saída desse comando deve ser semelhante à ilustrada na Figura 36.

Figura 36 – Saída do comando *julius*.



Fonte: Autoria própria

Com isso, o *engine* de reconhecimento de voz Julius está corretamente instalado no sistema. Informações detalhadas sobre configuração e utilização do *software* podem ser encontradas em (LEE, 2010).

A.5 Instalando e configurando o Asterisk

Para a instalação do Asterisk optou-se pela versão 1.8.13.1, distribuída pelos repositórios do Debian. Assim, o servidor é instalado executando-se o comando:

```
1 # apt-get install asterisk
```

Durante o processo de instalação foi informado o código numérico do país 55 (Brasil). Ao fim desse processo, o Asterisk está instalado com sucesso no sistema.

A.5.1 Realizando testes com o Asterisk

Para verificar o funcionamento do Asterisk, foi realizado um teste em que se acrescentou ao arquivo de configuração de canais */etc/asterisk/sip.conf* as seguintes linhas:

```
1   [general]
2       context = naoautenticado
3       alowguest = no
4       udpbindaddr = 192.168.12.1
5       bindport = 5060
6       disallow = all
7       allow = alaw,ulaw,gsm
8       language = pt_BR
9
10  [comum_a_ramais](!)
11      type = friend
12      context = ramais
13      host = dynamic
14      secret = 1234
15
16  [enderecoMAC1](comum_a_ramais)
17
18  [enderecoMAC2](comum_a_ramais)
```

As linhas 1 a 8 apresentam as opções gerais de configuração para os módulos dos canais SIP criados para dispositivos neste arquivo de configuração. A linha 2 indica o contexto padrão para chamadas recebidas (colocado somente com o propósito de lidar com segurança sobre chamadas não registradas). A linha 3 desabilita chamadas não registradas. A linha 4 escuta as requisições UDP na interface indicada pelo endereço IP informado (para escutar todas as interfaces utiliza-se 0.0.0.0). Já na linha 5 é especificada a porta por onde são recebidas as chamadas.

A linha 6 reinicia os *codecs* de voz que serão utilizados ou oferecidos pelos dispositivos dos canais criados (pode ser especificado para um dispositivo específico nas opções de seu canal). Na linha 7 são determinados os *codecs* de áudio que podem ser aceitos ou requisitados aos dispositivos. A linha 8 indica o idioma.

As linhas 10 a 14 estão relacionadas à seção modelo de nome *comum_a_ramais* com informações comuns aos canais relacionados a essa seção nos parênteses após o nome do canal, listados nas linhas 16 e 18. A linha 11 significa que o módulo controlador do canal fará a correspondência com o nome do usuário primeiro e com o endereço IP depois. Já na linha 12, é definido o nome do contexto para onde as chamadas dos canais relacionados a essa seção serão encaminhados no plano de discagem. A linha 13 força a necessidade de que o dispositivo relacionado a um canal registre-se no Asterisk. Enquanto na linha 14 é configurada a senha para registro dos dispositivos relacionados a essa seção.

As linhas 16 e 18 criam canais para um dispositivo utilizando as configurações da seção modelo *comum_a_ramais*. A identificação dos canais foi realizada indicando-se o endereço MAC (*Media Access Control*) de cada dispositivo na linha referente à criação de seu respectivo canal.

Definidas as configurações acima, foi criado um plano de discagem no arquivo */etc/asterisk/extensions.conf* para a realização de um teste inicial com o Asterisk, conforme apresentado a seguir.

```
1  [ramais]
2      exten => 202,1,Set(Falar=Victor)
3          same => n,Answer()
4          same => n,Wait(2)
5          same => n,SayAlpha(${Falar})
6          same => n,Wait(2)
7          same => n,Hangup()
```

Nesse arquivo, a linha 1 indica que o plano de discagem será executado pelos canais que cheguem ao Asterisk para o contexto *ramais*, conforme definido no arquivo *sip.conf*. A linha 2 cria uma extensão definindo os passos (linhas 3 a 7) que a chamada realizará.

Nesse exemplo, a chamada é automaticamente atendida no servidor, em seguida espera dois segundos. Depois responde, a partir de gravações de áudio padrão já contidas na instalação, pronunciando as letras da palavra *Victor*, que está definida na variável *Falar*. Em seguida, espera outros doi segundos e, então, encerra a chamada.

Maiores detalhes sobre a configuração do Asterisk, criação e configuração de canais e planos de discagem podem ser encontradas em Madsen, Meggelen e Bryant (2011).

A.5.1.1 Configuração do Zoiper para realização do teste

Para efetuar o teste, foi realizada uma chamada a partir de um *smartphone* com o *softphone* Zoiper instalado.

Na área de configuração do aplicativo Zoiper, foram configurados os seguintes parâmetros:

```
1  Host 192.168.12.2
2  Nome Android
3  Proxy 192.168.12.1
4  Senha 1234
```

A linha 1 representa a indicação do endereço IP com que o *smartphone* foi configurado para acesso à rede disponibilizada pelo *access point*. A linha 2 representa o nome do dispositivo na rede. Já na linha 3, é indicado o endereço IP do servidor Asterisk (Beagleboard). Enquanto na linha 4, é fornecida a senha de acesso ao servidor, que foi definida anteriormente no arquivo *sip.conf*.

Uma vez que o *smartphone* esteja conectado à rede WiFi disponibilizada pelo *access point*, estando o *softphone* configurado com as informações apresentadas, o ele é, então, capaz de se registrar no servidor Asterisk. Estando registrado, é possível a realização de uma chamada à extensão criada no arquivo *extensions.conf*, discando-se diretamente o número da extensão. Nesse caso, *202*.

A.5.1.2 Ramal configurado para a gravação das amostras de treinamento

Com o intuito de realizar a gravação das amostras de treinamento para a criação dos modelos acústicos a serem utilizados pela interface em desenvolvimento, foi elaborada uma extensão que realiza a gravação do fluxo de áudio do canal de uma chamada no plano de discagem do Asterisk, armazenando esse áudio em um arquivo com o formato *.wav*, taxa de amostragem de 8KHz, monocanal, com 16 bits por amostra (características do canal do Asterisk).

A seguir é apresentada uma forma geral da extensão desenvolvida:

```
1   [ramais]
2       exten => 2002,1,Answer
3           same => n,Wait(2)
4           same => n,Playback(beep)
5           same => n,Record(/diretorio/de/destino/do/audio/sample%d:wav)
6           same => n,Wait(2)
7           same => n,Playback(${RECORDED_FILE})
8           same => n,Wait(2)
9           same => n,Hangup()
```

Após realizar a gravação do canal de áudio, essa extensão reproduz o arquivo gravado para audição pelo usuário.

A.5.1.3 Ramal configurado para a integração com reconhecimento automático da fala via EAGI

A integração da capacidade de reconhecimento automático da fala ao servidor Asterisk foi implementada através de uma aplicação EAGI, que executa um *scritp* externo escrito na linguagem Python, o qual está disponível no Anexo C.

A seguir é apresentado o código da extensão responsável por interagir com o *script* EAGI de integração entre Asterisk e Julius, como também por sinalizar o controle das GPIO da Beaglebone Black ao sistema Linux.

```
1   [ramais]
2       exten => 206,1,Answer()
3           same => 2,Set(OUT=0)
4           same => n,Wait(1)
5           same => n,Playback(prompt0)
6           same => n,Playback(beep)
7           same => n,EAGI(eagi-pyspeech-shell.py)
8           same => n,GotoIf($[${OUT} = 1]?outdoor,1:indoor,1)
9           exten => indoor,1,System(${GPIO})
10              same => n,System(${ESTADO})
11              same => n,Goto(206,2)
12          exten => outdoor,1,System(${GPIO0})
13              same => n,System(${GPIO1})
14              same => n,System(${GPIO2})
15              same => n,System(${GPIO3})
16              same => n,System(${GPIO4})
```

```
17                              same => n,System(${ESTADO0})
18                              same => n,System(${ESTADO1})
19                              same => n,System(${ESTADO2})
20                              same => n,System(${ESTADO3})
21                              same => n,System(${ESTADO4})
22                              same => n,Goto(206,2)
```

A.6 Configurando *access point* USB na Beagleboard

O desenvolvimento da segunda abordagem do projeto envolveu a configuração de um adaptador de rede sem fio (WiFi 802.11), conectado à porta USB da Beaglebone Black, funcionando como *access point*.

Para que o sistema Debian passe a dar suporte ao adaptador WiFi USB Intelbrás WBN900 foi inicialmente necessário se obter a informação do *driver* necessário. A sessão de especificações técnicas do guia de instalação[6] desse adaptador contém a informação do *chipset* utilizado por ele, que é o Ralink RT5370.

Assim, seguindo os procedimentos encontrados na página *wiki* do Debian[7] , foi realizada a instalação dos referidos *drivers*, conforme a seguir.

Inicialmente, é necessário adicionar o repositório *Non-free* no arquivo de configuração do Linux */etc/apt/sources.list*. Para isso, adiciona-se a seguinte linha ao arquivo:

```
1   deb http://http.debian.net/debian/ wheezy main contrib non-free
```

Em seguida, deve-se atualizar a lista de repositórios do *apt* e realizar a instalação do pacote com os *drivers* desejados, através do seguinte comando:

```
1   # apt-get update && apt-get install firmware-ralink
```

Finalizada a instalação, agora podemos desligar a placa Beaglebone Black, conectar o adaptador WiFi em sua porta USB e, então, ligar a placa novamente.

Pode-se verificar que os módulos necessários para o correto funcionamento do adaptador *wireless* foram carregados através do seguinte comando:

```
1   # lsmod
```

A saída desse comando deve mostrar uma lista contendo os módulos rt2800, conforme ilustra a Figura 37.

6 INTELBRÁS. *Guia de instalação*: WBN900. Disponível em: <http://www.intelbras.com.br/sites/default/files/downloads/guia_de_instalacao_-_adaptador_usb_wireless_n_150_mbps_wbn_900.pdf>. Acesso em: 29 jan. 2016.

7 WIKI DEBIAN. *rt2800usb*. Disponível em: <https://wiki.debian.org/rt2800usb>. Acesso em: 29 jan. 2016.

Figura 37 – Saída do comando *lsmod*.

```
Module                  Size  Used by
g_ether                23958  0
libcomposite           15028  1 g_ether
8021q                  14719  0
garp                    5420  1 8021q
stp                     1824  1 garp
llc                     5042  2 stp,garp
arc4                    1691  2
rt2800usb              13470  0
rt2800lib              41926  1 rt2800usb
rt2x00usb               9839  1 rt2800usb
rt2x00lib              35799  3 rt2x00usb,rt2800lib,rt2800usb
mac80211              424813  3 rt2x00lib,rt2x00usb,rt2800lib
cfg80211              354018  2 mac80211,rt2x00lib
rfkill                 16672  2 cfg80211
omap_rng                4062  0
mt7601Usta            639170  0
```

Fonte: Autoria própria.

Com os módulos devidamente instalados, deve-se realizar a configuração do *access point* WiFi. Para tanto, é necessário modificar o arquivo */etc/network/interfaces*. Nesse arquivo, a seção referente às configurações do adaptador de rede sem fio, no caso desse tutorial o *wlan0*, devem ser definidas de forma estática, conforme ilustra o exemplo a seguir:

```
1  # auto wlan0 ok
2      iface wlan0 inet static ok
3              address 192.168.12.1
4              network 192.168.12.0
5              netmask 255.255.255.0
6              broadcast 192.168.12.255
```

Após isso, deve-se reiniciar as interfaces de rede para que as configurações tomem efeito, através do seguinte comando:

```
1  # /etc/init.d/networking restart
```

Utilize o comando *ifconfig* para visualizar se a placa de rede sem fio está devidamente configurada, conforme o exemplo da Figura 38.

Em seguida, deve-se instalar o *hostapd*, que é o pacote responsável por prover a função de *access point*, executando-se o seguinte comando:

```
1  # apt-get install hostapd
```

Feito isso, deve-se modificar o arquivo */etc/default/hostapd*, que passa a ter a seguinte linha:

```
1  DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Figura 38 – Visualizando configuração IP da placa de rede sem fio.

```
wlan0     Link encap:Ethernet  Endereço de HW 00:1a:3f:2b:70:e4
          inet end.: 192.168.12.1  Bcast:192.168.12.255  Masc:255.255.255.0
          endereço inet6: fe80::21a:3fff:fe2b:70e4/64 Escopo:Link
          UP BROADCASTRUNNING MULTICAST  MTU:1500  Métrica:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          colisões:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:576 (576.0 B)
```

Fonte: Autoria própria.

Estando salvas as configurações realizadas anteriormente, deve-se criar o arquivo */etc/hostapd/hostapd.conf*, adicionando-lhe o seguinte conteúdo:

```
### Wireless network name ###
    interface=wlan0
    #
    ### Set your bridge name ###
    #bridge=br0

    #driver
    driver=nl80211

    country_code=BR

    ssid=beaglebone

    channel=7

    hw_mode=g

    # # Static WPA2 key configuration
    # #1=wpa1, 2=wpa2, 3=both
    wpa=2

    wpa_passphrase=senhadarede

    ## Key management algorithms ##
    wpa_key_mgmt=WPA-PSK
    #
    ## Set cipher suites (encryption algorithms) ##
    ## TKIP = Temporal Key Integrity Protocol
    ## CCMP = AES in Counter mode with CBC-MAC
    wpa_pairwise=TKIP
    #rsn_pairwise=CCMP
    #
    ## Shared Key Authentication ##
    auth_algs=1
    ## Accept all MAC address ###
    macaddr_acl=0
    #enables/disables broadcasting the ssid
    ignore_broadcast_ssid=0
    # Needed for Windows clients
    eapol_key_index_workaround=0
```

Em seguinda, reinicia-se o serviço *hostapd*, através do comando a seguir:

```
# hostapd /etc/hostapd/hostapd.conf
```

Após isso, os dispositivos *wireless* a área de abrangência da rede provida pelo *access point* já devem ser capazes de encontrá-la com o nome beaglebone.

No entanto, ao conectar-se à rede, as configurações IP dos dispositivos conectados ainda deverão ser feitas manualmente.

Para que as configurações de IP sejam distribuídas automaticamente, deve-se instalar um servidor *dhcp* na beagleboard. Na sequência são apresentados os passos para instalação e configuração do servidor *isc-dhcp-server*.

Para instalar o serviço de *dhcp*, executa-se o seguinte comando:

```
# apt-get update && apt-get install isc-dhcp-server
```

Após instalado, o servidor *dhcp* deve ser configurado para distribuir as configurações de IP para a mesma rede com que foi configurada a interface de rede sem fio. No caso desse exemplo, acrescenta-se ao arquivo */etc/dhcp/dhcpd.conf* o seguinte conteúdo:

```
subnet 192.168.12.0 netmask 255.255.255.0 {
        range 192.168.12.2 192.168.12.10;
      }
```

Uma vez que a configuração anterior foi salva, deve-se reiniciar o serviço *dhcp* do Linux executando-se o seguinte comando:

```
# /etc/init.d/isc-dhcp-server restart
```

Terminada esta etapa, a conexão do *smartphone* com a beagleboard através do *access point* configurado pode ser realizada com sucesso.

A.7 Configurando permissões de acesso às GPIO na Beagleboard

No caso desse projeto, o controle das GPIO da Beagleboard é realizado a partir do *script* EAGI implementado, que retorna ao plano de discagem do Asterisk os comandos em linguagem *shell* responsáveis pelo controle do estado desses pinos na plataforma.

Assim, utilizando-se no plano de discagem a aplicação *System()*, os comandos *shell* são passados ao sistema Linux, permitindo o controle das GPIO.

Para isso, é necessário que as configurações de permissão para o usuário *asterisk* no sistema Linux estejam habilitadas para que o controle das GPIO aconteça. O procedimento para realizar isso é descrito nessa seção e foi adaptado de GitHub (2014).

Inicialmente, deve-se criar um arquivo no diretório */etc/udev/rules.d*, com o nome *80-gpiobone.rules* e o seguinte conteúdo:

```
1  KERNEL=="gpio*", SUBSYSTEM=="gpio", ACTION=="add", PROGRAM="/bin/fix_bone_udev_gpio.sh"
```

Em seguida, cria-se também um arquivo de nome *fix_bone_udev_gpio.sh* no diretório */bin* com o seguinte conteúdo:

```
1  #!/bin/bash
2  #
3  # This file will change the user, group and permissions in both the
4  # /sys/devices/virtual/gpio and /sys/class/gpio path directories.
5  #
6  # DO NOT change the order of the commands below, they are optimized so that
7  # commonly created files and directories are changed first.
8  #
9
10 chown -R debian:gpio /sys/devices/virtual/gpio
11 chown -R debian:gpio /sys/class/gpio
12 find /sys/devices/virtual/gpio -type d -exec chmod 4775 {} \;
13 find /sys/devices/virtual/gpio -name "direction" -exec chmod 0660 {} \;
14 find /sys/devices/virtual/gpio -name "edge" -exec chmod 0660 {} \;
15 find /sys/devices/virtual/gpio -name "value" -exec chmod 0660 {} \;
16
17 find /sys/devices/virtual/gpio -name "active_low" -exec chmod 0660 {} \;
18 chmod 0220 /sys/class/gpio/export
19 chmod 0220 /sys/class/gpio/unexport
20 find /sys/devices/virtual/gpio -name "uevent" -exec chmod 0660 {} \;
21 find /sys/devices/virtual/gpio -name "autosuspend_delay_ms" -exec chmod 0660 {} \;
22 find /sys/devices/virtual/gpio -name "control" -exec chmod 0660 {} \;
```

Configura-se permissão de execução para o último arquivo criado através do comando:

```
1  # chmod 744 /bin/fix_bone_udev_gpio.sh
```

Cria-se um grupo de nome *gpio*, executando-se o seguinte comando:

```
1  # sudo groupadd gpio
```

E, então, adiciona-se o usuário *asterisk* ao referido grupo, através do comando:

```
1  # adduser asterisk gpio
```

Por fim, reinicia-se o sistema:

```
# reboot
```

Então, as permissões já estão configuradas para o controle das GPIOs na Beagleboard pelo usuário *asterisk*.

Ao ser iniciada a Beaglebone, notou-se que alguns pinos GPIO utilizados permanecem num estado indefinido, nem ligado, nem desligado, apresentando-se uma certa luminosidade nos LEDs do protótipo. Para resolver esse problema, é necessário zerar na inicialização do sistema o estado das GPIO em que isso ocorre.

No arquivo /emph/etc/rc.local adiciona-se o seguinte conteúdo:

```
# P8_8
echo 67 > /sys/class/gpio/export
echo low > /sys/class/gpio/gpio67/direction

# P8_7
echo 66 > /sys/class/gpio/export
echo low > /sys/class/gpio/gpio66/direction

# P8_9
echo 69 > /sys/class/gpio/export
echo low > /sys/class/gpio/gpio69/direction

# P8_10
echo 68 > /sys/class/gpio/export
echo low > /sys/class/gpio/gpio68/direction

# P8_11
echo 45 > /sys/class/gpio/export
echo low > /sys/class/gpio/gpio45/direction

# P8_12
echo 44 > /sys/class/gpio/export
echo low > /sys/class/gpio/gpio44/direction

# P8_13
echo 23 > /sys/class/gpio/export
echo low > /sys/class/gpio/gpio23/direction

# P8_14
echo 26 > /sys/class/gpio/export
echo low > /sys/class/gpio/gpio26/direction

# P8_15
echo 47 > /sys/class/gpio/export
echo low > /sys/class/gpio/gpio47/direction

# P8_16
echo 46 > /sys/class/gpio/export
echo low > /sys/class/gpio/gpio46/direction

# P8_17
echo 27 > /sys/class/gpio/export
echo low > /sys/class/gpio/gpio27/direction

```

```
45        # P8_18
46        echo 65 > /sys/class/gpio/export
47        echo low > /sys/class/gpio/gpio65/direction
48
49        # P8_19
50        echo 22 > /sys/class/gpio/export
51        echo low > /sys/class/gpio/gpio22/direction
52
53        # P8_26
54        echo 61 > /sys/class/gpio/export
55        echo low > /sys/class/gpio/gpio61/direction
56
57        # P9_12
58        echo 60 > /sys/class/gpio/export
59        echo low > /sys/class/gpio/gpio60/direction
```

APÊNDICE B – Tutorial de instalação e utilização do HTK

Adaptado dos tutoriais da Voxforge (VOXFORGE, 2016), bem como do HTK Book (YOUNG et al., 2009).

Para a obtenção e utilização do HTK é necessário registrar-se no *site de registro* e concordar com os termos de licença.

Apesar de haverem limitações para a distribuição do HTK Toolkit, o *software* é de código aberto e não implica em nenhuma limitação para a distribuição dos modelos criados com a utilização do *toolkit*.

A seguir, será ilustrado um passo a passo para a instalação do pacote HTK *Toolkit* versão 3.4.1, num sistema Linux Ubuntu 14.04.2 LTS *Desktop*, com arquitetura de 64 bits (amd64).

B.1 Instalando o HTK

B.1.1 Obtendo os pacotes

Primeiramente, deve ser criado um diretório de nome *bin* na pasta do usuário do sistema, conforme o comando a seguir (é importante lembrar de substituir *nomedousuario* pelo nome de seu usuário no sistema utilizado):

```
1  $ mkdir /home/nomedousuario/bin
```

Após ter sido realizado o registro no *site* do HTK, conforme mencionado anteriormente, deve-se acessar a página de downloads para se obter os pacotes *HTK* Toolkit e Samples, devendo serem salvos no diretório *bin* criado na pasta *home* do usuário.

Também é recomendável realizar o *download* do *HTK* Book, pois ele apresenta uma excelente referência para os comandos do *toolkit*.

B.1.2 Descompactando os arquivos

Os arquivos obtidos podem ser descompactados diretamente no gerenciador de arquivos *Nautilus* do Linux, clicando-se com o botão direito do *mouse* em cada arquivo *.tar.gz* e selecionando a opção *Extrair aqui*; ou através dos seguintes comandos no terminal:

```
1  $ tar –xzvf HTK–3.4.1.tar.gz
2  $ tar –xzvf HTK–samples–3.4.1.tar.gz
```

Com isso, na pasta *bin* devem ter sido criadas as subpastas *htk* e *samples*. Então, em seguida, move-se (recorte e cole) a subpasta *samples* para o interior da subpasta *htk*.

B.1.3 Compilando e instalando o HTK

A instalação do HTK requer que o sistema operacional possua o compilador *gcc* instalado. Para verificar se o *gcc* está instalado em seu computador, executa-se o seguinte comando no terminal:

```
1 $ dpkg -l | grep gcc
```

A Figura 39, a seguir, ilustra a saída do comando *dpkg*, de modo que as letras *ii* que aparecem no início de cada linha indicam um pacote instalado no sistema.

Figura 39 – Saída do comando *dpkg*.

```
ii  gcc                                  4:4.8.2-1ubuntu6                amd64        GNU C compiler
ii  gcc-4.6                              4.6.4-6ubuntu2                  amd64        GNU C compiler
ii  gcc-4.6-base:amd64                   4.6.4-6ubuntu2                  amd64        GCC, the GNU Compiler Collection (base package)
ii  gcc-4.8                              4.8.2-19ubuntu1                 amd64        GNU C compiler
ii  gcc-4.8-base:amd64                   4.8.2-19ubuntu1                 amd64        GCC, the GNU Compiler Collection (base package)
ii  gcc-4.9-base:amd64                   4.9.1-0ubuntu1                  amd64        GCC, the GNU Compiler Collection (base package)
ii  lib32gcc1                            1:4.9.1-0ubuntu1                amd64        GCC support library (32 bit Version)
ii  libgcc-4.8-dev:amd64                 4.8.2-19ubuntu1                 amd64        GCC support library (development files)
ii  libgcc1:amd64                        1:4.9.1-0ubuntu1                amd64        GCC support library
```

Fonte: Autoria própria

Caso o *gcc* não esteja instalado, deve ser instalada a versão mais nova. Para verificar o nome do pacote com a versão mais recente do gcc, pode-se inicialmente realizar uma busca com a ferramenta *apt* através do seguinte comando:

```
1 $ apt-cache search gcc | grep compilador
```

Esse comando irá mostrar uma lista reduzida de pacotes, onde será mais facilmente encontrado o pacote do compilador desejado. Após isso, escolhe-se o nome do pacote com a versão mais recente e realiza-se sua instalação com o seguinte comando:

```
1 $ sudo apt-get install gcc-4.8
```

É importante notar que, no caso, a versão mais nova é a 4.8. Voxforge (2016) atenta para o fato de que, caso sua distribuição Linux possua o *gcc* em uma versão superior ou igual à versão 4, será necessário instalar os módulos de compatibilidade para a versão 3.4 do *gcc* para que o HTK seja compilado adequadamente. No entanto, o FAQ (*Frequently Asked Questions*) do HTK (CUED, 2015) afirma que a partir da versão 3.4, o HTK é compatível com a versão 4 do *gcc*. O *apt* irá instalar o pacote resolvendo as dependências que sejam necessárias.

Agora, antes de iniciar a compilação, deve-se instalar o pacote *libx11-dev*, que fornecerá algumas bibliotecas adicionais necessárias para compatibilidade com sistemas de 32 bits (i386). Essa instalação é de suma importância para o sucesso do processo de compilação e instalação do HTK no sistema. A instalação do pacote deve ser realizada através do comando a seguir:

```
$ sudo apt-get install libx11-dev
```

Feito isso, deve-se acessar o modo *super usuário* no terminal através do comando:

```
$ sudo su
```

Já no modo *super usuário*, o sistema está agora preparado para executar os comandos de compilação e instalação da ferramenta HTK. Então, com o terminal localizado na pasta */home/numedousuario/bin/htk*, devem ser executados os seguintes comandos:

```
# ./configure --prefix=/home/nomedousuario/bin/htk
# make all
# make install
```

Finalizado esse processo, é necessário ainda atualizar a variável de ambiente *PATH* do sistema linux para indicar o caminho onde o *shell* deve buscar os comandos da *toolbox* HTK quando estes forem executados.

Para que esses caminhos sejam adicionados ao *PATH* do sistema sempre na inicialização, deve-se alterar o arquivo de configuração */etc/profile*, acrescentando após os comentários do início do arquivo a linha:

PATH=$PATH:/home/nomedousuario/bin:/home/nomedousuario/bin/htk/bin

Com isso, o referido arquivo terá uma forma semelhante ao ilustrado na Figura 40, a seguir.

Feito isso, é necessário agora reiniciar a sessão do usuário para que as configurações tomem efeito.

Após ter sido realizada a reinicialização da sessão do usuário, deve-se testar o funcionamento da instalação do HTK através do seguinte comando:

```
$ HVite -V
```

Caso a instalação tenha sido realizada com sucesso, a saída do comando anterior não deve apresentar nenhuma mensagem de erro, tendo a forma ilustrada na Figura 41, a seguir.

Com isso, a instalação foi realizada com sucesso! A *toolbox* HTK já pode ser utilizada!

Figura 40 – Arquivo */etc/profile*.

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

PATH=$PATH:/home/nomedousuario/bin:/home/nomedousuario/bin/htk/bin

if [ "$PS1" ]; then
  if [ "$BASH" ] && [ "$BASH" != "/bin/sh" ]; then
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi

# The default umask is now handled by pam_umask.
# See pam_umask(8) and /etc/login.defs.

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi
```

Fonte: Autoria própria

B.2 Criando modelo acústico com o HTK

Esse tutorial se baseia no *How-to*[1] e no *Tutorial*[2] do projeto Voxforge para a criação de modelo acústico para o *engine* Julius, a partir de *script*, utilizando o HTK. Essa abordagem segue as orientações descritas no HTK *Book*[3] , mas utiliza um *script* para automatizar a maioria dos passos. É mais rápida e não requer profundo conhecimento sobre o funcionamento das ferramentas do pacote HTK. Com ela, foi possível a criação de modelo acústico baseado em gramática no idioma inglês.

Presume-se que os *softwares* necessários já estejam devidamente instalados e configurados no sistema.

[1] VOXFORGE. *How-to*: Create acoustic model - with script. Disponível em: <http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/how-to>. Acesso em: 03 jul. 2015.

[2] VOXFORGE. *Tutorial*: Create Acoustic Model - Manually. 2016. Disponível em: <http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/tutorial>. Acesso em: 03 jul. 2015.

[3] YOUNG, S. et al. *The HTK Book*. Cambridge: Cambridge Univertsity Engineering Department, 2009. 384 p.

Figura 41 – Saída esperada para o comando *HVite -V*.

```
HTK Version Information
Module      Version    Who    Date      : CVS Info
HVite       3.4.1      CUED   12/03/09  : $Id: HVite.c,v 1.1.1.1 2006/10/11 09:55:02 jal58 Exp $
HShell      3.4.1      CUED   12/03/09  : $Id: HShell.c,v 1.1.1.1 2006/10/11 09:54:58 jal58 Exp $
HMem        3.4.1      CUED   12/03/09  : $Id: HMem.c,v 1.1.1.1 2006/10/11 09:54:58 jal58 Exp $
HLabel      3.4.1      CUED   12/03/09  : $Id: HLabel.c,v 1.1.1.1 2006/10/11 09:54:57 jal58 Exp $
HMath       3.4.1      CUED   12/03/09  : $Id: HMath.c,v 1.1.1.1 2006/10/11 09:54:58 jal58 Exp $
HSigP       3.4.1      CUED   12/03/09  : $Id: HSigP.c,v 1.1.1.1 2006/10/11 09:54:58 jal58 Exp $
HWave       3.4.1      CUED   12/03/09  : $Id: HWave.c,v 1.1.1.1 2006/10/11 09:54:59 jal58 Exp $
HAudio      3.4.1      CUED   12/03/09  : $Id: HAudio.c,v 1.1.1.1 2006/10/11 09:54:57 jal58 Exp $
HVQ         3.4.1      CUED   12/03/09  : $Id: HVQ.c,v 1.1.1.1 2006/10/11 09:54:59 jal58 Exp $
HModel      3.4.1      CUED   12/03/09  : $Id: HModel.c,v 1.2 2006/12/07 11:09:08 mjfg Exp $
HParm       3.4.1      CUED   12/03/09  : $Id: HParm.c,v 1.1.1.1 2006/10/11 09:54:58 jal58 Exp $
HDict       3.4.1      CUED   12/03/09  : $Id: HDict.c,v 1.1.1.1 2006/10/11 09:54:57 jal58 Exp $
HNet        3.4.1      CUED   12/03/09  : $Id: HNet.c,v 1.1.1.1 2006/10/11 09:54:58 jal58 Exp $
HRec        3.4.1      CUED   12/03/09  : $Id: HRec.c,v 1.1.1.1 2006/10/11 09:54:58 jal58 Exp $
HUtil       3.4.1      CUED   12/03/09  : $Id: HUtil.c,v 1.1.1.1 2006/10/11 09:54:59 jal58 Exp $
HAdapt      3.4.1      CUED   12/03/09  : $Id: HAdapt.c,v 1.2 2006/12/07 11:09:07 mjfg Exp $
HMap        3.4.1      CUED   12/03/09  : $Id: HMap.c,v 1.1.1.1 2006/10/11 09:54:57 jal58 Exp $
```

Fonte: Autoria própria

B.2.1 Preparação dos dados

B.2.1.1 Passo 1 - *Task Grammar*

Deve-se criar um novo subdiretório na pasta *home* do usuário, podendo-se utilizar qualquer nome desejado. Já no interior desse último criado, cria-se um novo subdiretório, que deve ter o nome *auto*.

```
1  $ mkdir /home/nomedousuario/novodiretorio/auto
```

Então, com base nos comandos que se deseja, deve-se criar na pasta *auto* um arquivo de texto, com o nome *sample.voca*. Esse arquivo deve conter a definição das palavras, dividindo-as em categorias específicas, conforme o exemplo abaixo:

```
1   % NS_B
2   <s> sil
3
4   % NS_E
5   </s> sil
6
7   % AMB
8   BATHROOM  b ae th r uw m
9   CHAMBER  ch ey m b er
10  EXTERNAL  ih k s t er n ah l
11  HALL  hh ao l
12  KITCHEN  k ih ch ah n
13
14  % DISP
15  LIGHT  l ay t
16  TV  t iy v iy
17
18  % ACES
19  DOOR  d ao r
20
21  % CONTR
```

```
22      ON    aa n
23      OFF   ao f
24
25      % PERM
26      OPEN   ow p ah n
27      CLOSE  k l ow s
```

As categorias de palavras são especificadas nas linhas com o caractere "%", de modo que abaixo da definição da categoria devem ser definidas suas respectivas palavras, uma em cada linha. A primeira coluna se refere à própria palavra que será gerada na saída do reconhecimento, e o restante da linha apresenta a pronúncia (fonemas) dessa palavra.

Os fonemas para as palavras do vocabulário utilizado nesse exemplo foram determinados utilizando-se a ferramenta CMUdict[4] .

No arquivo *sample.voca*, os caracteres de tabulação e espaço em branco são campos de separação. No exemplo, as categorias *NS_B* e *NS_E* apresentam uma única entrada com a palavra representando o modelo de silêncio, que correspondem ao silêncio no início e no final das locuções de entrada.

Uma vez criado o arquivo com a lista de palavras divididas em categorias, é conveniente que se defina as conexões permitidas entre as palavras. Isso será feito criando-se outro arquivo, com o nome *sample.grammar* no mesmo diretório *auto*, conforme o seguinte exemplo:

```
1       S: NS_B SENT NS_E
2       SENT: AMB DISP CONTR
3       SENT: AMB ACES PERM
```

No arquivo *sample.grammar*, o símbolo "*S*"deve ser empregado para o início das sentenças. As chamadas "regras de reescrita"(*rewrite rules*), devem ser definidas em cada linha, utilizando-se o sinal de dois pontos (":") como delimitador.

O nome escolhido para as categorias, de acordo com o arquivo *.voca*, será utilizado como símbolo terminal (à direita do sinal de dois pontos). As possíveis conexões entre categorias serão indicadas em cada linha.

Caracteres do alfabeto ASCII, números e sublinhado são permitidos para nomear os símbolos, diferenciando letras maiúsculas de minúsculas (*case sensitive*).

Embora essa sintaxe de gramática permita uma classe livre de contexto, o Julius somente é capaz de manipular classes de expressões regulares, uma vez que utiliza o analisador DFA.

[4] LENZO, K. *The CMU Pronouncing Dictionary*. Carnegie Mellon University. Disponível em: <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>. Acesso em: 17 jul. 2015.

Portanto, é necessário realizar a conversão da gramática para o formato compatível com o Julius. Os arquivos *.voca* e *.grammar* devem ser convertidos em *.dfa* e *.dict*. Isso é feito através do compilador de gramáticas *mkdfa.pl*[5] .

Para tanto, deve-se acessar no terminal de comando a pasta em que encontram-se os arquivos criados e executar o *script*, que é instalado juntamente com o Julius.

```
1 prompt@decomando:~/diretorio/auto$ mkdfa.pl sample
```

A saída da execução do *script* deve ser semelhante à ilustrada na Figura 42. Serão criados, no respectivo diretório, três arquivos: *sample.dfa*, *sample.term* e *sample.dict*.

Figura 42 – Saída da execução do compilador de gramática *mkdfa.pl*.

```
sample.grammar has 3 rules
sample.voca    has 7 categories and 14 words
---
Now parsing grammar file
Now modifying grammar to minimize states[-1]
Now parsing vocabulary file
Now making nondeterministic finite automaton[8/8]
Now making deterministic finite automaton[8/8]
Now making triplet list[8/8]
7 categories, 8 nodes, 8 arcs
-> minimized: 7 nodes, 7 arcs
---
generated: sample.dfa sample.term sample.dict
```

Fonte: Autoria própria.

B.2.1.2 Passo 2 - *Dicionário de pronúncia*

Para que o HTK seja capaz de compilar o áudio de treinamento e suas transcrições num modelo acústico, é requerido um dicionário de pronúncia foneticamente balanceado.

No conteúdo deste devem constar, pelo menos, de 30 a 40 sentenças com 8 a 10 palavras cada. Caso a gramática possua menos palavras que essa quantidade mínima exigida (como é o caso desse tutorial), ou, caso o dicionário não seja foneticamente balanceado (com alguns fonemas ocorrendo somente uma ou duas vezes), então é necessário adicionar palavras de modo a garantir a ocorrência de cada fonema pelo menos 3 a 5 vezes.

Isso é feito somente para que se tenha o mínimo de entradas que permitirão ao HTK compilar – criar um modelo acústico que produza resultados mais consistentes de reconhecimento requer muito mais entradas e, consequentemente, mais áudio da fala correspondente. No entanto, os dados aqui utilizados são suficientes para um reconhecimento eficiente em aplicações de comando.

[5] JULIUS DEVELOPMENT TEAM. *How to write a recognition grammar for Julius*. Disponível em: <http://julius.osdn.jp/en_index.php?q=en_grammar.html>. Acesso em: 17 jul. 2015.

Inicialmente, para a gerar o dicionário de pronúncia, é necessário criar um arquivo, com o nome *prompts.txt*. Ele conterá uma lista de palavras, conforme a Figura 43, que serão gravadas no passo seguinte do tutorial.

Figura 43 – Lista de palavras para gravação das amostra de treinamento (*prompts.txt*).

```
*/sample1 BATHROOM LIGHT ON BATHROOM LIGHT OFF DOOR CLOSE
*/sample2 CHAMBER DOOR OPEN CHAMBER DOOR CLOSE EXTERNAL LIGHT
*/sample3 ON EXTERNAL LIGHT OFF HALL DOOR OPEN CLOSE TV
*/sample4 KITCHEN LIGHT OFF HALL TV ON KITCHEN LIGHT ON
*/sample5 LIGHT ON LIGHT OFF LIGHT ON LIGHT OFF EXTERNAL LIGHT
*/sample6 DOOR OPEN DOOR CLOSE DOOR OPEN DOOR CLOSE CHAMBER TV
*/sample7 HALL LIGHT ON HALL LIGHT OFF TV ON TV OFF
*/sample8 CHAMBER TV ON CHAMBER TV OFF BATHROOM DOOR OPEN
*/sample9 BATHROOM DOOR CLOSE ON OFF EXTERNAL LIGHT ON
*/sample10 HALL TV OFF KITCHEN DOOR OPEN KITCHEN DOOR CLOSE
*/sample11 CHAMBER LIGHT ON CHAMBER LIGHT OFF HALL DOOR CLOSE
*/sample12 EXTERNAL LIGHT ON EXTERNAL LIGHT OFF TV LIGHT
*/sample13 BATHROOM KITCHEN EXTERNAL CHAMBER HALL LIGHT DOOR TV
*/sample14 LIGHT DOOR LIGHT DOOR LIGHT LIGHT DOOR TV DOOR TV
*/sample15 ON OFF OPEN CLOSE ON OFF OPEN CLOSE ON OFF
*/sample16 ABASH RANCHERIAS GROUP UNITIES STREIGHT KITCHEN TALK LAYOUT
*/sample17 LENGTH BATHROOM PATHLESS DEATHBLOW MATH BLUETOOTH THEATER BIRTH
*/sample18 DRAFT CHALDEAN AUNTS CALORIC LATTER BACTERIA CRAMB TRAFFIC
*/sample19 HERTZ ABRAHAM HEALTH ALCOHOL HEART DUNHILL EDENHALL HARM
*/sample20 SCHOOL CHEW BREW SALOON HALFMOON BATHROOM SOON RESTROOM
*/sample21 MATCH CHAMBER PATCH KITCHEN ACHIEVE CHERRY FRENCH CHINA
*/sample22 RAVEN MAIL DISPLAY MAYBE CHAMBER FLAME CHAMPION BRAVE
*/sample23 INFER EXTERNAL MODERN CHAMBER ETERNAL GAMER FRATERNAL REMEMBER
*/sample24 RENAULT HALL ABNORMALLY GLORIOUS DEFAULT FALL PINBALL CUTOFF
*/sample25 FIGHTER BYTE FIND DIOCESAN IONIC PYTHON RIGHTS SITE
*/sample26 MOVIE VILLAGE WOLVES FLAVOR VISIT HEAVEN VOLT VIDEO
*/sample27 ARROW OWNER OPEN ANTILOPE CLOSE PROMOTION THRONE REMOTE
*/sample28 GRAZE PROPHET WATER SENSOR ACTUATOR ELECTRIC OPEN RAY
*/sample29 ORANGE MASTER SYMBOL GRAIL SPEECH GARAGE LOCK HARM
*/sample30 LIVING PROUD FRIEND PRAY SHUTTER GARDEN OUTSIDE KITCHEN
*/sample31 BIKE HOME COUNTRY CITY MOTHER BROTHER SISTER BATHROOM
*/sample32 BUSBOYS CHOICE COILS COIN ENJOY TOYSTORE PARANOID ANDROID
*/sample33 BASS WOOD MUSIC HARMONY MOUTH EARS HEAR SPEAK
*/sample34 BEDROOM EXTERNAL KEYBOARD WINDOW DATA GERMANY WISDOM RULE
*/sample35 FREEDOM SPORTS BOOKS KITCHEN EJECT LEFT ROUND BEATLE
*/sample36 PEPPER FLIGHT BATHROOM PARTY CLOTHES FUEL WORK BATTLE
*/sample37 DESK HALL FLOOR WALL EXTERNAL KITCHEN BATHROOM BOX
*/sample38 BURGER TONGUE HAND HOLD BROAD ROAD THEATER SOCCER
*/sample39 TOWN TRAVEL DREAM GOLDEN BEACH SUNRISE SAND HOPE
*/sample40 SPEECH RECOGNITION EMBEDDED SYSTEM BEAGLE ASTERISK AUTOMATION
```

Fonte: Autoria própria.

A primeira coluna da lista *prompts* devem conter os nomes dos arquivos de áudio de treinamento que serão criados. Já as colunas seguintes de cada linha, devem conter as transcrições das palavras que serão gravadas no respectivo arquivo de áudio a que se refere.

Esse novo arquivo será utilizado pelo *script* de criação do modelo acústico para gerar outra lista de palavras (*wlist*). Esta usada na criação do dicionário de pronúncia.

Antes de seguir ao próximo passo, o dicionário fonético *VoxforgeDict.txt*[6] deve ser salvo num novo subdiretório, de nome *lexicon* na pasta *auto*.

[6] VOXFORGE. *VoxforgeDict.txt*. Disponível em: <http://www.voxforge.org/uploads/2D/Oe/2DOeE3hM6livfkY6yxM97g/VoxForgeDict.txt>. Acesso em: 18 jul. 2015.

B.2.1.3 Passo 3 - Gravando os dados

Para realizar a gravação das amostras de treinamento de áudio, pode-se utilizar o *software* Audacity. No entanto, para o caso desse projeto, é desejável que as amostras sejam gravadas a partir de uma chamada a ramal do servidor Asterisk, utilizando-se o *softphone* instalado num *smartphone*.

Isso é necessário para que o modelo criado tenha as características do canal do Asterisk, de modo a se alcançar um reconhecimento eficiente.

Os arquivos devem ser salvos em um novo subdiretório, no mesmo nível da pasta *auto*, ficando o caminho final como */home/usuario/diretorio/train/wav*. Para realizar as gravações, deve se basear no arquivo *prompts.txt* (na primeira coluna o nome do arquivo de áudio e demais colunas o texto transcrito).

As gravações devem ser realizadas em ambiente silencioso. Uma boa recomendação é desligar os auto-falantes do PC para evitar *feedback* acústico nos arquivos.

O microfone deve estar ajustado corretamente. Para um ajuste ótimo de captação, é interessante utilizar um *headset* para a gravação. O microfone deve estar um ou dois centímetros de distância da boca em direção ao lado e para abaixo (evitando captar a respiração do locutor).

Para o caso de se utilizar o Audacity, seus níveis de gravação devem ser testados antes de se iniciar o processo. O volume de captação deve ser ajustado de forma que os níveis se mantenham entre 0.5 e -0.5, estando com um nível de 0.3 a -0.3 em média, conforme ilustra a Figura 44.

É normal a ocorrência de alguns picos fora da faixa de 0.5 a -0.5. No entanto, deve-se evitar picos que ultrapassem a faixa de 1.0 a -1.0, o que acabará gerando distorção.

O projeto Voxforge recomenda que se utilize a capacidade máxima da placa de áudio, configurando-se a máxima taxa de amostragem que ela suporta, para a gravação de amostras de treinamento. Posteriormente pode ser realizada uma re-amostragem de modo a adequar-se o sinal para a taxa de amostragem desejada.

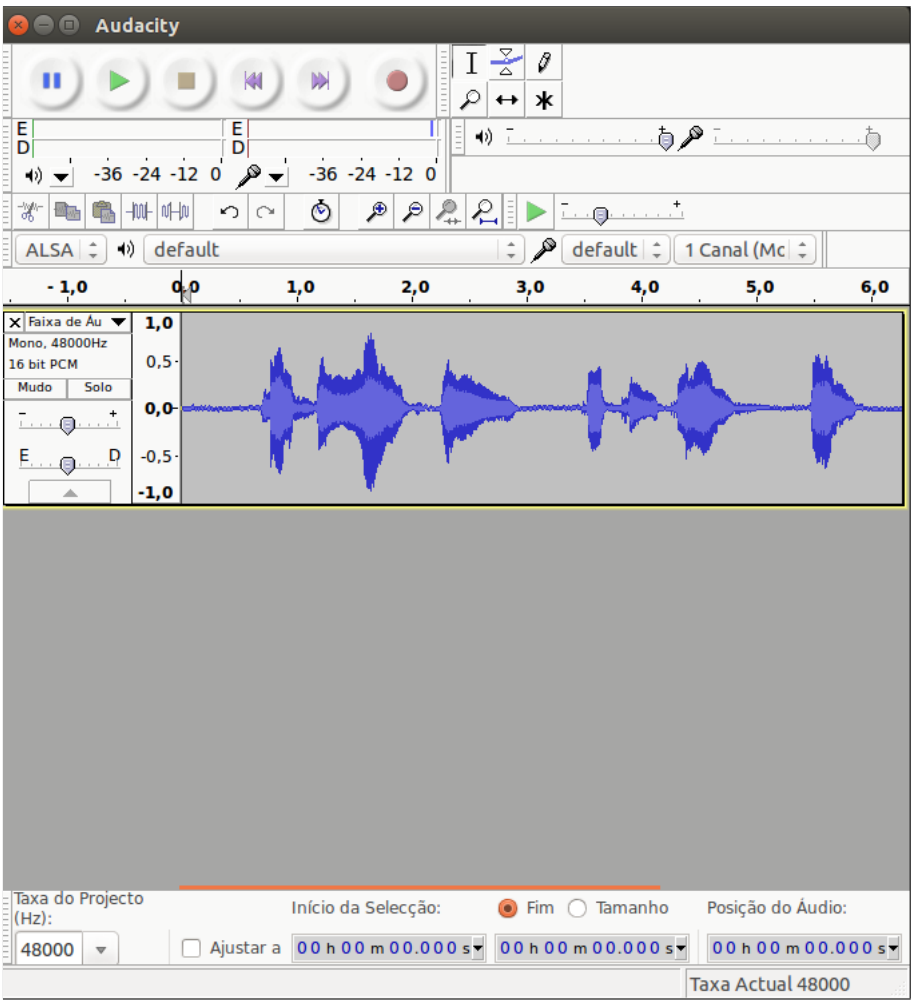
Inicialmente, utilizou-se a taxa de 48kHz, com 16 bits por amostra, em canal mono. Ao exportar o áudio, recomenda-se o formato WAV (Microsoft 16 bit PCM).

No caso, para a gravação a partir do Asterisk, pode-se utilizar a aplicação *Record()* no plano de discagem. Os arquivos podem ser posteriormente importados no Audacity para averiguação dos níveis de captação. A taxa de amostragem utilizada pelo Asterisk é de 8kHz, com 16 bits por amostra e canal mono.

O arquivo *prompts.txt* criado no passo anterior deve servir de guia para as locuções que devem ser gravadas. Cada linha do arquivo de texto corresponde ao conteúdo transcrito de um arquivo de áudio, conforme mencionado anteriormente.

Com tudo pronto, deve-se iniciar a gravação no Audacity, pronunciando-se as palavras de primeira linha do arquivo *prompts*:

Figura 44 – Níveis de áudio no Audacity.



Fonte: Autoria própria.

```
1 BATHROOM LIGHT ON BATHROOM LIGHT OFF DOOR CLOSE
```

Deve-se procurar falar normalmente - nem tão rápido, nem tão devagar - e claramente. Faz-se uma breve pausa antes de pronunciar a próxima palavra (de cerca de meio segundo). Tomar cuidado para evitar a respiração nas pausas.

Realizada a gravação, exporte o arquivo em formato *.wav*, salvando no diretório criado. Cada arquivo deve ter o nome correspondente a cada linha do *prompts* (*sample*1, *sample*2,...).

B.2.1.4 Passo 4 - Criar *script* de treinamento do HTK

Agora, deve-se criar, na pasta *auto* um arquivo de texto, a ser nomeado de *code-train.scp*, contendo a informação da localização das amostras de treinamento, bem como

indicando o caminho onde devem ser salvos os arquivos convertidos no formato *.mfc*. A Figura 45[7] apresenta os caminhos, relativos à pasta *auto*, utilizados nesse exemplo.

Figura 45 – *Script codetrain.scp*.

```
../train/wav/8k/sample1.wav ../train/mfcc/8k/sample1.mfc
../train/wav/8k/sample2.wav ../train/mfcc/8k/sample2.mfc
../train/wav/8k/sample3.wav ../train/mfcc/8k/sample3.mfc
../train/wav/8k/sample4.wav ../train/mfcc/8k/sample4.mfc
../train/wav/8k/sample5.wav ../train/mfcc/8k/sample5.mfc
../train/wav/8k/sample6.wav ../train/mfcc/8k/sample6.mfc
../train/wav/8k/sample7.wav ../train/mfcc/8k/sample7.mfc
../train/wav/8k/sample8.wav ../train/mfcc/8k/sample8.mfc
../train/wav/8k/sample9.wav ../train/mfcc/8k/sample9.mfc
../train/wav/8k/sample10.wav ../train/mfcc/8k/sample10.mfc
../train/wav/8k/sample11.wav ../train/mfcc/8k/sample11.mfc
../train/wav/8k/sample12.wav ../train/mfcc/8k/sample12.mfc
../train/wav/8k/sample13.wav ../train/mfcc/8k/sample13.mfc
../train/wav/8k/sample14.wav ../train/mfcc/8k/sample14.mfc
../train/wav/8k/sample15.wav ../train/mfcc/8k/sample15.mfc
../train/wav/8k/sample16.wav ../train/mfcc/8k/sample16.mfc
../train/wav/8k/sample17.wav ../train/mfcc/8k/sample17.mfc
../train/wav/8k/sample18.wav ../train/mfcc/8k/sample18.mfc
../train/wav/8k/sample19.wav ../train/mfcc/8k/sample19.mfc
../train/wav/8k/sample20.wav ../train/mfcc/8k/sample20.mfc
../train/wav/8k/sample21.wav ../train/mfcc/8k/sample21.mfc
../train/wav/8k/sample22.wav ../train/mfcc/8k/sample22.mfc
../train/wav/8k/sample23.wav ../train/mfcc/8k/sample23.mfc
../train/wav/8k/sample24.wav ../train/mfcc/8k/sample24.mfc
../train/wav/8k/sample25.wav ../train/mfcc/8k/sample25.mfc
../train/wav/8k/sample26.wav ../train/mfcc/8k/sample26.mfc
../train/wav/8k/sample27.wav ../train/mfcc/8k/sample27.mfc
../train/wav/8k/sample28.wav ../train/mfcc/8k/sample28.mfc
../train/wav/8k/sample29.wav ../train/mfcc/8k/sample29.mfc
../train/wav/8k/sample30.wav ../train/mfcc/8k/sample30.mfc
../train/wav/8k/sample31.wav ../train/mfcc/8k/sample31.mfc
../train/wav/8k/sample32.wav ../train/mfcc/8k/sample32.mfc
../train/wav/8k/sample33.wav ../train/mfcc/8k/sample33.mfc
../train/wav/8k/sample34.wav ../train/mfcc/8k/sample34.mfc
../train/wav/8k/sample35.wav ../train/mfcc/8k/sample35.mfc
../train/wav/8k/sample36.wav ../train/mfcc/8k/sample36.mfc
../train/wav/8k/sample37.wav ../train/mfcc/8k/sample37.mfc
../train/wav/8k/sample38.wav ../train/mfcc/8k/sample38.mfc
../train/wav/8k/sample39.wav ../train/mfcc/8k/sample39.mfc
../train/wav/8k/sample40.wav ../train/mfcc/8k/sample40.mfc
```

Fonte: Autoria própria.

B.2.2 Executando *script* de criação do modelo acústico

Cria-se, no mesmo nível do diretório *auto*, uma pasta com o nome *bin*. Nela, devem ser salvos os seguintes *scripts* disponibilizados pelo projeto voxforge[8] :

- fixfulllist.jl
- mkclscript.jl

[7] VOXFORGE. *Codetrain.scp*. Disponível em: <https://raw.githubusercontent.com/VoxForge/develop/master/howto/codetrain.scp>. Acesso em: 18 jul. 2015.

[8] VOXFORGE. Scripts. Disponível em: <http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/how-to/script>. Acesso em: 18 jul. 2015.

- mktrihed.jl
- prompts2mlf.jl
- prompts2wlist.jl
- trainAM.jl

Em seguida, cria-se um subdiretório nomeado *input_files*, no interior da pasta *auto*, para salvar os seguintes arquivos:

- config
- mkphones0.led
- mktri.led
- sil.hed
- wav_config
- global.ded
- mkphones1.led
- proto
- tree1.hed

Todos esses arquivos estão disponíveis na referência citada ao rodapé de número 8.

Recentemente, a *toolkit* do projeto Voxforge para a criação de modelos acústicos foi reescrita na linguagem Julia. Assim, é necessário realizar a instalação desse pacote.

```
1 # apt-get install julia
```

Após isso, acesse a pasta *auto* e, em seguida, execute o *script trainAM.jl*, indicando o caminho relativo à pasta *auto*.

```
1 # julia ../bin/trainAM.jl
```

Caso esteja tudo configurado corretamente, conforme exposto no tutorial, devem ter sido criados um novo diretório no interior da pasta *auto*, denominado *acoustic_model_files*. Nesse novo subdiretório se encontrarão os arquivos do modelo acústico, que são o *hmmdefs* e o *tiedlist*.

Para realizar-se reconhecimento com o Julius utilizando o modelo criado, esses dois últimos arquivos, bem como os arquivos *sample.dfa* e *sample.dict*, devem ser especificados no arquivo de configuração do *engine*.

Anexos

ANEXO A – Código desenvolvido em linguagem Python para programação da interface baseada em módulo de *hardware*

```
#!/usr/bin/python

# Instituto Federal de Educacao, Ciencia e Tecnologia da Paraiba
# Programa de Pos-Graduacao em Engenharia Eletrica
# Linha de Pesquisa em Processamento Digital de Sinais

# Código para controle das luzes de LED RGB com Voice Recognitino Module V2

import Adafruit_BBIO.GPIO as GPIO
import time
import Adafruit_BBIO.UART as UART
import serial

GPIO.setup("P9_11", GPIO.IN)
GPIO.setup("P9_12", GPIO.IN)
GPIO.setup("P9_13", GPIO.IN)
GPIO.setup("P9_15", GPIO.IN)
GPIO.setup("P9_16", GPIO.IN)
GPIO.setup("P8_7", GPIO.OUT)
GPIO.setup("P8_9", GPIO.OUT)
GPIO.setup("P8_11", GPIO.OUT)

UART.setup("UART2")

ser = serial.Serial(port = "/dev/ttyO2", baudrate=9600)
ser.close()
ser.open()

ser.write(bytearray([0xAA, 0x21]))
ser.write(bytearray([0xAA, 0x33]))
ser.write(bytearray([0xAA, 0x46]))

rubro = "P8_7"
verde = "P8_9"
azul = "P8_11"

GPIO.output(rubro, GPIO.LOW)
GPIO.output(verde, GPIO.LOW)
GPIO.output(azul, GPIO.LOW)

while True:
        r = GPIO.input("P9_11")
        g = GPIO.input("P9_12")
        b = GPIO.input("P9_13")
        br = GPIO.input("P9_15")
        a = GPIO.input("P9_16")

        if r == 1 and a == 0:
                GPIO.output(rubro, GPIO.HIGH)
                time.sleep(0.2)
                ser.write(bytearray([0xAA, 0x46]))
        elif g == 1 and a == 0:
                GPIO.output(verde, GPIO.HIGH)
                time.sleep(0.2)
                ser.write(bytearray([0xAA, 0x46]))
        elif b == 1 and a == 0:
                GPIO.output(azul, GPIO.HIGH)
                time.sleep(0.2)
                ser.write(bytearray([0xAA, 0x46]))
        elif br == 1 and a == 0:
                GPIO.output(rubro, GPIO.HIGH)
                GPIO.output(verde, GPIO.HIGH)
                GPIO.output(azul, GPIO.HIGH)
                time.sleep(0.2)
                ser.write(bytearray([0xAA, 0x46]))
        elif a == 1:
                GPIO.output(rubro, GPIO.LOW)
                GPIO.output(verde, GPIO.LOW)
                GPIO.output(azul, GPIO.LOW)
                time.sleep(0.2)
                ser.write(bytearray([0xAA, 0x46]))
```

ANEXO B – Planilhas com dados dos testes da interface baseada em módulo de *hardware*.

Locutor 1 (voz feminina)				
	Acerto	Não Acendeu	Falso Positivo	%acerto
Rubro	17	2	1	85
Verde	20	0	0	100
Azul	20	0	0	100
Branco	20	0	0	100
Apaga	19	1	0	95
			Taxa de acerto	96

Locutor 2 (voz feminina)				
	Acerto	Não Acendeu	Falso Positivo	%acerto
Rubro	19	1	0	95
Verde	19	1	0	95
Azul	20	0	0	100
Branco	20	0	0	100
Apaga	19	1	0	95
			Taxa de acerto	97

Locutor 3 (voz feminina)				
	Acerto	Não Acendeu	Falso Positivo	%acerto
Rubro	20	0	0	100
Verde	20	0	0	100
Azul	20	0	0	100
Branco	18	2	0	90
Apaga	20	0	0	100
			Taxa de acerto	98

Locutor 4 (voz feminina)				
	Acerto	Não Acendeu	Falso Positivo	%acerto
Rubro	20	0	0	100
Verde	15	5	0	75
Azul	20	0	0	100
Branco	17	3	0	85
Apaga	18	2	0	90
			Taxa de acerto	90

Locutor 5 (voz masculina)				
	Acerto	Não Acendeu	Falso Positivo	%acerto
Rubro	20	0	0	100
Verde	20	0	0	100
Azul	20	0	0	100
Branco	20	0	0	100
Apaga	17	3	0	85
			Taxa de acerto	97

Locutor 6 (voz masculina)				
	Acerto	Não Acendeu	Falso Positivo	%acerto
Rubro	18	2	0	90
Verde	20	0	0	100
Azul	20	0	0	100
Branco	17	3	0	85
Apaga	17	3	0	85
			Taxa de acerto	92

Locutor 7 (voz masculina)				
	Acerto	Não Acendeu	Falso Positivo	%acerto
Rubro	20	0	0	100
Verde	20	0	0	100
Azul	18	2	0	90
Branco	20	0	0	100
Apaga	19	0	1	95
			Taxa de acerto	97

Locutor 8 (voz masculina)				
	Acerto	Não Acendeu	Falso Positivo	%acerto
Rubro	20	0	0	100
Verde	15	0	5	75
Azul	20	0	0	100
Branco	20	0	0	100
Apaga	20	0	0	100
			Taxa de acerto	95

Locutor 9 (voz feminina)				
	Acerto	Não Acendeu	Falso Positivo	%acerto
Rubro	20	0	0	100
Verde	20	0	0	100
Azul	20	0	0	100
Branco	19	1	0	95
Apaga	18	2	0	90
			Taxa de acerto	97

Locutor 10 (voz masculina)				
	Acerto	Não Acendeu	Falso Positivo	%acerto
Rubro	20	0	0	100
Verde	20	0	0	100
Azul	20	0	0	100
Branco	20	0	0	100
Apaga	20	0	0	100
			Taxa de acerto	100

ANEXO C – Código desenvolvido em linguagem Python para programação da interface baseada em ferramentas de *software* integradas a servidor de comunicação

```
#!/usr/bin/python

# Instituto Federal de Educacao, Ciencia e Tecnologia da Paraiba
# Programa de Pos Graduacao em Engenharia Eletrica
# Linha de Pesquisa em Processamento Digital de Sinais

# Extended AGI para reconhecimento de voz com o Julius

# Importando os modulos necessarios
import sys, os, subprocess, re

# Definindo numero do descritor de arquivo
ndes = 3

# Lendo as variaveis passadas pelo Asterisk para o script
while 1:
        line = sys.stdin.readline().strip()
        if line == '':
                break

# Abrindo descritor de arquivo
audio = os.fdopen(ndes,'rb')

# Escrevendo 5 segundos de audio em arquivo raw
with open('/home/debian/projetoraft/audio/comando.raw', 'wb') as out_file:
        out_file.write(audio.read(2*8000*5))

# Convertendo arquivo raw para wav
subprocess.call(['sox', '-t', 'raw', '-r', '8000', '-b', '16', '-e', 'signed-integer', '-c', '1',
                '/home/debian/projetoraft/audio/comando.raw', '/home/debian/projetoraft/audio/
comando.wav'])

# Executando Julius passando o comando de voz como entrada
resjulius = subprocess.Popen(['julius', '-C', '/home/debian/projetoraft/julius.jconf'],
                                stdin=subprocess.PIPE, stdout=subprocess.PIPE)

# Lendo as linhas de saida do Julius
while 1:
        saida = resjulius.stdout.readline()
        if saida == '':
                break
        sys.stderr.write(saida)
        sys.stderr.flush()
        comando = re.search('(?<=> )(.*?)(?= <)', saida)
        if comando is not None:
                if comando.group() == 'EXTERNAL LIGHT ON':
                        # DEFINE OUTDOOR
                        sys.stdout.write('SET VARIABLE OUT 1\n')
                        sys.stdout.flush()
                        # P8_8 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO0
                                        "echo 67 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_7 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO1
                                        "echo 66 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_16 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO2
                                        "echo 46 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P9_12 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO3
                                        "echo 60 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_26 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO4
                                        "echo 61 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_8 ON
                        sys.stdout.write("""SET VARIABLE ESTADO0
                                        "echo high > /sys/class/gpio/gpio67/direction"\n""")
                        sys.stdout.flush()
                        # P8_7 ON
```

```
                        sys.stdout.write("""SET VARIABLE ESTADO1
                                        "echo high > /sys/class/gpio/gpio66/direction"\n""")
                        sys.stdout.flush()
                        # P8_16 ON
                        sys.stdout.write("""SET VARIABLE ESTADO2
                                        "echo high > /sys/class/gpio/gpio46/direction"\n""")
                        sys.stdout.flush()
                        # P9_12 ON
                        sys.stdout.write("""SET VARIABLE ESTADO3
                                        "echo high > /sys/class/gpio/gpio60/direction"\n""")
                        sys.stdout.flush()
                        # P8_26 ON
                        sys.stdout.write("""SET VARIABLE ESTADO4
                                        "echo high > /sys/class/gpio/gpio61/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'EXTERNAL LIGHT OFF':
                        # DEFINE OUTDOOR
                        sys.stdout.write('SET VARIABLE OUT 1\n')
                        sys.stdout.flush()
                        # P8_8 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO0
                                        "echo 67 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_7 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO1
                                        "echo 66 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_16 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO2
                                        "echo 46 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P9_12 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO3
                                        "echo 60 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_26 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO4
                                        "echo 61 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_8 OFF
                        sys.stdout.write("""SET VARIABLE ESTADO0
                                        "echo low > /sys/class/gpio/gpio67/direction"\n""")
                        sys.stdout.flush()
                        # P8_7 OFF
                        sys.stdout.write("""SET VARIABLE ESTADO1
                                        "echo low > /sys/class/gpio/gpio66/direction"\n""")
                        sys.stdout.flush()
                        # P8_16 OFF
                        sys.stdout.write("""SET VARIABLE ESTADO2
                                        "echo low > /sys/class/gpio/gpio46/direction"\n""")
                        sys.stdout.flush()
                        # P9_12 OFF
                        sys.stdout.write("""SET VARIABLE ESTADO3
                                        "echo low > /sys/class/gpio/gpio60/direction"\n""")
                        sys.stdout.flush()
                        # P8_26 OFF
                        sys.stdout.write("""SET VARIABLE ESTADO4
                                        "echo low > /sys/class/gpio/gpio61/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'KITCHEN DOOR OPEN':
                        # P8_13 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 23 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_13 ON
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo high > /sys/class/gpio/gpio23/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'KITCHEN DOOR CLOSE':
                        # P8_13 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 23 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_13 OFF
```

```
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo low > /sys/class/gpio/gpio23/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'KITCHEN LIGHT ON':
                        # P8_18 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 65 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_18 ON
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo high > /sys/class/gpio/gpio65/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'KITCHEN LIGHT OFF':
                        # P8_18 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 65 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_18 OFF
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo low > /sys/class/gpio/gpio65/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'HALL DOOR OPEN':
                        # P8_9 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 69 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_9 ON
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo high > /sys/class/gpio/gpio69/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'HALL DOOR CLOSE':
                        # P8_9 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 69 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_9 OFF
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo low > /sys/class/gpio/gpio69/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'HALL TV ON':
                        # P8_19 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 22 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_19 ON
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo high > /sys/class/gpio/gpio22/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'HALL TV OFF':
                        # P8_19 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 22 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_19 OFF
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo low > /sys/class/gpio/gpio22/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'HALL LIGHT ON':
                        # P8_12 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 44 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_12 ON
                        sys.stdout."""write('SET VARIABLE ESTADO
                                        "echo high > /sys/class/gpio/gpio44/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'HALL LIGHT OFF':
                        # P8_12 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 44 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_12 OFF
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo low > /sys/class/gpio/gpio44/direction"\n""")
```

```
                        sys.stdout.flush()
                elif comando.group() == 'CHAMBER DOOR OPEN':
                        # P8_10 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 68 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_10 ON
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo high > /sys/class/gpio/gpio68/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'CHAMBER DOOR CLOSE':
                        # P8_10 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 68 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_10 OFF
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo low > /sys/class/gpio/gpio68/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'CHAMBER TV ON':
                        # P_15 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 47 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P_15 ON
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo high > /sys/class/gpio/gpio47/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'CHAMBER TV OFF':
                        # P_15 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 47 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P_15 OFF
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo low > /sys/class/gpio/gpio47/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'CHAMBER LIGHT ON':
                        # P8_11 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 45 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_11 ON
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo high > /sys/class/gpio/gpio45/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'CHAMBER LIGHT OFF':
                        # P8_11 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 45 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_11 OFF
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo low > /sys/class/gpio/gpio45/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'BATHROOM DOOR OPEN':
                        # P8_14 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 26 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_14 ON
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo high > /sys/class/gpio/gpio26/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'BATHROOM DOOR CLOSE':
                        # P8_14 EXPORT
                        sys.stdout.write("""SET VARIABLE GPIO
                                        "echo 26 > /sys/class/gpio/export"\n""")
                        sys.stdout.flush()
                        # P8_14 OFF
                        sys.stdout.write("""SET VARIABLE ESTADO
                                        "echo low > /sys/class/gpio/gpio26/direction"\n""")
                        sys.stdout.flush()
                elif comando.group() == 'BATHROOM LIGHT ON':
```

```
296                         # P8_17 EXPORT
297                         sys.stdout.write("""SET VARIABLE GPIO
298                                         "echo 27 > /sys/class/gpio/export"\n""")
299                         sys.stdout.flush()
300                         # P8_17 ON
301                         sys.stdout.write("""SET VARIABLE ESTADO
302                                         "echo high > /sys/class/gpio/gpio27/direction"\n""")
303                         sys.stdout.flush()
304                 elif comando.group() == 'BATHROOM LIGHT OFF':
305                         # P8_17 EXPORT
306                         sys.stdout.write("""SET VARIABLE GPIO
307                                         "echo 27 > /sys/class/gpio/export"\n""")
308                         sys.stdout.flush()
309                         # P8_17 OFF
310                         sys.stdout.write("""SET VARIABLE ESTADO
311                                         "echo low > /sys/class/gpio/gpio27/direction"\n""")
312                         sys.stdout.flush()
313
314 # Encerrando o script
315 sys.exit(0)
```

ANEXO D – Arquivo de configuração do Julius

```
#
# Arquivo de configuração do Julius (rev.4.1.1)
#
# 1) Options can also be specified in command line option.
#    The values are default values in Julius.
# 2) For file name, relative path must be relative to THIS FILE.
# 3) Texts after '#' at each line will be ignored.  If you want to specify
#    '#', use `\#'.
# 4) Each line should be no longer than 512 bytes.
#
#

#################################################################
#### JULIUS APPLICATION OPTION (not a part of JuliusLib)
#################################################################
-outfile                        # save each result in separate file

#################################################################
#### GLOBAL OPTIONS
#################################################################
####
#### Audio Input
####
-input rawfile                  # wavefile
-filelist lista.txt             # input file list

#################################################################
#### LANGUAGE MODEL (-LM)
#################################################################
####
#### Only one type of LM can be specified for a LM configuration.
####

####
#### Grammar
####
#### "-gram", "-gramlist" can be used multiple times

# Forma classica de especificar a gramatica

-dfa sample.dfa
-v sample.dict

#################################################################
#### ACOUSTIC MODEL (-AM)  (-AM_GMM)
#################################################################
####
#### Acoustic analysis parameters are included in this section, since
#### the AM defines the required parameter.  You can use different MFCC
#### type for each AM.
#### For GMM, the same parameter should be specified after "-AM_GMM"
####
#### When using multiple AM,  the values of "-smpPeriod", "-smpFreq",
#### "-fsize" and "-fshift" should be the same among all AM.
####

## Acoustic model

# Modelo Acustico Locutor1
-h acoustic_model_files/hmmdefsLocutor1         # acoustic HMM (ascii or Julius binary)
-hlist acoustic_model_files/tiedlistLocutor1            # HMMList to map logical phone to
physical

# Modelo Acustico Locutor2
#-h acoustic_model_files/hmmdefsLocutor2
#-hlist acoustic_model_files/tiedlistLocutor2

# Modelo Acustico Locutor3
#-h acoustic_model_files/hmmdefsLocutor3
#-hlist acoustic_model_files/tiedlistLocutor3

# Modelo Acustico Locutor4
#-h acoustic_model_files/hmmdefsLocutor4
#-hlist acoustic_model_files/tiedlistLocutor4
```

```
74
75  # Modelo Acustico Locutor5
76  #-h acoustic_model_files/hmmdefsLocutor5
77  #-hlist acoustic_model_files/tiedlistLocutor5
78
79  # Modelo Acustico Locutor6
80  #-h acoustic_model_files/hmmdefsLocutor6
81  #-hlist acoustic_model_files/tiedlistLocutor6
82
83  # Modelo Acustico Locutor7
84  #-h acoustic_model_files/hmmdefsLocutor7
85  #-hlist acoustic_model_files/tiedlistLocutor7
86
87  # Modelo Acustico Locutor8
88  #-h acoustic_model_files/hmmdefsLocutor8
89  #-hlist acoustic_model_files/tiedlistLocutor8
90
91  # Modelo Acustico Locutor9
92  #-h acoustic_model_files/hmmdefsLocutor9
93  #-hlist acoustic_model_files/tiedlistLocutor9
94
95  # Modelo Acustico Locutor10
96  #-h acoustic_model_files/hmmdefsLocutor10
97  #-hlist acoustic_model_files/tiedlistLocutor10
98
99  -spmodel "sp"                                   # name of a short-pause silence model
100
101 -gprune safe                                    # Gaussian pruning method
102
103 -iwcd1 max                                      # Inter-word triphone approximation method
104
105 -iwsppenalty -70.0              # pause insertion penalty for "-iwsp"
106
107 ## Analysis
108 -smpFreq 8000                   # sampling rate (Hz)
109
110 ################################################################
111 #### RECOGNIZER (-SR)
112 ################################################################
113 ####
114 #### Default values for beam width and LM weights will change
115 #### according to compile-time setup of JuliusLib and model specification.
116 #### Please see the startup log for the actual values.
117 ####
118
119 ####
120 #### parameter (common)
121 ####
122 -iwsp                           # append a skippable sp at all word ends
123
124 ####
125 #### parameter (1st pass)
126 ####
127 -penalty1 5.0           # word insertion penalty for grammar (pass1)
128
129 ####
130 #### parameter (2nd pass)
131 ####
132 -penalty2 20.0          # word insertion penalty for grammar (pass2)
133
134 -b2 200                         # envelope beam width of 2nd pass (#word)
135
136 -sb 200.0                       # envelope score width at 2nd pass
137
138 ################################################################ end of file
```

ANEXO E – Projeto submetido ao comitê de ética em pesquisa do IFPB

Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Programa de Pós-Graduação em Engenharia Elétrica

MESTRADO EM ENGENHARIA ELÉTRICA

Projeto de Pesquisa

Autor:
Victor Costa de Andrade Pimentel

Orientadora:
Suzete Elida Nóbrega Correia

Coorientadora:
Silvana Luciene do Nascimento Cunha Costa

Linha de Pesquisa:
Processamento Digital de Sinais

João Pessoa, 2015

Nome do Projeto
Plataforma de reconhecimento da fala, baseada em Asterisk e Beaglebone, para o auxílio de pessoas com mobilidade reduzida

Desenho

O estudo envolve a avaliação do funcionamento de uma plataforma de reconhecimento da fala, livre e de baixo custo, dependente de locutor – desenvolvida no Instituto Federal de Educação, Ciência e Tecnologia da Paraíba como resultado da pesquisa de dissertação de mestrado do aluno Victor Costa de Andrade Pimentel – a ser empregada no controle de tecnologias assistivas a partir da interação humano-computador através da pronúncia de comandos pelo usuário. Componentes fundamentais das ferramentas com esse propósito, os modelos acústicos e modelos de linguagem são gerados através da aplicação de um processamento estatístico, cuja entrada de dados se constitui de amostras de áudio da fala, associadas a uma lista de palavras foneticamente balanceada e a um dicionário fonético do idioma utilizado. As referidas amostras de áudio, conhecidas como amostras de treinamento, são obtidas realizando-se a gravação da voz de indivíduos. Para tanto, o procedimento de coleta das amostras de treinamento, requer a participação de dez sujeitos amostrais, dentre cidadãos brasileiros comuns, sendo cinco do sexo masculino e cinco do sexo feminino, que apresentem preservada boa capacidade das habilidades da fala, com idade maior ou igual a 18 anos, escolhidos por acessibilidade. Além disso, uma vez que os modelos da plataforma estejam treinados para a fala de um sujeito amostral, este deverá testá-la repetindo os comandos determinados para o cenário de avaliação especificado na metodologia do projeto. Com isso, pretende-se medir a taxa percentual de acertos que o sistema apresenta ao decodificar os sinais da fala e gerar uma saída de controle conforme esperado, de forma que uma alta taxa percentual de acertos indica um bom desempenho do sistema. Trata-se, portanto, de uma pesquisa de abordagem metodológica qualitativa, que observa o comportamento do sistema desenvolvido com relação a sua resposta aos comandos de voz, de modo a caracterizar seu desempenho.

Palavras-chave

Tecnologias assistivas, Reconhecimento automático da fala, Sistemas embarcados, Automação residencial.

Resumo

O foco de atuação do projeto volta-se para o desenvolvimento de interface de comandos por voz que seja facilmente adaptada e incorporada aos já existentes sistemas e ferramentas de auxílio ao controle do ambiente doméstico (domótica). Esses sistemas atuam fundamentalmente permitindo o controle remoto das facilidades do lar por indivíduos que apresentam alguma restrição motora ou dificuldade quanto a sua mobilidade. Representam, assim, formas de auxiliar no cotidiano dessas pessoas dando suporte à execução de tarefas como: controlar eletrodomésticos, iluminação ambiente, dispositivos de cuidados com a saúde, dentre outros. Nesses casos, convém utilizar interfaces alternativas de controle. Um exemplo é o uso do reconhecimento de voz para a interação com tais facilidades, de modo a superar a exigência de habilidades de coordenação motora para a entrada de comandos aos referidos sistemas. Os compromissos a serem atingidos envolvem, portanto, o atendimento de critérios de portabilidade e mobilidade, exigindo que as interfaces propostas apresentem um caráter multifuncional, através de uma estrutura embarcável capaz de prover a entrada de comandos de voz para a tomada de decisões de controle. Além disso, devem ser observados requisitos de baixo custo e alta flexibilidade, utilizando-se de tecnologias livres e de código aberto que contribuam para o desenvolvimento de aplicações de tecnologia assistiva acessíveis.

Introdução

As mais recentes conquistas tecnológicas nas áreas da informação e comunicação vêm causando profundas mudanças na relação entre os usuários e os sistemas de computador. A cada dia, visando melhorar a qualidade de vida das pessoas, surgem novos conceitos e paradigmas que determinam o predomínio das máquinas e dispositivos eletroeletrônicos em praticamente todos os cenários do cotidiano (SILVA, 2011; YANG et al., 2014; SEBESTYEN et al., 2014). Esses conceitos geram um forte impacto no desenvolvimento de novas tecnologias que permitam a integração e intercomunicação entre diferentes dispositivos, ou simplesmente "coisas", através da Internet, caracterizando soluções computacionais ubíquas e pervasivas, associadas à concepção de ambientes inteligentes. Emerge daí uma ampla gama de aplicações potenciais que tem ganhado significativa atenção por parte da indústria e do meio acadêmico (PERERA et al., 2014), dentre as quais, as tecnologias assistivas, em diversas modalidades, são apenas uma parte. Nesse contexto, a Internet das Coisas (do termo inglês Internet of Things - IoT), abarca soluções em eHealth, sistemas de home healthcare, elderly aid e ambient assisted living (AAL) (DOHR et al., 2010), que apresentam uma relação muito próxima aos conceitos mencionados, podendo agregar ainda funcionalidades de controle e automação num ambiente doméstico, constituindo, por exemplo, residências inteligentes. Em meio a esse mundo de integração tecnológica e interconectividade, não se tem medido esforços no sentido de proporcionar modalidades de entrada através das quais os humanos interajam da maneira mais simples e natural possível com as máquinas. Aqui observa-se o "... crescente interesse por softwares e equipamentos que 'compreendam', reconheçam e simulem a voz humana..."(SILVA, 2011). A popularização dessa forma de interação com as máquinas envolve a disponibilização de recursos para o desenvolvimento de interfaces de voz acessíveis e portáveis, que possam ser facilmente adaptadas às mais diversas aplicações (BATISTA, 2013), inclusive, integrando-se a poderosos servidores de comunicação (DIAS; LUCENA; SANTOS, 2014). Propõe-se a implementação de interface de comandos por voz para sistemas embarcados, flexível e modular, alinhada ao contexto da IoT, e que auxilie indivíduos idosos e outras pessoas com deficiência motora ou mobilidade restrita a executar tarefas domésticas cotidianas.

Hipótese

A busca por maior independência e autonomia para as pessoas com deficiência tem se apresentado como um fator decisivo, ao proporcionar mais qualidade de vida a esses indivíduos. A superação desses desafios se evidencia na importância de se combinar o uso de tecnologias assistivas com o suporte humano tradicional. O emprego dessas tecnologias não deve ser visto como uma prática que substitua o cuidado humano. Contudo, existem diversas aplicações em que as tecnologias assistivas atendem necessidades que os cuidados humanos não seriam capazes dar o devido suporte. Como, por exemplo, o monitoramento dos pacientes vinte e quatro horas por dia, com o intuito de prevenir acidentes e auxiliar em tarefas diárias simples. Na maioria desses casos a voz das pessoas é razoavelmente inalterada em comparação com as debilidades de coordenação motora (QIDWAI; SHAKIR, 2012). Dessa forma, a incorporação de interfaces de controle por voz se apresenta como uma conveniente alternativa às interfaces de controle tradicionais para tecnologias assitivas. No entanto, o suporte proporcionado pelas tecnologias assistivas atualmente ainda tem sua ampla utilização e integração tecnológica limitada por, estar condicionada a soluções proprietárias. Demanda esforços voltados para o emprego de tecnologias, padrões e protocolos abertos. Além disso, os custos para aquisição e manutenção dos equipamentos de tecnologia assistiva ainda são muito altos. Diante desse contexto, o desenvolvimento de tecnologias assistivas de baixo custo utilizando-se de tecnologias livres e de código aberto que contribuam para o desenvolvimento de aplicações de tecnologia assistiva acessíveis se apresenta como um relevante papel a ser desempenhado em benefício de melhorias na funcionalidade (SARTORETTO; BERSCH, 2014) de pessoas com deficiência.

Objetivo Primário

Desenvolver uma plataforma de reconhecimento da fala, empregando a placa de desenvolvimento Beaglebone Black e o servidor de comunicação Asterisk, para o auxílio de pessoas com mobilidade reduzida.

Objetivo Secundário

Realizar estudo sobre tecnologias eficientes, portáveis e de baixo custo que se mostrem viáveis ao desenvolvimento interfaces de comandos por voz para a interação humano-computador. Elaborar concepção de interface de baixo custo, portável e de uso pessoal, para aplicações de comandos através do reconhecimento de fala em tecnologias assistivas que atuem no controle de dispositivos domésticos. Desenvolver recursos de base de texto (gramática) e base de áudio para a criação de modelo acústico específico para aplicações de comando e controle em domótica. Avaliar a interface concebida com fins de validar sua eficiência.

Metodologia proposta

A plataforma desenvolvida integra componentes de hardware aberto e software livre e de código aberto, sendo os comandos de voz fornecidos ao sistema através de smartphone configurado com softphone VoIP. Esse, se registra no servidor de comunicação Asterisk, que implementa central telefônica com unidade de resposta audível (URA). Integrada ao servidor, está a ferramenta de reconhecimento automático da fala, Julius. Esses componentes estão embarcados na plataforma Beaglebone Black, de baixo custo. Uma vez que a integração entre os componentes foi realizada, o sistema precisa de modelos acústicos e de linguagem para o Julius. Utilizando esses modelos, a ferramenta será capaz de receber uma sequência de palavras pronunciadas e retornar a representação textual da frase comparando-a, posteriormente, com comandos previamente definidos no cenário de validação da plataforma, levando o sistema a tomar decisão e gerar o sinal de controle correspondente. Como o intuito é proporcionar entrada de comandos de voz para o controle de tecnologias assistivas, o cenário constitui-se de um protótipo que ilustra a geração de sinais de controle para a iluminação, televisão e acesso (portas), num ambiente doméstico hipotético constituído de sala, cozinha, quarto, banheiro e área externa, através do acendimento de LEDs (Light Emitter Diodes). Sistema dependente do locutor, capaz de reconhecer frases com três palavras, utilizando modelo de linguagem de estados finitos (finite-state grammar). Esse tipo de aplicação, associado à central VoIP, exige o treinamento de modelos adaptados às condições do canal de áudio. Para isso, é necessária a gravação, em sessões individuais, de amostras de áudio com a fala de dez sujeitos amostrais, dentre cidadãos brasileiros comuns, cinco do sexo masculino e cinco do feminino, com preservada boa capacidade das habilidades da fala, idade igual ou superior a 18 anos, escolhidos por acessibilidade. As gravações, em ambiente silencioso, serão obtidas através de headset conectado ao smartphone, a partir de uma chamada originada do softphone para ramal da URA no servidor. A fala a ser gravada deve observar a leitura de uma lista de palavras no idioma inglês, observando um balanceamento fonético específico para os comandos do cenário, divididas em quarenta amostras (samples), obtendo-se quarenta arquivos de áudio, cada um com a fala das palavras listadas na respectiva amostra. O processo de gravação deve durar de três a quatro horas, podendo ser realizado em até três sessões distintas para cada sujeito amostral. Essas gravações serão registradas no formato WAV (Microsoft 16 bit PCM), com taxa de amostragem de 8 Khz, monocanal, sendo posteriormente utilizadas para a geração dos modelos compatíveis com o decodificador Julius, empregando-se o pacote de ferramentas HTK (Hidden Markov Model Tookit), que possui limitações de uso acadêmico para sua distribuição; não havendo, no entanto, limitações para a distribuição dos modelos criados com esse pacote. De posse do modelo criado para a fala de um indivíduo, o Julius será configurado para utilizá-lo no reconhecimento da fala durante a

execução do procedimento avaliativo da plataforma, onde o respectivo indivíduo fornecerá à entrada do sistema vinte repetições de cada um dos comandos, observando-se o comportamento do sistema. Serão registradas as ocasiões em que o sistema comportou-se conforme o esperado, não gerou resposta, ou ainda, haja a ocorrência de falsos positivos. Será calculada a taxa média de acertos do sistema, esperando-se um índice a partir de 90%. O tempo de duração do teste é de cerca de duas hora e meia, podendo ser realizado em até duas sessões distintas para cada sujeito amostral. As amostras de áudio serão mantidas para a formação de banco de dados, podendo ser publicadas, sem identificação do dono da voz, no projeto Voxforge, com a devida autorização dos participantes formalizada pelo TCLE, para uso em ferramentas de reconhecimento de fala de código aberto.

Critério de inclusão

Participarão da pesquisa dez cidadãos comuns, sendo cinco do sexo masculino e cinco do sexo feminino, com preservada boa capacidade das habilidades da fala, idade igual ou superior a 18 anos, escolhidos por acessibilidade, e que manifestem sua vontade de permanecer na pesquisa por intermédio de manifestação expressa, livre e esclarecida.

Critério de exclusão

Como a amostra de interesse engloba restrições somente relacionadas à preservação das habilidades da fala, não sendo imprescindível que o sujeito amostral apresente um estado de saúde que caracterize mobilidade reduzida – apesar do potencial da pesquisa em trazer benefício às pessoas com mobilidade reduzida, dentre os quais idosos e portadores de necessidades especiais, que podem apresentar alguma vulnerabilidade – será dada preferência à participação de indivíduos com autonomia plena, evitando a exposição daqueles aos riscos envolvidos no estudo.

Riscos

Os participantes da pesquisa estarão sujeitos a riscos relacionados à realização de esforço prolongado ao falar repetidamente para a gravação das amostras de treinamento, como também para a realização dos testes da plataforma, podendo ocasionar fadiga. Visando minimizar esse risco, a realização das gravações de amostras de áudio de treinamento poderão ser realizadas em até três sessões distintas para cada indivíduo, como também o teste do sistema ser realizado em até duas sessões distintas. Durante as sessões, os participantes serão ainda orientados a realizar pausas a cada 30 minutos de gravação, sendo fornecida água para sua hidratação. Além disso, os participantes poderão sentir-se constrangidos com a possibilidade de publicação no projeto Voxforge dos arquivos de áudio gravados com sua voz. Visando diminuir esse desconforto, a publicação do arquivo será realizada sem a identificação do locutor gravado, ou ainda, o participante poderá optar pela não publicação do arquivo gravado com sua voz, conforme marcação refente a essa opção que será devidamente formalizada pelo TCLE.

Benefícios

A pesquisa disponibilizará recursos de base de texto e base de áudio para a criação de modelos acústicos e uma plataforma de reconhecimento de comandos de fala livres, de código aberto e de baixo custo, contribuindo para a popularização dessas tecnologias no âmbito das tecnologias assistivas que proporcionem maior independência e autonomia para as pessoas com mobilidade reduzida, tendo em vista sua inclusão social e digital.

Metodologia de análise de dados

De posse do modelo criado para a fala de um indivíduo, o Julius será configurado para utilizá-lo no

reconhecimento da fala durante a execução do procedimento avaliativo da plataforma, onde o respectivo indivíduo fornecerá à entrada do sistema vinte repetições de cada um dos comandos, observando-se o comportamento do sistema. Serão registradas as ocasiões em que o sistema comportou-se conforme o esperado, não gerou resposta, ou ainda, haja a ocorrência de falsos positivos. Por fim, será calculada a taxa média de acertos do sistema.

Desfecho primário

Ao final da pesquisa, espera-se que os recursos de base de texto e base de áudio sejam disponibilizados para a comunidade de desenvolvimento de ferramentas de reconhecimento da fala de código aberto, bem como que a plataforma de reconhecimento de fala desenvolvida alcance o percentual de acerto maior que 90%.

Cronograma

* **Cronograma de execução:**

Identificação da Etapa	Início (dd/mm/aaaa)	Término (dd/mm/aaaa)
Submissão do projeto ao Comitê de Ética do IFPB por meio da Plataforma Brasil	26/10/2015	26/10/2015
Gravação das amostras de treinamento com os participantes	30/11/2015	11/12/2015
Realização dos testes de avaliação da plataforma com os participantes	07/12/2015	18/12/2015
Análise dos resultados dos testes	17/12/2015	23/12/2015
Relatório final	24/12/2015	08/01/2016

Orçamento financeiro

* **Orçamento Financeiro:**

Detalhamento do Orçamento:

Identificação do Orçamento	Tipo	Valor em Reais (R$)	Ações
Placa de desenvolvimento Beaglebone Black	Custeio	527,86	
Adaptador Wireless USB Intelbrás WBN900	Custeio	55,00	
10 LEDs 10mm de alto brilho	Custeio	15,00	
10 LEDs 5mm de alto brilho	Custeio	10,00	
Caixa organizadora de MDF	Custeio	22,50	
Matriz de contatos 2860 pontos	Custeio	65,50	
15 Resistores 330Ohm	Custeio	3,00	

Total em Reais (R$): 698,86

Adicionar Despesa

Referências

BATISTA, P. dos S. Avanços em Reconhecimento de Fala para Português Brasileiro e Aplicações: Ditado no libreoffice e unidade de resposta audível com asterisk. 95 f. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal do Pará, Belém, 2013.

BEAGLEBONE Black. Disponível em: <http://beagleboard.org/black>. Acesso em: 12 jun. 2015.

CHANDRAMOULI, C.; AGARWAL, V. Speech recognition based computer keyboard

replacement for the quadriplegics, paraplegics, paralytics and amputees. In: INTERNATIONAL WORKSHOP ON MEDICAL MEASUREMENTS AND APPLICATIONS, 2009, Cetraro. Proceedings... Cetraro: IEEE, 2009. p. 241–245.

CMU. Training Acoustic Model For CMUSphinx. Disponível em: <http://cmusphinx.sourceforge.net/wiki/tutorialam>. Acesso em: 10 out 2015.

DIAS, M. C.; LUCENA, D. C.; SANTOS, E. P. O uso do asterisk para o controle remoto de sistemas de automação. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, XX., 2014, MG. Anais... Belo horizonte: Universidade Federal de Minas Gerais, 2014.

DOHR, A. et al. The internet of things for ambient assisted living. In: INTERNAITONAL CONFERENCE ON INFORMATION TECHNOLOGY, 7., 2010, Estados Unidos. Proceedings... Las Vegas: IEEE, 2010.

JULIUS: Open-source large vocabulary csr engine. 2014. Disponível em: <http: //julius.osdn.jp/en_index.php?q=index-en.html>.

MADSEN, L.; MEGGELEN, J. V.; BRYANT, R. Asterisk TM : The Definitive Guide. 3. ed. Sebastopol:
O'Reilly Media, Inc., 2011. 734 p. ISBN 978-0-569-51734-2.

PERERA, C. et al. A survey on internet of things from industrial market perspective. The Journal for rapid open access publishing, v. 2, p. 1660–1679, Dec. 2014.

PIMENTEL, V. C. A. et al. Uma plataforma de baixo custo comandada por voz para tecnologias assistivas com programação em python. In: CONGRESSO BRASILEIRO DE ENGENHARIA BIOMÉDICA, XXIV., 2014, MG. Anais... Uberlândia: Universidade Federal de Uberlândia, 2014.

QIDWAI, U.; SHAKIR, M. Ubiquitous arabic voice control device to assist people with disabilities. In: INTERNATIONAL CONFERENCE ON INTELLIGENT AND ADVANCED SYSTEMS, 4., 2012, Kuala Lumpur. Proceedings... Kuala Lumpur: IEEE, 2012. p. 333–338.

SARTORETTO, M. L.; BERSCH, R. Tecnologia Assistiva. Assistiva ® •Tecnologia e Educação, 2014. Disponível em: <http://www.assistiva.com.br/tassistiva.html>. Acesso em: 26 mar. 2014.

SEBESTYEN, G. et al. ehealth solutions in the context of internet of things. In: INTERNAITONAL CONFERENCE ON AUTOMATION, QUALITY AND TESTING, ROBOTICS,
19., 2014, Romênia. Proceedings... Cluj-Napoca: IEEE, 2014.

SILVA, D. D. C. da. Reconhecimento de Fala Contínua para o Português Brasileiro em Sistemas Embarcados. 194 f. Tese (Doutorado em Ciências) — Universidade Federal de Campina Grande, Campina Grande, 2011.

SPAANS, M. A. On Developing Acoustic Models Using HTK. 113 f. Dissertação (Master of Science) — Delft University of Technology, Delft, 2004.

VOXFORGE. Disponível em: <http://www.voxforge.org/>. Acesso em: 11 jun. 2015.

VOXFORGE. How-to: Create acoustic model - with script. Disponível em: <http:

//www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/how-to>. Acesso em: 03 jul. 2015.

VOXFORGE. Tutorial: Create Acoustic Model - Manually. Disponível em: <http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/tutorial>. Acesso em: 10 out. 2015.

YANG, L. et al. A home mobile healthcare system for wheelchair users. In: INTERNAITONAL CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK IN DESIGN, 18., 2014, Taiwan. Proceedings... Hsinchu: IEEE, 2014.

ANEXO F – Modelo do Termo de Consentimento Livre e Esclarecido

Termo de Consentimento Livre e Esclarecido (TCLE)

Prezado participante,

Você está sendo convidado a participar da pesquisa **Plataforma de reconhecimento da fala, baseada em Asterisk e Beaglebone, para o auxílio de pessoas com mobilidade reduzida,** desenvolvida por **Victor Costa de Andrade Pimentel**, discente do Curso de Pós-Graduação *Strictu Sensu* de Mestrado em Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - IFPB, sob orientação da Professora Dra. **Suzete Elida Nóbrega Correia** e coorientação da professora Dra. **Silvana Luciene do Nascimento Cunha Costa**.

O objetivo central do estudo é: Desenvolver uma plataforma de reconhecimento da fala para o auxílio de pessoas com mobilidade reduzida no controle dispositivos do ambiente doméstico. Os objetivos específicos são: Realizar estudo sobre tecnologias eficientes e de baixo custo que se mostrem viáveis ao desenvolvimento interfaces de comandos por voz para a interação humano-computador. Elaborar concepção de interface de baixo custo, portável e de uso pessoal, para aplicações de comandos através do reconhecimento de voz em tecnologias assistivas que atuem no controle de dispositivos domésticos. Desenvolver recursos de base de texto (gramática) e base de áudio para a criação de modelo acústico específico para aplicações de comando e controle em domótica. Avaliar a interface concebida com fins de validar sua eficiência.

O motivo de sua participação se deve ao fato de você ser cidadão brasileiro, com preservadas boas capacidades da fala e idade igual ou superior a 18 anos. Sua participação é voluntária e você tem plena autonomia para decidir se quer ou não participar, bem como retirar sua participação a qualquer momento. Você não será penalizado de nenhuma maneira caso decida não consentir sua participação, ou desistir do seu consentimento. Contudo, ela é muito importante para a execução da pesquisa.

A sua participação consistirá em sua fala ser gravada, em ambiente silencioso, utilizando um *headset* (fone de ouvido e microfone), pronunciando a leitura de uma lista de palavras no idioma inglês (fornecida pelo pesquisador), divididas em quarenta amostras, obtendo-se quarenta arquivos de áudio, cada um com a fala das palavras listadas na respectiva amostra. O processo de gravação deve durar de três a quatro horas, podendo ser realizado em até três sessões distintas. Essas gravações serão registradas e utilizadas para treinar um sistema de reconhecimento da fala, que se tornará, então, capaz de reconhecer comandos de voz com sua fala. Numa próxima etapa, realizaremos um procedimento para testar o sistema de comandos por voz, em que você pronunciará vinte repetições de cada um dos comandos definidos, no idioma inglês, para o cenário de testes do sistema. Esses testes serão utilizados para verificar se a plataforma está realizando os comandos de voz de forma satisfatória. O tempo de duração do teste é de cerca de duas hora e meia, podendo ser realizado em até duas sessões distintas. Com o intuito de contribuir com a comunidade de desenvolvimento de tecnologias livres e de código aberto, de baixo custo, para o reconhecimento de fala, os arquivos gravados com sua voz poderão ser disponibilizados no projeto Voxforge (http://www.voxforge.org/), conforme sua opção escolhida no devido campo desse

termo, sem a necessidade de que você seja identificado. Ao final da pesquisa, todo material será mantido em arquivo, por pelo menos 5 anos, conforme Resolução nº466/12 do Conselho Nacional de Saúde.

Com a sua colaboração nesta pesquisa, você não terá benefício pessoal direto, mas a sua participação proporcionará a obtenção de dados que permitirão contribuir com a comunidade de desenvolvimento de tecnologias livres e de código aberto, de baixo custo, para o reconhecimento de fala, que auxiliem no cotidiano de pessoas com mobilidade reduzida, proporcionando-lhes maior independência, autonomia e, consequentemente, mais qualidade de vida.

Em decorrência de sua participação na pesquisa, você poderá sentir cansaço devido ao esforço prolongado ao falar repetidamente para a gravação das amostras de treinamento, como também para a realização dos testes da plataforma. Para minimizar esse desconforto, a realização das gravações de amostras de áudio de treinamento poderão ser realizadas em até 3 sessões distintas, como também o teste do sistema ser realizado em até duas sessões distintas. Durante as sessões, serão realizadas pausas a cada 30 minutos de gravação, sendo fornecida água para sua hidratação. Além disso, você poderá sentir-se constrangido com a possibilidade de publicação no projeto Voxforge dos arquivos de áudio gravados com sua voz. Visando minimizar esse desconforto, a publicação do arquivo será realizada sem sua identificação, ou você pode ainda optar pela não publicação do arquivo gravado com sua voz, conforme a opção feita a seguir.

O participante permite a publicação dos arquivos de áudio gravados com sua voz no projeto Voxforge (http://www.voxforge.org/)? ( ) Sim. ( ) Não.

Os resultados desta pesquisa serão divulgados em artigo científico e em dissertação do Curso de Pós-Graduação *Strictu Sensu* de Mestrado em Engenharia Elétrica do IFPB. Seu nome não será identificado em nenhum desses documentos.

Este Termo de Consentimento Livre e Esclarecido é redigido em duas vias, sendo uma para o participante e outra para o pesquisador. Todas as páginas do documento serão rubricadas pelo participante da pesquisa e pelo pesquisador responsável, com exceção da última página, onde serão apostas ambas as assinaturas.

Para qualquer outra informação sobre a pesquisa, você poderá entrar em contato com o pesquisador responsável pelo telefone (84) 99604-6260, e-mail victor.andrade@ifrn.edu.br. Em caso de dúvida quanto à condução ética do estudo, entre em contato com o Comitê de Ética em Pesquisa do IFPB através dos seguintes canais de comunicação: telefone (83) 3612-9725, e-mail eticaempesquisa@ifpb.edu.br, endereço Avenida João da Mata, 256, Jaguaribe, João Pessoa-PB.

Consentimento Pós-Informação

Eu, _____ , abaixo assinado, fui devidamente esclarecido quanto os objetivos da pesquisa, aos procedimentos aos quais serei submetido e os possíveis riscos decorrentes da minha participação. Diante do exposto, aceito livremente participar do estudo intitulado **Plataforma de reconhecimento da fala, baseada em Asterisk e Beaglebone, para o auxílio de pessoas com mobilidade**

reduzida, desenvolvido pelo pesquisador **Victor Costa de Andrade Pimentel,** sob a orientação da Professora Dra. **Suzete Elida Nóbrega Correia** e coorientação da professora Dra. **Silvana Luciene do Nascimento Cunha Costa**.

João Pessoa, _____ de _____ de _____

_____
Assinatura do participante

_____
Assinatura do pesquisador responsável

ANEXO G – Parecer do comitê de ética aprovando a pesquisa

PARECER CONSUBSTANCIADO DO CEP

DADOS DO PROJETO DE PESQUISA

Título da Pesquisa: Plataforma embarcada de reconhecimento da fala, baseada em Beaglebone e Asterisk, para o auxílio de pessoas com mobilidade reduzida.

Pesquisador: Victor Costa de Andrade Pimentel

Área Temática:

Versão: 1

CAAE: 50486815.6.0000.5185

Instituição Proponente: Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - IFPB

Patrocinador Principal: Financiamento Próprio

DADOS DO PARECER

Número do Parecer: 1.350.868

Apresentação do Projeto:

O projeto de pesquisa em análise consiste de uma pesquisa de abordagem metodológica qualitativa, que observa o comportamento do sistema desenvolvido com relação a sua resposta aos comandos de voz, de modo a caracterizar seu desempenho. O estudo envolve a avaliação do funcionamento de uma plataforma de reconhecimento da fala, livre e de baixo custo, dependente de locutor – desenvolvida no Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – a ser empregada no controle de tecnologias assistivas a partir da interação humano-computador através da pronúncia de comandos pelo usuário. Os modelos acústicos e modelos de linguagem, componentes fundamentais das ferramentas com esse propósito, são gerados através da aplicação de um processamento estatístico, cuja entrada de dados se constitui de amostras de áudio da fala, associadas a uma lista de palavras foneticamente balanceada e a um dicionário fonético do idioma utilizado. As referidas amostras de áudio, conhecidas como amostras de treinamento, são obtidas realizando-se a gravação da voz de indivíduos. Para tanto, o procedimento de coleta das amostras de treinamento, requer a participação de 10 (dez) sujeitos amostrais, dentre cidadãos brasileiros comuns, que apresentem preservada boa capacidade das habilidades da fala, com idade maior ou igual a 18 anos, escolhidos por acessibilidade. Uma vez que os modelos da plataforma estejam treinados para a fala de um sujeito amostral, este deverá testá-la repetindo os comandos

determinados para o cenário de avaliação especificado na metodologia do projeto. Com isso, pretende-se medir a taxa percentual acertos que o sistema apresenta ao decodificar os sinais da fala e gerar uma saída de controle conforme esperado, de forma que uma alta taxa percentual de acertos indica um bom desempenho do sistema.

Objetivo da Pesquisa:

Primário:

Desenvolver uma plataforma de reconhecimento da fala, empregando a placa de desenvolvimento Beaglebone Black e o servidor de comunicação Asterisk, para o auxílio de pessoas com mobilidade reduzida.

Secundários:

- Realizar estudo sobre tecnologias eficientes, portáveis e de baixo custo que se mostrem viáveis ao desenvolvimento interfaces de comandos por voz para a interação humano-computador;
- Elaborar concepção de interface de baixo custo, portável e de uso pessoal, para aplicações de comandos através do reconhecimento de fala em tecnologias assistivas que atuem no controle de dispositivos domésticos;
- Desenvolver recursos de base de texto (gramática) e base de áudio para a criação de modelo acústico específico para aplicações de comando e controle em domótica (controle do ambiente doméstico);
- Avaliar a interface concebida com fins de validar sua eficiência.

Avaliação dos Riscos e Benefícios:

Riscos:

De acordo com a Resolução nº 466/2012, item V do Conselho Nacional de Saúde - CNS "Toda pesquisa com seres humanos envolve risco em tipos e gradações variados uma vez que esse não se limitam a danos à dimensão física e/ou moral, mas também psíquica, intelectual, social, cultural ou espiritual do ser humano, em qualquer fase de uma pesquisa e dela decorrente". No projeto em análise o autor afirma que os participantes da pesquisa estarão sujeitos a riscos relacionados à realização de esforço prolongado ao falar repetidamente para a gravação das amostras de treinamento, como também para a realização dos testes da plataforma, podendo ocasionar fadiga. Visando minimizar esse risco, a realização das gravações de amostras de áudio de treinamento poderão ser realizadas em até três sessões distintas para cada indivíduo, como também o teste do sistema ser realizado em até duas sessões distintas. Durante as sessões, os participantes serão

ainda orientados a realizar pausas a cada 30 minutos de gravação, sendo fornecida água para sua hidratação. Além disso, os participantes poderão sentir-se constrangidos com a possibilidade de publicação dos arquivos de áudio gravados com sua voz. Visando diminuir esse desconforto, a publicação do arquivo será realizada sem a identificação do locutor gravado, ou ainda, o participante poderá optar pela não publicação do arquivo gravado com sua voz, conforme marcação refente a essa opção que será devidamente formalizada pelo TCLE.

Benefícios:
Os benefícios proporcionados por esta pesquisa serão de grande valia, uma vez que disponibilizará recursos de base de texto e base de áudio para a criação de modelos acústicos e uma plataforma de reconhecimento de comandos de fala livres, de código aberto e de baixo custo, contribuindo para a popularização dessas tecnologias no âmbito das tecnologias assistivas que proporcionem maior independência e autonomia para as pessoas com mobilidade reduzida, tendo em vista sua inclusão social e digital.

Comentários e Considerações sobre a Pesquisa:
A pesquisa em tela é de grande relevância, uma vez que o foco de atuação do projeto em análise volta-se para o desenvolvimento de interface de comandos por voz que seja facilmente adaptada e incorporada aos já existentes sistemas e ferramentas de auxílio ao controle do ambiente doméstico (domótica). Esses sistemas atuam fundamentalmente permitindo o controle remoto das facilidades do lar por indivíduos que apresentam alguma restrição motora ou dificuldade quanto a sua mobilidade e representam, assim, formas de auxiliar no cotidiano dessas pessoas dando suporte à execução de tarefas como: controlar eletrodomésticos, iluminação ambiente, dispositivos de cuidados com a saúde, dentre outros. O uso do reconhecimento de voz para a interação com tais facilidades, de modo a superar a exigência de habilidades de coordenação motora para a entrada de comandos aos referidos sistemas é um exemplo claro da utilização de interfaces alternativas de controle. Os compromissos a serem atingidos envolvem, portanto, o atendimento de critérios de portabilidade e mobilidade, exigindo que as interfaces propostas apresentem um caráter multifuncional, através de uma estrutura embarcável capaz de prover a entrada de comandos de voz para a tomada de decisões de controle.
Ressaltamos que o autor considera alguns aspectos éticos, recomendados na Resolução 466/2012 do Conselho Nacional de Saúde - CNS, que regulamenta as pesquisas envolvendo seres humanos

no cenário brasileiro apresentando o Termo de Consentimento Livre e Esclarecido – TCLE direcionado aos participantes da pesquisa, respeitando sua autonomia, e define forma clara e adequada os critérios de inclusão/exclusão utilizados na pesquisa.

Considerações sobre os Termos de apresentação obrigatória:

Folha de Rosto presente, devidamente preenchida e assinada pelo Diretor Geral da instituição proponente;

- TCLE presente e devidamente elaborado, conforme estabelecido pela Resolução nº466/2012 do Conselho Nacional de Saúde - CNS;
- Critérios de inclusão/exclusão claros e adequados;
- Metodologia de Coleta de Dados claramente definida e adequada à pesquisa;
- Cronograma de atividades presente;
- Orçamento Financeiro presente.

Recomendações:

Não há recomendações a serem feitas.

Conclusões ou Pendências e Lista de Inadequações:

Após avaliação do parecer apresentado pelo relator, o Comitê de Ética em Pesquisa do IFPB discutiu sobre os diversos pontos da análise ética que preconiza a Resolução 466/2012 do Conselho Nacional de Saúde e deliberou o parecer de APROVADO para o referido protocolo de pesquisa.

Informamos ao pesquisador responsável que observe as seguintes orientações:

1- O participante da pesquisa tem a liberdade de recusar-se a participar ou retirar seu consentimento em qualquer fase da pesquisa, sem penalização alguma e sem prejuízo ao seu cuidado (Res. CNS 466/2012 - Item IV.3.d).

2- O Termo de Consentimento Livre e Esclarecido dever ser elaborado em duas vias, rubricadas em todas as suas páginas e assinadas, ao seu término, pelo convidado a participar da pesquisa, ou por seu representante legal, assim como pelo pesquisador responsável, ou pela(s) pessoa(s) por ele delegada(s), devendo as páginas de assinaturas estar na mesma folha. Em ambas as vias deverão constar o endereço e contato telefônico ou outro, dos responsáveis pela pesquisa e do CEP local e da CONEP, quando pertinente (Res. CNS 466/2012 - Item IV.5.d) e uma das vias

entregue ao participante da pesquisa.

3- O pesquisador deve desenvolver a pesquisa conforme delineada no protocolo aprovado e descontinuar o estudo somente após análise das razões da descontinuidade por parte do CEP que aprovou (Res. CNS 466/2012 - Item III.2.u), aguardando seu parecer, exceto quando perceber risco ou dano não previsto ao sujeito participante ou quando constatar a superioridade de regime oferecido a um dos grupos da pesquisa (Item V.4) que requeiram ação imediata.

4- O CEP deve ser informado de todos os efeitos adversos ou fatos relevantes que alterem o curso normal do estudo (Res. CNS 466/2012 Item V.5).

5- Eventuais modificações ou emendas ao protocolo devem ser apresentadas ao CEP de forma clara e sucinta, identificando a parte do protocolo a ser modificada e suas justificativas.

6- Deve ser apresentado ao CEP relatório final até 22/01/2016.

Considerações Finais a critério do CEP:

Este parecer foi elaborado baseado nos documentos abaixo relacionados:

Tipo Documento	Arquivo	Postagem	Autor	Situação
Informações Básicas do Projeto	PB_INFORMAÇÕES_BÁSICAS_DO_PROJETO_605890.pdf	26/10/2015 20:35:17		Aceito
Projeto Detalhado / Brochura Investigador	ProjetoDetalhado_v3.pdf	26/10/2015 20:31:41	Victor Costa de Andrade Pimentel	Aceito
Cronograma	Cronogramav2.png	26/10/2015 20:29:54	Victor Costa de Andrade Pimentel	Aceito
Outros	ComandosCenarioValidacaoPlataforma.png	21/10/2015 16:22:08	Victor Costa de Andrade Pimentel	Aceito
Outros	CenarioValidacaoPlataforma.png	21/10/2015 16:20:29	Victor Costa de Andrade Pimentel	Aceito
Outros	ListaDePrompts.pdf	21/10/2015 16:19:12	Victor Costa de Andrade Pimentel	Aceito
TCLE / Termos de Assentimento / Justificativa de Ausência	TCLE.pdf	21/10/2015 16:16:06	Victor Costa de Andrade Pimentel	Aceito
Folha de Rosto	folhaDeRosto_Assinada.PDF	21/10/2015 16:14:01	Victor Costa de Andrade Pimentel	Aceito

Situação do Parecer:
Aprovado

Necessita Apreciação da CONEP:
Não

JOAO PESSOA, 04 de Dezembro de 2015

Assinado por:
Aleksandro Guedes de Lima
(Coordenador)

Endereço: Avenida João da Mata, 256 - Jaguaribe
Bairro: Jaguaribe **CEP:** 58.015-020
UF: PB **Município:** JOAO PESSOA
Telefone: (83)3612-9725 **E-mail:** eticaempesquisa@ifpb.edu.br

ANEXO H – Artigo publicado no XXIV CBEB.

UMA PLATAFORMA DE BAIXO CUSTO COMANDADA POR VOZ PARA TECNOLOGIAS ASSISTIVAS COM PROGRAMAÇÃO EM PYTHON

V. C. A. Pimentel*, I. L. Barbacena*, S. E. N. Correia* e M. C. Dias**

*Programa de Pós-Graduação em Engenharia Elétrica, Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, João Pessoa, Brasil
** Coordenação do Curso Superior de Bacharelado em Engenharia Elétrica, Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, João Pessoa, Brasil
e-mail: victor.andrade@ifrn.edu.br

Resumo: Descreve o projeto de uma plataforma básica para o reconhecimento de comandos de voz em sistemas embarcados que proporciona uma redução do custo e do tempo de desenvolvimento de tecnologias assistivas controladas por voz. A plataforma consiste de um módulo de reconhecimento de voz (*Voice Recognition Module V2*) e uma plataforma de desenvolvimento *BeagleBone Black*. Inicialmente foi desenvolvida uma aplicação para o controle de um led RGB utilizando o sistema operacional Ubuntu Linux configurado com o ambiente de desenvolvimento para a linguagem Python. Destaca-se a potencialidade de se estender a utilização do dispositivo para numerosas aplicações que possam auxiliar pessoas com mobilidade reduzida.
Palavras-chave: Sistemas embarcados, reconhecimento de comandos de voz, tecnologias assitivas.

Abstract: *Describes the design of a basic platform for recognizing voice commands in embedded systems to provide a reduction in costs as well as reduce the development time of assistive technologies controlled by voice. The platform consists of a voice recognition module (Voice Recognition Module V2) and a development platform BeagleBone Black. Initially an application was developed to control an RGB led using the Ubuntu Linux operating system configured with the development environment for the Python language. Highlights the potential to extend the use of the device for numerous applications that can assist people with disabilities.*
Keywords: *Embedded systems, recognizing voice commands, assistive technologies.*

Introdução

A busca por maior independência e autonomia para pessoas com deficiência tem se apresentado como fator decisivo na melhoria da qualidade de vida desses indivíduos. O desenvolvimento de tecnologias assistivas visa prover melhoramentos na forma como se comunicam, em sua mobilidade, em passarem a ter um maior controle sobre o ambiente onde estão inseridos e aprimorar suas experiências de aprendizado e interação social [1].

Dispositivos e sistemas controlados por voz têm sido empregados em diversas soluções [2, 3, 4, 5, 6, 7, 8].

Neste artigo é apresentado um projeto de integração entre um módulo de reconhecimento de voz (*Voice Recognition Module V2*) [9] e uma plataforma de desenvolvimento para sistemas embarcados (*BeagleBone Black*) [10], ambos de baixo custo.

Pretende-se disponibilizar a arquitetura flexível e alta capacidade de processamento da referida plataforma para o tratamento e tomada de decisões, a partir de comandos por voz, permitindo a utilização dessa interface no desenvolvimento de sistemas que auxiliem no cotidiano de pessoas com mobilidade reduzida como, por exemplo, idosos, paraplégicos ou, até mesmo, tetraplégicos, proporcionando a esses indivíduos maior sensação de segurança, controle e independência com o uso de tecnologias assistivas abertas (*open source*) de baixo custo que ajudem em tarefas como controlar eletrodomésticos, iluminação ambiente, dispositivos de cuidados com a saúde, etc.

Materiais e métodos

A Figura 1 ilustra o diagrama em blocos da interface implementada, destacando-se que o módulo de reconhecimento de voz processa o áudio, enquanto a plataforma *BeagleBone Black* recebe os sinais digitais referentes a cada comando, podendo utilizá-los nas mais diversas aplicações.

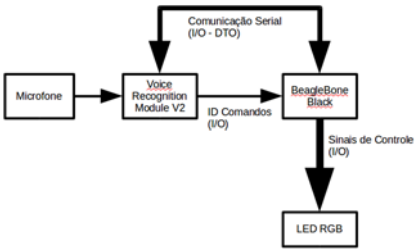


Figura 1: Diagrama em blocos da plataforma para comandos de voz.

Para validar o reconhecimento dos comandos de voz executados pela plataforma, foi montado um circuito básico para o controle das luzes de um *led RGB*, tendo sido produzido um vídeo demonstrativo do funcionamento do dispositivo, conforme referenciado na Figura 2.

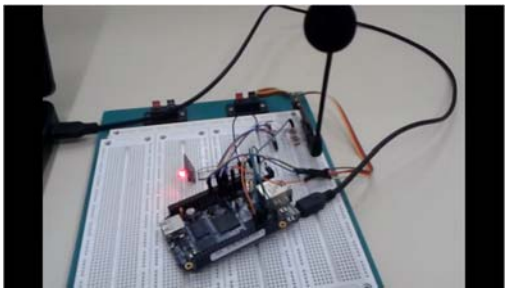


Figura 2: *Frame* do vídeo demonstrativo do funcionamento da plataforma [11].

BeagleBone Black – O *BeagleBone Black* se constitui numa plataforma de desenvolvimento que conta com uma vasta comunidade de suporte.

Dentre suas especificações, possui um processador AM335x 1GHz ARM® Cortex-A8, 512MB DDR3 RAM, memória flash interna de 4GB, acelerador gráfico 3D, dentre outras funcionalidades e recursos que ampliam seu potencial para o desenvolvimento de aplicações de sistemas embarcados [10]. Além disso, sua compatibilidade com diversos sistemas operacionais *open source* – como Debian, Ubuntu, Android e outros – e por se tratar de uma plataforma *open hardware*, apresenta uma alta flexibilidade quanto às ferramentas de desenvolvimento, podendo adicionar facilmente novos módulos de *hardware*, abrindo oportunidades para o desenvolvimento das mais diversas aplicações. Com alto potencial para sistemas ubíquos e pervasivos, ou ainda alinhados ao conceito de Internet das Coisas (*Internet of Things* – IoT).

A referida plataforma foi configurada com o sistema operacional Ubuntu Linux, que possui em sua instalação padrão as ferramentas de desenvolvimento para a linguagem Python.

Foi necessário ainda, a configuração de uma biblioteca para o acesso aos pinos de entrada e saída da plataforma *BeagleBone* utilizando a linguagem Python.

Com o ambiente adequadamente preparado, foi elaborado um código, desenvolvido em linguagem Python, para a recepção dos sinais provindos do módulo de reconhecimento de voz, que identificam cada comando e geram uma saída correspondente ao comando de voz acionado, de modo a controlar o acendimento ou desligamento das luzes do *led RGB*.

Voice Recognition Module V2 – É um módulo de reconhecimento de comandos de voz, dependente de locutor, de baixo custo, e que pode ser utilizado para controlar os sistemas num carro ou outros dispositivos eletroeletrônicos.

É capaz de armazenar até 15 comandos de voz, os quais são organizadas em 3 grupos. Assim, é necessário antes realizar o treinamento dos comandos de voz no referido módulo. Cada grupo de comandos só pode ser ativado individualmente, de modo que somente os 5 comandos do grupo atual serão disponibilizados.

Dessa forma, para o teste de validação da plataforma, foram gravados somente 5 comandos, conforme a Tabela 1.

Tabela 1: Comandos para validação da plataforma.

Comando de voz	**Pino**	**Resposta esperada do sistema**
/rubro/	O1	Acendimento da luz vermelha
/verde/	O2	Acendimento da luz verde
/azul/	O3	Acendimento da luz azul
/branco/	O4	Acendimento das luzes vermelha, verde e azul
/apaga/	O5	Desliga as luzes do led

A realização do treinamento do módulo, bem como ativação e desativação de seus modos de funcionamento é realizada enviando-se comandos via interface serial, conforme ilustrado na Figura 1. Além disso, através dessa mesma interface, é selecionado o grupo de comandos que deverá estar ativo num dado momento. Disponibiliza ainda a opção de que o grupo de comandos seja carregado a partir da utilização de pinos de seleção (GCH e GCL), conforme o manual [9].

As memórias internas onde os comandos são gravados durante o treinamento suportam elocuções de, no máximo, 1300ms, de modo que o sistema somente suporta comandos de palavras isoladas.

Por questões de diferença entre os níveis de tensão gerado pelas saídas do módulo de reconhecimento de voz (5V) e o nível suportado pelos pinos de entrada do *BeagleBone* (3,3V), foram utilizados divisores de tensão, evitando a queima das portas na plataforma. A Equação 1 apresenta o cálculo para dimensionamento dos divisores de tensão. Foram utilizados resistores comerciais de 330Ω e 1KΩ.

$$R_2 = \frac{-V_2 R_1}{V_2 - V_1} \quad (1)$$

Para o acionamento das luzes do *led* RGB, foram utilizados resistores limitadores de corrente (220Ω cada).

Testes realizados – De acordo com o manual do módulo *Voice Recognition Module V2* [9], sua precisão quanto ao reconhecimento de comandos num ambiente ideal é de 99%.

No entanto, mostra-se necessário sua verificação em um ambiente ruidoso, em que o sistema será naturalmente utilizado. Para tanto, foram realizados testes em um ambiente com nível de ruído na faixa de 45 a 65dB, com vistas a verificar a precisão do sistema quanto ao reconhecimento de comandos num contexto próximo de uma situação real de utilização.

Foram convidados 10 locutores para a os testes de validação, em seções individuais, sendo 5 locutores do sexo feminino e outros 5 do sexo masculino. Cada um

dos locutores realizou o treinamento do módulo de reconhecimento de voz com os comandos da Tabela 1.

Posteriormente, a execução do código implementado em linguagem Python realiza o envio de um comando no formato "*Head + Key*" [9], em hexadecimal, ao módulo de reconhecimento de voz, através da interface serial, fazendo com que o referido módulo importe o grupo de instruções de teste, definidas na etapa de treinamento, e permaneça no estado pronto para recebê-las.

Nesse estado de operação, cada locutor forneceu 20 repetições para cada um dos comandos de voz através do microfone, tendo sido realizada a execução de 1000 comandos por diferentes indivíduos, observando-se o comportamento do sistema, conforme a Tabela 1.

Resultados

A Tabela 2 apresenta a taxa de acertos por locutor, bem como a taxa média de acertos, relacionada à totalidade dos 1000 comandos executados.

Tabela 2: Resultados dos testes realizados

Locutor	**Taxa de acertos (%)**
Locutor 1	96%
Locutor 2	97%
Locutor 3	98%
Locutor 4	90%
Locutor 5	97%
Locutor 6	92%
Locutor 7	97%
Locutor 8	95%
Locutor 9	97%
Locutor 10	100%
Taxa média de acertos	95,9%

Discussão

Os resultados permitem concluir que, em ambientes ruidosos, a taxa média global de acertos (de 95,9%) é bastante alta, garantindo a validação do sistema. No entanto, durante a realização dos testes por cada locutor, ocorreram momentos isolados em que o comportamento desviou-se do esperado. Por exemplo, o acendimento do *led* verde para o locutor 5, não funcionou como esperado em 25% das vezes. Além disso, certa quantidade de vezes, o sistema permaneceu inoperante após ter sido fornecido o comando, representando 3,5% das vezes.

A escolha de comandos com sonoridade semelhante (homófonos) aumenta a ocorrência de acionamento indevido dos *leds*, podendo dois comandos resultar na mesma resposta. Há ainda relação entre a ocorrência de erros e o cansaço vocal do locutor, tendo sido as repetições dos comandos realizadas em sequência. Essas questões foram também discutidas em [12].

Não raras vezes, o sistema foi acionado acidentalmente, influenciado pelo ruído provindo de conversas de terceiros no ambiente de teste (situação próxima da real). Assim, as aplicações desenvolvidas utilizando o dispositivo proposto devem apresentar mecanismos que tratem questões de segurança, fornecendo modalidades de entrada diversas para ativar ou desativar o mecanismo comandado por voz [7].

A possibilidade de treinamento utilizando a língua português brasileiro permite sua ampla aceitação no país, uma vez que os usuários com mobilidade reduzida, como idosos ou indivíduos com lesão na coluna vertebral, paraplégicos ou tetraplégicos, tendem a preferir utilizar sua própria língua no controle por voz de dispositivos [5].

Um dispositivo ubíquo controlado por voz na língua árabe para ajudar pessoas com deficiência foi implementado em [5]. O autor apresenta 3 casos de teste utilizando o dispositivo projetado: o controle de mouse e teclado, o controle de cadeira de rodas e o controle de braço robótico industrial. A unidade de controle utilizada foi um microcontrolador Atmega8, da Atmel.

A utilização de microcontrolador é uma prática recorrente para esses dispositivos, conforme observado também em [2]. No entanto, a utilização de unidades de controle baseadas em computador, tendo um módulo externo específico responsável pela tarefa de realizar o reconhecimento de comandos por voz também vem sendo observada na literatura [3], onde a unidade de reconhecimento de voz se constitui numa placa que funciona como "*plug-in*" para o PC.

Essa prática tem o intuito de retirar o pesado processamento de áudio do computador (*Beaglebone Black*), de modo que os comandos de voz sejam utilizados somente para sinalizar tarefas, liberando a capacidade de processamento da plataforma para a utilização de recursos avançados, que podem exigir alta capacidade de processamento, como, por exemplo, a associação de diversas outras modalidades de entrada, permitindo o desenvolvimento de aplicações com interfaces multimodais [6, 13], de modo a aperfeiçoar a experiência dos usuários com mobilidade reduzida; como também o controle de diversos tipos de dispositivos utilizando protocolos de comunicação avançados, característica concernente ao caráter de uso multidisciplinar e estrutura heterogênea dos sistemas inteligentes de automação, podendo ainda envolver questões de segurança [8].

Nesse sentido, a grande flexibilidade e compatibilidade com as mais variadas tecnologias proporcionadas com a utilização da plataforma apresentada permitem que sua alta capacidade de processamento esteja disponível para as aplicações que demandem recursos avançados, fazendo com que os sinais enviados pela plataforma para o controle do acendimento e desligamento das luzes do *led RGB* possam ser adaptados para comandar os mais diversos tipos de dispositivos, otimizando o desenvolvimento rápido de aplicações inicialmente proposto. Além disso, o baixo custo viabiliza ainda mais sua utilização no

desenvolvimento de soluções acessíveis. A Tabela 2 apresenta os custos relacionados à plataforma.
Tabela 2: Custos dos componentes do projeto.

Item	Qtd.	Valor em 14/07/2014 (R$)
BeagleBone Black	1	121,76
Voice Recognition Module V2	1	52,03
Resistor 220 Ω	3	0,30
Resistor 330 Ω	5	0,50
Resistor 1K Ω	5	0,50
Total		175,09

Assim, além de possibilitar o desenvolvimento de aplicações que envolvam maior complexidade, se atinge também uma redução nos custos dos dispositivos, o que é percebido quando comparado ao sistema apresentado por [4], em que, somente a interface de reconhecimento de voz, gera custos da ordem de $329,98 dólares (R$730,54 em 14 de Julho de 2014), o que sinaliza uma redução de gastos de cerca de 76%. Com relação ao dispositivo apresentado em [2], que tem uma previsão de custos da ordem de $200 dólares (R$ 447,20 em 04 de Setembro de 2014), o dispositivo aqui proposto apresenta-se 60% mais barato.

Conclusão

É apresentada uma interessante alternativa para o desenvolvimento rápido de tecnologias assistivas. O pequeno tamanho de seus componentes permite que seja desenvolvido um encapsulamento que possa ser adaptável a fixação em locais como uma cabeceira de cama ou adaptado a uma cadeira de rodas. O baixo custo dessa interface de reconhecimento de comandos de voz contribui para reduzir o preço final das aplicações que venham a ser desenvolvidas. Os resultados dos testes realizados apontam a confiabilidade necessária para sua adoção.

Convém estender a utilização dessa solução para aplicações com entradas multimodais para o controle de iluminação do ambiente, controle de aparelhos eletrodomésticos, integrando-os com mecanismos de segurança, bem como dispositivos domésticos de cuidados com a saúde (*Home Health Care*).

Referências

[1] Bersch R. Introdução à Tecnologia Assistiva. Assistiva® Tecnologia e Educação. Porto Alegre, 2013 [cited 2014 Set 3]. Available from: http://www.assistiva.com.br/Introducao_Tecnologia_Assistiva.pdf.

[2] Jiang H, Han Z, Success P, Robidoux S, Sun Y. Voice-Activated Environmental Control System for Persons with Disabilities. In: Proceedings of the IEEE 26th Annual Northeast Bioengineering Conference; 2000 Sep 08-09; Storrs, Connecticut, Estados Unidos. 2000. p. 167-8.

[3] Aguilera F, Ataaefar A, Brothers R, Castellano M, Ginart A, Grangeia G, La L, McCrae W, McKernan M, Moonsammy A, Noel Y, Padillha J, Patel B, Reyes R, Ryan J, Rydzak T, Shan J, Sinha T, Valatkavage J, Zelano JA, Members, IEEE. A Personal Computer Based Environmental Control System For The Disabled. In: 14th Annual International Conference of the Engineering in Medicine and Biology Society; 1992 Oct 29 – Nov 1; Paris, França. 1992. p. 1533-4.

[4] Hall B, Molloy J. Installing A Voice Activated Environmental Control Unit For Under 500 Dollars. RESNA 26th International Annual Conference [internet]. 2003 Jun [cited 2014 Jul 14]. Available from:http://web.resna.org/conference/proceedings/2003/Papers/EA/Hall_EA.htm.

[5] Shakir, U. Ubiquitous Arabic Voice Control Device To Assist People With Disabilities. In: 4th Conference on Intelligent and Advanced Systems; 2012 Jun 12 – 14; Kuala Lumpur, Malaysia. 2012. p. 333-8.

[6] Vallés M, Manso F, Arredondo T, Pozo F. Multimodal Environmental Control System For Elderly And Disabled People. In: 18th International Conference of The IEEE Engineering in Medicine And Biology Society; 1996 Oct 31 – Nov 3; Amsterdam, The Netherlands. 1996. p. 516-7.

[7] Lu Y, Chen Y. Prototyping Potential Control Systems to Assist Complete Quadriplegics. In: The Biomedical Engineering International Conference; 2012 Dec 5-7. p. 1-5.

[8]Uzunai Y, Bicakci K. SHA: A Secure Voice Activated Smart Home For Quadriplegia Patients. In: IEEE International Conference on Bioinformatics and Biomedicine Workshops; 2007 Nov 2-4. p. 151-8.

[9] Shen W. Voice Recognition Module V2. Elechouse [internet]. 2013 Mar [cited 2014 Jul 12]. Available from:http://www.elechouse.com/elechouse/images/product/Voice%20Recognition%20Module/Manual.pdf.

[10] Beagleboard. BeagleBone Black. Beagleboard [internet]. 2014 [cited 2014 Jul 12]. Available from: http://beagleboard.org/black.

[11] Pimentel V. Reconhecimento de Comandos por Voz. [internet]. 2014 Sep. Available from: https://www.youtube.com/watch?v=53tq-M9TslQ

[12] Silva L. Reconhecimento de Voz Utilizando o Kit SR07 HM2007 [trabalho de conclusão de curso]. João Pessoa: Instituto Federal de Educação, Ciência e Tecnologia da Paraíba; 2013.

[13] Hui P, Meng H. Latent Semantic Analisys for Multimodal User Input With Speech and Gestures. In: IEEE/ACM Transactions On Audio, Speech, And Language Processing, Vol. 22, n. 2, 2014 Feb. p. 417-29.

ANEXO I – Planilhas com dados dos testes da interface baseada em ferramentas de *software* integradas a servidor de comunicação.

Locutor 1 (voz masculina)				
Ambiente	**Comando**	**Acerto**	**Falso positivo**	**Taxa acerto (%)**
Bathroom	Light On	19	1	95,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	19	1	95,00
Kitchen	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
External	Light On	18	2	90,00
	Light Off	20	0	100,00
Chamber	Light On	17	3	85,00
	Light Off	14	6	70,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	17	3	85,00
	TV Off	18	2	90,00
Hall	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	19	1	95,00
	TV Off	20	0	100,00
	Taxa média de acertos			95,68

Locutor 2 (voz masculina)				
Ambiente	**Comando**	**Acerto**	**Falso positivo**	**Taxa acerto (%)**
Bathroom	Light On	20	0	100,00
	Light Off	19	1	95,00
	Door Open	14	6	70,00
	Door Close	20	0	100,00
Kitchen	Light On	18	2	90,00
	Light Off	19	1	95,00
	Door Open	18	2	90,00
	Door Close	19	1	95,00
External	Light On	19	1	95,00
	Light Off	17	3	85,00
Chamber	Light On	19	1	95,00
	Light Off	18	2	90,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	17	3	85,00
	TV Off	18	2	90,00
Hall	Light On	18	2	90,00
	Light Off	16	4	80,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	18	2	90,00
	TV Off	17	3	85,00
	Taxa média de acertos			91,82

Locutor 3 (voz feminina)				
Ambiente	**Comando**	**Acerto**	**Falso positivo**	**Taxa acerto (%)**
Bathroom	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
Kitchen	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
External	Light On	12	8	60,00
	Light Off	9	11	45,00
Chamber	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	17	3	85,00
	Door Close	15	5	75,00
	TV On	19	1	95,00
	TV Off	20	0	100,00
Hall	Light On	19	1	95,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	18	2	90,00
	TV On	20	0	100,00
	TV Off	20	0	100,00
	Taxa média de acertos			92,95

Locutor 4 (voz feminina)				
Ambiente	**Comando**	**Acerto**	**Falso positivo**	**Taxa acerto (%)**
Bathroom	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
Kitchen	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	13	7	65,00
External	Light On	15	5	75,00
	Light Off	15	5	75,00
Chamber	Light On	12	8	60,00
	Light Off	15	5	75,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	20	0	100,00
	TV Off	20	0	100,00
Hall	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	17	3	85,00
	Door Close	19	1	95,00
	TV On	20	0	100,00
	TV Off	20	0	100,00
	Taxa média de acertos			92,27

Locutor 5 (voz feminina)				
Ambiente	**Comando**	**Acerto**	**Falso positivo**	**Taxa acerto (%)**
Bathroom	Light On	19	1	95,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
Kitchen	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
External	Light On	15	5	75,00
	Light Off	15	5	75,00
Chamber	Light On	16	4	80,00
	Light Off	19	1	95,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	20	0	100,00
	TV Off	20	0	100,00
Hall	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	20	0	100,00
	TV Off	20	0	100,00
	Taxa média de acertos			96,36

Locutor 6 (voz masculina)				
Ambiente	**Comando**	**Acerto**	**Falso positivo**	**Taxa acerto (%)**
Bathroom	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
Kitchen	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	18	2	90,00
	Door Close	20	0	100,00
External	Light On	19	1	95,00
	Light Off	17	3	85,00
Chamber	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	20	0	100,00
	TV Off	20	0	100,00
Hall	Light On	20	0	100,00
	Light Off	19	1	95,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	20	0	100,00
	TV Off	20	0	100,00
	Taxa média de acertos			98,41

Locutor 7 (voz feminina)				
Ambiente	**Comando**	**Acerto**	**Falso positivo**	**Taxa acerto (%)**
Bathroom	Light On	19	1	95,00
	Light Off	15	5	75,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
Kitchen	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	18	2	90,00
	Door Close	19	1	95,00
External	Light On	13	7	65,00
	Light Off	19	1	95,00
Chamber	Light On	17	3	85,00
	Light Off	20	0	100,00
	Door Open	19	1	95,00
	Door Close	19	1	95,00
	TV On	20	0	100,00
	TV Off	20	0	100,00
Hall	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	19	1	95,00
	Door Close	16	4	80,00
	TV On	19	1	95,00
	TV Off	20	0	100,00
	Taxa média de acertos			93,64

Locutor 8 (voz masculina)				
Ambiente	**Comando**	**Acerto**	**Falso positivo**	**Taxa acerto (%)**
Bathroom	Light On	19	1	95,00
	Light Off	18	2	90,00
	Door Open	19	1	95,00
	Door Close	19	1	95,00
Kitchen	Light On	20	0	100,00
	Light Off	15	5	75,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
External	Light On	13	7	65,00
	Light Off	10	10	50,00
Chamber	Light On	19	1	95,00
	Light Off	18	2	90,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	20	0	100,00
	TV Off	20	0	100,00
Hall	Light On	20	0	100,00
	Light Off	17	0	85,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	20	0	100,00
	TV Off	20	0	100,00
	Taxa média de acertos			92,50

Locutor 9 (voz feminina)				
Ambiente	**Comando**	**Acerto**	**Falso positivo**	**Taxa acerto (%)**
Bathroom	Light On	20	0	100,00
	Light Off	19	1	95,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
Kitchen	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
External	Light On	17	3	85,00
	Light Off	16	4	80,00
Chamber	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	20	0	100,00
	TV Off	20	0	100,00
Hall	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	19	0	95,00
	TV On	18	0	90,00
	TV Off	18	0	90,00
	Taxa média de acertos			97,05

Locutor 10 (voz masculina)				
Ambiente	**Comando**	**Acerto**	**Falso positivo**	**Taxa acerto (%)**
Bathroom	Light On	19	1	95,00
	Light Off	18	2	90,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
Kitchen	Light On	20	0	100,00
	Light Off	19	1	95,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
External	Light On	20	0	100,00
	Light Off	20	0	100,00
Chamber	Light On	20	0	100,00
	Light Off	19	1	95,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	18	2	90,00
	TV Off	14	6	70,00
Hall	Light On	20	0	100,00
	Light Off	20	0	100,00
	Door Open	20	0	100,00
	Door Close	20	0	100,00
	TV On	20	0	100,00
	TV Off	20	0	100,00
	Taxa média de acertos			97,05

ANEXO J – Parecer do comitê de ética aprovando o relatório final da pesquisa

PARECER CONSUBSTANCIADO DO CEP

DADOS DO PROJETO DE PESQUISA

Título da Pesquisa: Plataforma embarcada de reconhecimento da fala, baseada em Beaglebone e Asterisk, para o auxílio de pessoas com mobilidade reduzida.

Pesquisador: Victor Costa de Andrade Pimentel

Área Temática:

Versão: 1

CAAE: 50486815.6.0000.5185

Instituição Proponente: Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - IFPB

Patrocinador Principal: Financiamento Próprio

DADOS DA NOTIFICAÇÃO

Tipo de Notificação: Envio de Relatório Final

Detalhe:

Justificativa: Envio relatório final, tendo sido concluídas as etapas de gravação das amostras de

Data do Envio: 09/01/2016

Situação da Notificação: Parecer Consubstanciado Emitido

DADOS DO PARECER

Número do Parecer: 1.397.087

Apresentação da Notificação:

Trata-se da apresentação do relatório final de pesquisa realizada.

Segundo o pesquisador responsável, de acordo com os objetivos do estudo, foi desenvolvida uma plataforma de reconhecimento automático da fala (RAF), livre e de baixo custo, com o propósito de ser empregada no controle de tecnologias assistivas a partir da interação humano-computador através da pronúncia de comandos pelo usuário. Ainda segundo o autor "O intuito é que essa ferramenta se constitua numa interface de comandos por voz que seja facilmente adaptada e incorporada aos já existentes sistemas e dispositivos de auxílio ao controle do ambiente doméstico (domótica)".

No relatório o pesquisador responsável informa que "... tendo em vista a necessidade de se avaliar o funcionamento da referida plataforma, realizou-se a gravação de amostras de áudio de

treinamento com a voz de 10 (dez) indivíduos, dentre cidadãos brasileiros comuns, tendo sido 5 (cinco) do sexo masculino e 5 (cinco) do sexo feminino, conforme a metodologia proposta no projeto detalhado".
No detalhamento dos procedimentos, o autor relata:
- "Foi enfrentada uma certa dificuldade quanto a compatibilização de horários para a realização das sessões com os participantes, de modo que se pôde observar uma grande preferência pela realização de uma quantidade mínima de encontros possível. Não houveram maiores empecilhos, de modo que as gravações e testes ocorreram com sucesso, tendo despertado o interesse e a curiosidade dos participantes, que manifestaram disposição em contribuir, inclusive, com a disponibilização dos arquivos de áudio gravados com sua voz publicamente no projeto Voxforge, conforme opção manifestada no TCLE, sendo que somente 1 (um) dos participantes optou por não permitir a publicação, tendo sido respeitada sua escolha".
- "Os dados coletados durante a execução do procedimento avaliativo da plataforma foram registrados em planilhas, sendo registrado o comportamento do sistema para a entrada de 20 (vinte) repetições de cada um dos comandos constantes no diagrama" apresentado.
- "Com o intuito de observar o compromisso de não identificar o nome dos participantes na divulgação dos resultados da pesquisa, foi atribuída uma nomenclatura referindo-se de cada um como sendo do Locutor “X”, em que “X” é um número que varia de 1 (um) a 10 (dez)".

Objetivo da Notificação:

Relatar sobre as atividades desenvolvidas na pesquisa realizada, com enfoque nos procedimentos metodológicos, do quantitativo efetivo de participantes e documentação utilizada, objetivos, coleta de dados e resultados obtidos.

Avaliação dos Riscos e Benefícios:

Segundo o relato do pesquisador principal, o estudo foi desenvolvido conforme delineado no projeto detalhado e respeitando as decisões dos participantes quanto a utilização dos dados obtidos.
No tocante aos benefícios, o pesquisador responsável informa os objetivos foram alcançados e desta forma será possível de fato alcançar os benefícios esperados com o sucesso do estudo.

Comentários e Considerações sobre a Notificação:

Conforme já previsto a pesquisa se mostrou de relevância, uma vez que o foco de atuação do projeto voltou -se para o desenvolvimento de interface de comandos por voz que seja facilmente

adaptada e incorporada aos já existentes sistemas e ferramentas de auxílio ao controle do ambiente doméstico (domótica).

Nas conclusões do relatório o pesquisador responsável informou que: "...pode-se concluir que o sistema proposto apresenta elevado potencial de aplicação, configurando-se numa plataforma portável e flexível, que pode ser facilmente integrada a outras tecnologias, por ser baseada em recursos de open hardware e open source, além de proporcionar comunicação através do procotolo da Internet (IP), desempenha seu propósito alcançando uma taxa média total de acertos da ordem de 94,77%, o que representa um funcionamento coerente para 95 de cada 100 comandos executados...".

"...Os dados obtidos com a pesquisa realizada fornecem base para justificar maior investimento na plataforma, uma vez que a qualidade do seu funcionamento pode ser ainda amplificada a partir de soluções simples, como a observação de um maior cuidado questões relacionadas somente ao volume das gravações de amostras de treinamento, como também a partir de sua adaptação para a utilização de comandos no idioma português, tendo sido demonstrada sua viabilidade".

Considerações sobre os Termos de apresentação obrigatória:

Como documentos foi apresentado inicialmente o relatório final descrevendo os procedimentos realizados na pesquisa, mas não foram inseridos os documentos relatados no relatório como anexos (TCLEs assindados pelos participantes e pesquisador responsável). No entanto, após contato do CEP com o pesquisador, os anexos foram enviados via e-mail para o CEP e estes foram adicionados à notificação pelo coordenador do CEP.

Recomendações:

Não há.

Conclusões ou Pendências e Lista de Inadequações:

Após análise do relatório final do estudo realizado, verifica-se que a pesquisa se desenvolveu conforme a metodologia planejada, observando os cuidados para preservar os direitos dos participantes e tomando os cuidados para minimizar os riscos destes com suas participações no estudo.

Entendemos que a pesquisa foi desenvolvida em acordo com o que preconiza a Resolução n. 466/2012 do Conselho Nacional de Saúde. Logo, o colegiado do CEP deliberou pela aprovação do relatório final apresentado.

Este parecer foi elaborado baseado nos documentos abaixo relacionados:

Tipo Documento	Arquivo	Postagem	Autor	Situação
Envio de Relatório Final	RelatorioFinal_v1.pdf	09/01/2016 04:57:27	Victor Costa de Andrade Pimentel	Aceito

Situação do Parecer:
Aprovado

Necessita Apreciação da CONEP:
Não

JOAO PESSOA, 28 de Janeiro de 2016

Assinado por:
Aleksandro Guedes de Lima
(Coordenador)

Endereço: Avenida João da Mata, 256 - Jaguaribe
Bairro: Jaguaribe **CEP:** 58.015-020
UF: PB **Município:** JOAO PESSOA
Telefone: (83)3612-9725 **E-mail:** eticaempesquisa@ifpb.edu.br