Plataforma de *Hardware* e *Software* com *Interface* de Reconhecimento de Voz

Felipe Negri[1], Leandro Camara Ledel[2]

[1]Curso de Tecnologia em Análise e Desenvolvimento de Sistemas – Câmpus Hortolândia - Instituto Federal de São Paulo (IFSP)

[2]Área de Informática – Câmpus Hortolândia - Instituto Federal de São Paulo (IFSP)

negri701@gmail.com, ledel@ifsp.edu.br

Abstract. *This paper presents a hardware and software platform with speech recognition for domestic use. The platform uses low-cost hardware, employs open-source software and adds software customized for this work. The proposed solution allows to switch on and off electronic devices from voice commands. Possible applications for this platform are the use by people with special needs, the elderly, and also ordinary users who seek for comfort and convenience in a residential setting.*

Resumo. *O presente artigo apresenta uma plataforma de hardware e software com reconhecimento de voz para uso doméstico. A plataforma utiliza hardware de baixo custo, emprega softwares de código-fonte abertos e acrescenta software desenvolvido especificamente para este trabalho. A solução proposta permite automatizar tarefas domésticas como a ligação e o desligamento de aparelhos eletroeletrônicos a partir de comandos de voz. Possíveis aplicações para esta plataforma são o emprego junto a pessoas portadoras de necessidades especiais, idosos, enfermos ou para o conforto e comodidade de usuários comuns, em um ambiente residencial.*

1. Introdução

A área da Computação tem presenciado diversos avanços nas últimas décadas. Notadamente, um dos temas mais em voga atualmente é a Internet das Coisas (IoT - *Internet of Things*), conceito amplo que se refere ao potencial de diversos dispositivos autômatos se conectarem à Internet, além dos já conhecidos computadores pessoais, *tablets*, aparelhos telefônicos etc. Segundo Singer (2012), a Internet das Coisas também abrange outros conceitos, tais como computação ubíqua e Internet do Futuro, bem como está ligada à ideia de Cidades Inteligentes.

Este crescimento de dispositivos conectados em rede traz também novos desafios, como a capacidade de se armazenar e gerenciar as informações geradas por eles. Desta forma, surgem também novos conceitos como Big Data, uma tecnologia inserida no processo de gestão de informação a fim de organizar, processar e armazenar os dados (RIBEIRO, 2014).

Outra possibilidade decorrente do surgimento de novos dispositivos é o da computação interagir com o ambiente, de forma semelhante ao que a eletrônica já permite, porém a partir de computadores pessoais tradicionais ou de baixo custo. Tal conceito é chamado de Computação Física, e expande as possibilidades de controle e

interação da computação tradicional (COMPUTAÇÃO Física BR, 2012). Novos sistemas mecânicos e eletrônicos permitem a expansão das capacidades de interação entre o homem e a máquina (CAMARATA, 2003).

Além dos avanços descritos anteriormente, também as formas com que se pode interagir com o computador evoluíram consideravelmente. As *interfaces* tradicionais entre o homem e a máquina empregavam o *mouse*, o teclado e o monitor de vídeo. Nas últimas décadas, foram acrescentadas as telas sensíveis ao toque (*touch screen*) a diversos dispositivos computacionais. Com o advento da Computação Física, pode-se prever que a interação com o computador ganhará inúmeras novas formas.

O estudo de tecnologias e recursos para o reconhecimento de voz, por exemplo, acrescentou ainda mais possibilidades à interação homem-máquina. Uma *interface* empregando comandos de voz pode ser utilizada por sistemas, os quais tanto podem ser utilizados para proporcionar maior conforto aos usuários, quanto permitir a interação a pessoas com alguma necessidade específica. Pode-se, por exemplo, automatizar uma casa pensando sob dois pontos de vista: gerar agilidade em tarefas cotidianas, como também permitir a realização de atividades diárias sem a necessidade da intervenção física do usuário no ambiente.

Amaral (2009) trata deste tema, buscando auxiliar e proporcionar mais independência a pessoas com necessidades especiais como também temporárias, como por exemplo, restrições médicas. Para isso, propõe a implantação de um sistema com agentes inteligentes, ou seja, capaz de agir sobre o ambiente por meio de sensores e atuadores. A solução proposta por Amaral, interage com o ambiente externo utilizando a porta paralela do computador.

Já Lucena (2006) propõe a soma do computador a um microcontrolador. Ela desenvolveu duas aplicações, uma para o computador e outra para o microcontrolador. A comunicação entre o computador e o microcontrolador, neste caso, utiliza portas seriais.

Em decorrência da pesquisa acerca de novas *interfaces* homem-máquina, e particularmente da interface empregando o reconhecimento de voz, identificou-se a possibilidade de proposição e criação de uma infraestrutura de *hardware* e *software*, capaz de identificar comandos de voz e de, a partir deles, interagir com o ambiente externo, permitindo assim ao usuário o controle de dispositivos elétricos e eletrônicos sem a necessidade de movimentos.

O presente trabalho tem, portanto, o objetivo de desenvolver um sistema genérico, de baixo custo, empregando reconhecimento de voz e capaz de interagir com o ambiente, o qual pode ser empregado em diversas aplicações, bem como aprimorado em futuras pesquisas acadêmicas.

2. Revisão Bibliográfica

A presente seção apresenta o resultado da pesquisa por referências e conceitos relacionados aos assuntos discutidos ao longo do trabalho. A seguir veremos um pouco dos trabalhos existentes, reconhecimento de voz, histórico e conceitos dos *hardwares* de baixo custo, bem como conceitos ligados à acessibilidade e a pessoas com necessidades específicas.

2.1. Trabalhos Correlatos

Encontrou-se diversos trabalhos que empregam o reconhecimento de voz com diferentes propósitos. Nesta seção serão apresentados os principais trabalhos correlatos, elencando suas características e possíveis limitações.

Chiele e Zerbetto (2010) propõem um cadeira de rodas controlada por voz com foco em baixo custo. Neste projeto, desenvolveram a cadeira motorizada e a aplicação. O ambiente escolhido foi um computador *desktop* com sistema operacional Windows XP e, para o reconhecimento de voz implementou o Via Voice, uma API *(Application Programming Interface),* desenvolvida pela IBM.

Para *desktop* temos diversas soluções implementadas, como o próprio mecanismo de pesquisa da Google. A maior parte das aplicações é voltada para a tecnologia assistiva. Como, o DOSVOX, criado pela Universidade Federal do Rio de Janeiro (UFRJ) voltado a atender deficientes visuais, e que permite pessoas cegas utilizarem computadores com alto nível de independência (BORGES, 2002).

Guimarães et al. (2005) propõem o Gvoicer, uma ferramenta que possibilita o desenvolvimento automático do controle de aplicações de realidade virtual por comandos de voz. Ele compõe a biblioteca libGlass, ambiente de desenvolvimento de aplicações de multiprojeção baseadas em aglomerados gráficos. Para sua implementação Guimarães utilizou a API Sphinx-4 e foi reportada a dificuldade com o suporte à língua Portuguesa.

Guilhoto e Rosa (2001) falam sobre as soluções existentes para o Reconhecimento de Voz. Eles citam as principais empresas que atuam neste contexto: Dragon System, Lernout & Hauspie (L&H), IBM e Philips. Dessas, apenas a Dragon e a IBM investiram na Língua Portuguesa. Atualmente com o avanço dos *Smartphones,* a Google, a Microsoft e a Apple voltaram a investir no mercado voltando-se aos agentes pessoais e recursos para acessibilidade, como realizar uma pesquisa por fala.

Ainda tratando de telefonia tem-se a URA, Unidade de Resposta Audível, conhecida nas centrais de atendimento a cliente. Esta, oferecendo uma gama de possibilidade programadas como selecionar opções em um menu, armazenar números, confirmar dados etc. (TOTALIP, 2013). Na década de 90 se tornou capaz de reconhecer a fala dos usuários (MIDIAVOX, 2003).

2.2. Comandos de voz

Segundo Marin e Cunha (2006) o reconhecimento de voz é "[...] Método pelo qual a linguagem natural ou convencional é registrada e o reconhecimento de fonemas é usado para identificar uma linguagem específica." Também afirmam que o reconhecimento distingue as ondas geradas pelos fonemas como também identifica-as através de um mapa de palavras, utilizando probabilidade de como as palavras são usadas em sequência.

Existem diversas API's que implementam tais conceitos. A seguir veremos as principais estudadas neste trabalho.

***Web Speech* API** - Criada em 2012 pela Google, a fim de disponibilizar aos usuários do Google Chrome pesquisas por fala como também, possibilitar aos desenvolvedores a utilização do recurso. Construída em JavaScript e assim, feita para navegadores. Esta ao reconhecer uma fala lança o texto em uma caixa de texto e pode

ser utilizada apenas no Google Chrome (W3C, 2012). É válido ressaltar que utiliza o recurso *Google Speech Recognition*, ou seja, para cada reconhecimento gera um áudio, o qual é encaminhado para o servidor da Google, que por sua vez retorna o texto correspondente. Desta forma, só pode ser utilizado se houver acesso à Internet.

Java *Speech* API (JSAPI) - Criada em 1998, desenvolvida pela Sun Microsystems, Inc. Por ser Java, suporta multiplataforma. Para reconhecer os comandos necessita de um dicionário de palavras, armazenado em um banco de dados (Oracle). A API é robusta e completa, com excelente grau de reconhecimento. Para sua utilização existem diversas implementações.

CMUSphinix - construído sobre a JSAPI, utiliza-se de Apache Marven e Grandles - ambas tecnologias JAVA - para sua implementação. Este possibilita a utilização de vários idiomas apenas instalando o idioma requerido. Os sintetizadores podem ser baixados no site da API porém, limitam-se a idiomas europeus. Suporta Windows e Android. É válido destacar que gerar um dicionário do idioma leva anos de trabalho e os idiomas já existentes são suportados por um fórum com centenas de pessoas (CMUSphinix, 2015).

Já a API **CloudGarden**, também construída em JSAPI, é voltada para Windows e utiliza a síntese em português (CloundGarden).

Speech Recognition suporta o CMUSphinix, a Google *Speech Recognition* e outras API's. Desenvolvida em Python, voltada para Linux, implementa o Google *Speech Recognition* como padrão. Esta pode implementar várias API's dando a possibilidade de utilizar várias implementações de acordo com a situação (UBERI, 2016).

2.3. *Hardwares* de baixo custo

Em 2003, com a proposta de interagir com projetos escolares, surge o **Arduino**. Com a filosofia *open-source*, ou seja, sem cobrar direitos autorais e código-aberto, de baixo custo, flexível e com linguagem de alto-nível o Arduino se tornou mundialmente conhecido e utilizado para diversos fins e projetos (ARDUINO, 2016).

Em 2008 foi possível a criação do **Raspberry Pi**, em virtude da popularização dos *Smartphones* e também com o fato de que a empresa Broadcom, produtora de semicondutores, passou a desenvolver processadores ARM - processadores de alto desempenho na escala de GHz e produzidos em baixo custo (RASPBERRY PI Foundation, 2012).

O Raspberry Pi é um microcomputador completo, pois suporta Linux, Android e Windows 10 *embedded*, com processador ARM de 900MHz, memória RAM entre 256MB e 1GB, conexão USB, HDMI e Ethernet. As versões lançadas contém placa de vídeo e de 26 a 40 pinos analógicos (Versão B, B+ e 2.0). A Figura 2 ilustra este *hardware* de baixo custo (RASPBERRY PI Foundation, 2012).

Figura 1. Foto do Raspberry Pi – Modelo 2.

(<https://www.raspberrypi.org/blog/raspberry-pi-2-on-sale/>)

Lançado em 2011, foi criado por uma equipe de pesquisadores britânicos, que mais tarde se tornou a Raspberry Pi Foundation - uma empresa sem fins lucrativos para produzir o Raspberry Pi - liderados por Eben Upton (UPTON, 2013). Upton relata que, em meados de 2005, percebeu que os novos alunos de ciência da computação ingressavam com menos conhecimento com relação a alunos de anos anteriores. Vendo essa deficiência, Upton propôs-se a criar um microcomputador que motivasse o aprendizado com formato divertido, flexível, de baixo custo e atrativo para crianças e adolescentes (RASPBERRY PI Foundation, 2012).

Aplicações utilizando o Raspberry Pi abrangem diversas áreas do conhecimento e da tecnologia, como por exemplo: a) em Internet das Coisas - que visa conectar dispositivos do dia a dia a rede como eletrodomésticos, carros e até roupas (Zambarda, 2014), bem como acessórios atuando em diversas áreas, como a esportiva e a hospitalar (Futurecom, 2014); b) em Drones, realidade aumentada e impressoras 3D, os quais vêm tornando o mundo real e virtual cada vez mais transparentes; c) Simulação de uma *smart* TV utilizando o sistema operacional Fedora Remix e uma TV comum [Brito, 2012]; d) uma câmera Otto, esta câmera cria Graphics Interchange Format (GIF'S) ao invés de fotos [Upton, 2014]; e) um *software*, professor de braile, desenvolvido para ensinar crianças cegas a ler braile [Mudra, 2016]; e f) o robô VolcanoBot 2, desenvolvido pela doutora Carolyn Parcheta, cujo propósito é escanear em 3D fissuras de um vulcão para estudos de vulcanologia, dentre outros [LIZ Upton, 2014].

Após o surgimento do Raspberry Pi surge o *hardware* Banana Pi, baseado no primeiro, porém com melhorias se comparado ao original. Dentre estas, um conector para telas LVDS, outro para HD com padrão SATA e um mini microfone.

Neste meio tempo outros *hardwares* de baixo custo surgiram com novos propósitos, tais como o **Beaglebone** e o **Galileu**, proposta da Intel com características semelhantes às do Arduino, porém com maior poder de processamento.

Com as novas tecnologias e *hardware* fomentando aplicações em diversas áreas, surgem novas possibilidades de pesquisa. Na área de área de Acessibilidade, por exemplo, o desenvolvimento de membros robóticos e sensores ligados a dispositivos embarcados, ampliam as possibilidades de portadores de necessidades específicas.

2.4. Acessibilidade

Segundo o Decreto nº 5.296/04 a acessibilidade corresponde a fornecer condições para utilização, com segurança e autonomia, total ou assistida, dos espaços, mobiliários e equipamentos urbanos, das edificações, dos serviços de transporte e dos dispositivos, sistemas e meios de comunicação e informação, por pessoa com deficiência ou com mobilidade reduzida.

Sobre a acessibilidade, Pressman (2011, p. 305) diz que, na medida que os computadores se tornam comuns, os engenheiros de *software* devem garantir que o projeto de *interface* de usuário englobe mecanismos que permitam fácil acesso para aqueles com necessidades especiais. O autor também diz que "a acessibilidade [...] é um imperativo por razões éticas, legais e comerciais".

Neste contexto, nos deparamos com a tecnologia Assistiva - utilizada para identificar todo o arsenal de recursos e serviços que contribuem para proporcionar ou ampliar habilidades funcionais de pessoas com deficiência e consequentemente promover vida independente e inclusão(BERSCH & TONOLLI, 2006). Sendo assim, "Para as pessoas sem deficiência a tecnologia torna as coisas mais fáceis. Para as pessoas com deficiência, a tecnologia torna as coisas possíveis" (BERSCH, 2013 *apud* RADABAUGH,1993).

2.5. Pessoas com necessidades especiais

Segundo o Decreto nº 5.296/04 uma pessoa com deficiência, possui limitação ou incapacidade para o desempenho de atividade e se enquadra nas seguintes categorias: deficiência física, deficiência auditiva, deficiência visual, deficiência mental. Em segundo momento o decreto define: "Pessoa com mobilidade reduzida, aquela que, não se enquadrando no conceito de pessoa com deficiência, tenha, por qualquer motivo, dificuldade de movimentar-se, permanente ou temporariamente, gerando redução efetiva da mobilidade, flexibilidade, coordenação motora e percepção".

Sobre pessoas com necessidades especiais, segundo Censo de 2010, cerca de 45,6 milhões - em torno de 24% da população brasileira - alegam possuir alguma deficiência. Em Hortolândia, segundo o IBGE (Instituto Brasileiro de Geografia e Estatística) 12,740 mil pessoas assumem possuir deficiência classificada como "grande dificuldade ou não conseguem de modo algum". Dessa forma, ainda que a quantidade de alunos portadores de necessidades especiais do IFSP - *Campus* Hortolândia em 2015 seja pequena, existe motivação de pesquisas pelo número de portadores de necessidades especiais na cidade e no Brasil.

3. Metodologia

A dinâmica do presente trabalho será conduzida com base no Modelo Evolutivo.

Tal modelo, segundo Pfleeger (2013), permite o desenvolvimento incremental, ou seja, o sistema, conforme especificado na documentação dos requisitos, é dividido em subsistemas que são desenvolvidos em etapas e por funcionalidades. Sommerville (2014), tratando do Modelo Evolutivo, aponta para uma série de incrementos, que são primeiramente definidos, e então cada incremento proporciona um subconjunto das funcionalidades do sistema. Juntamente ao modelo soma-se o paradigma de prototipação.

Segundo Pressman (2011), é comum utilizar esta técnica de prototipação em conjunto com os modelos de processo, tais como o incremental. Pressman afirma também que modelos evolucionários são iterativos, ou seja, apresentam características que permitem desenvolver versões cada vez mais completas do *software*. Cada ciclo de desenvolvimento obtém um conjunto de requisitos e implementa as funcionalidades correspondentes. Segundo Pressman, frequentemente "o cliente define uma série de objetivos gerais para o *software* mas não identifica detalhadamente os requisitos para as funções e recursos". A Figura 2 ilustra o processo de desenvolvimento evolutivo.

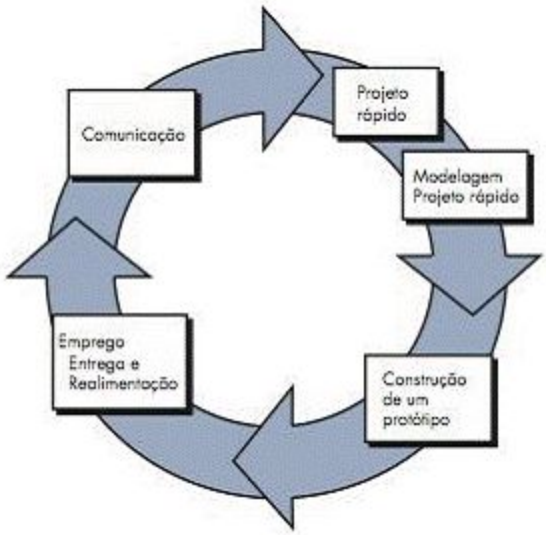

Figura 2. Etapas do Modelo Evolutivo.

(http://www.devmedia.com.br/introducao-aos-processos-de-software-e-o-modelo-incremental-e-evolucionario/29839)

Para o presente trabalho, as etapas e ferramentas utilizadas foram: Coleta e análise de requisitos, elaboração de casos de uso, desenvolvimento de diagramas de classes, implementação do sistema em linguagem orientada a objetos, instalação dos *softwares* pré-existentes escolhidos, Integração com o *software* desenvolvido, testes e ajustes finais e, por fim, a instalação do sistema.

4. Plataforma de *Hardware* e *Software*

O sistema foi projetado com o objetivo de permitir fácil manutenção, proporcionar ao usuário uma *interface* amigável e aberto a customizações por parte de outros desenvolvedores. Além disso, foram escolhidas tecnologias *open-source* e/ou de baixo custo, tanto para a camada de *software* quanto para o *hardware*.

Seu enfoque principal é na interação física por comandos de voz. Basicamente, o sistema possui cadastradas diversas ações, que são disparadas após o reconhecimento de palavras-chave proferidas pelo usuário próximas ao microfone. Na Figura 3 podemos ver o Diagrama de Casos de Uso do sistema.

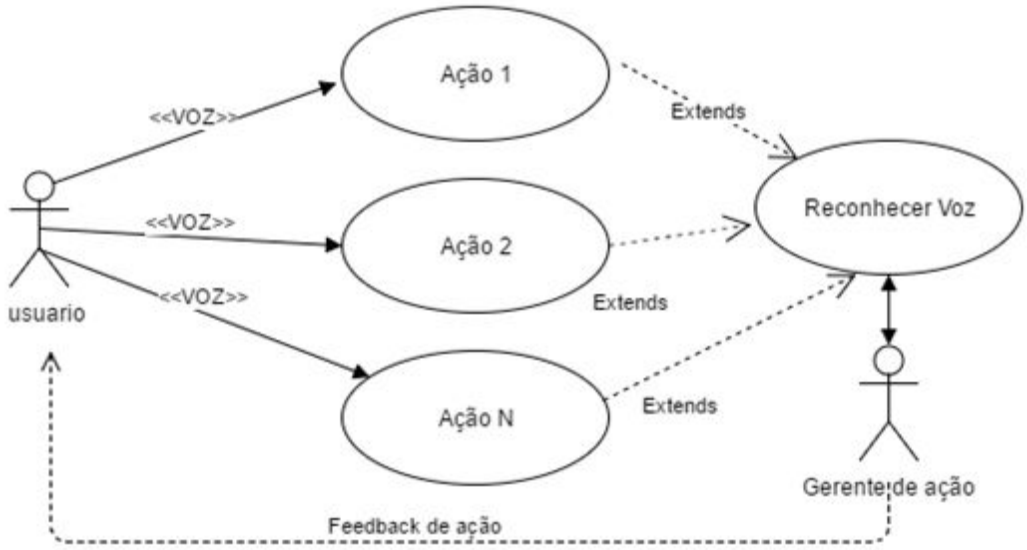

Figura 3. Casos de Uso do Sistema de *Hardware* e *Software* proposto.

Cada um dos casos de uso do sistema foi pensado como uma extensão do caso de uso "Reconhecer Voz", o qual efetivamente identifica a palavra proferida pelo usuário e a compara com os termos previamente cadastrados.

Uma vez identificada a palavra, a ação correspondente é executada pelo Gerente de Ação, módulo do sistema responsável pela execução das ações, sejam estas físicas (atuação no mundo real) ou a nível de *software*. Eventuais divergências, como a não-compreensão da palavra proferida pelo usuário, são comunicadas ao mesmo, a fim de que este possa repetir a instrução.

4.1. A arquitetura do *Software*

O sistema foi dividido em quatro partes principais, chamadas de pacotes de software, como retrata a Figura 4.

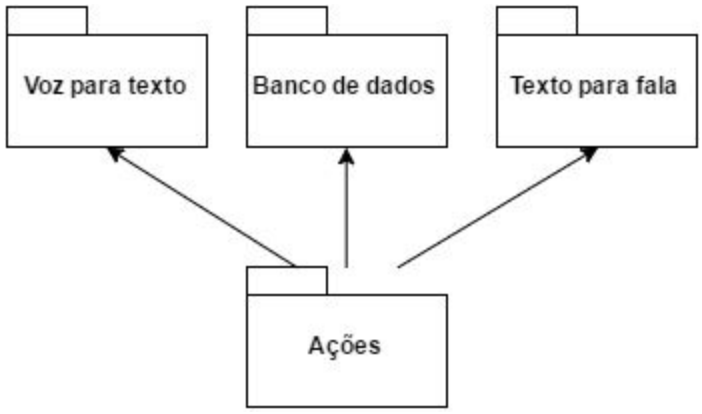

Figura 4. Arquitetura do sistema: Divisão em pacotes software.

Cada pacote definido na Figura 4 implementa uma função específica do sistema, porém todos trabalham em conjunto via o pacote de Ações. O pacote Ações tem a função de gerenciar todas as ações do sistema, logo toda interação gerada via usuário ou *Software* é tratada como uma ação. Para melhor entender o processo de gerenciamento das ações a Figura 5 especifica as classes do pacote Ações.

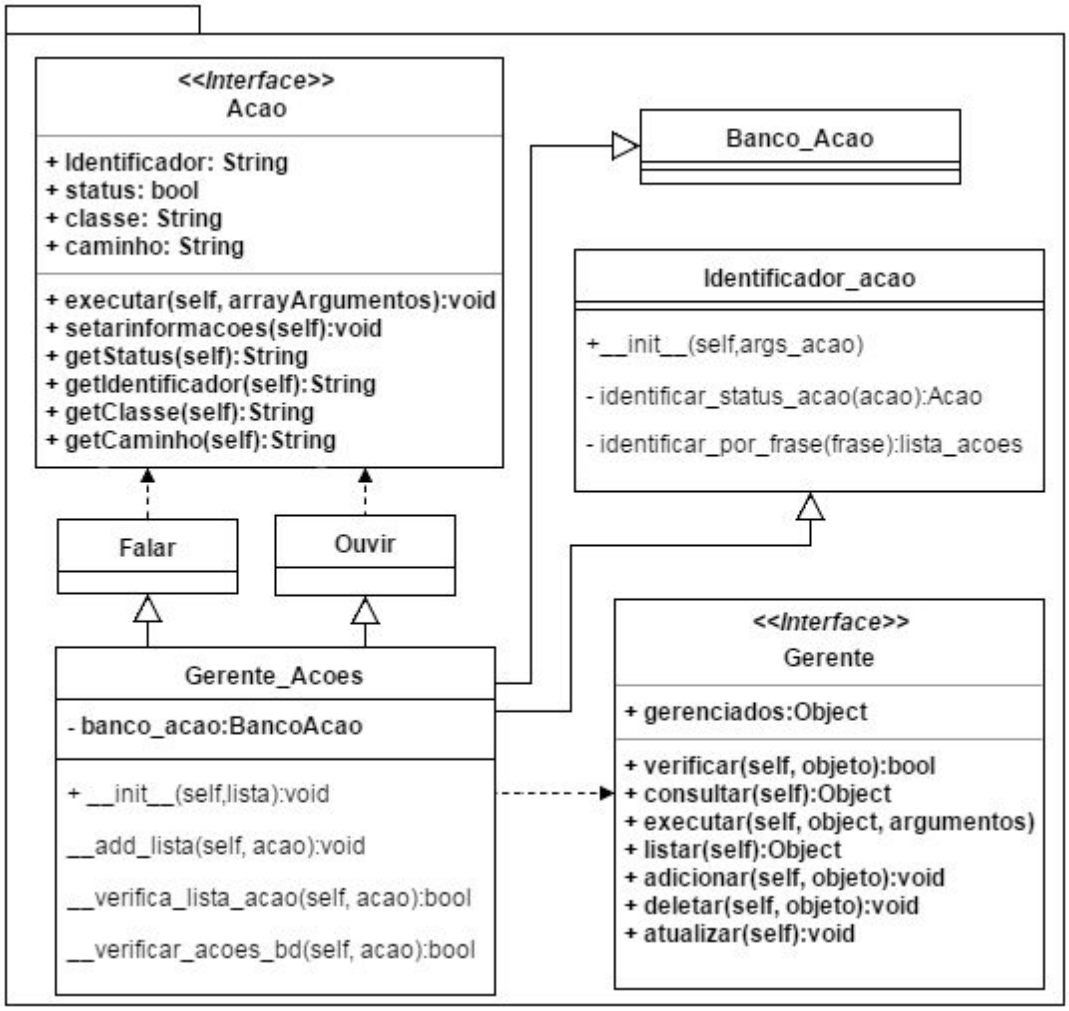

Figura 5. Pacote de Ações.

Podemos pontuar duas classes principais deste pacote. A *interface* "Acao", que viabiliza a implementação de novas ações sem modificações na estrutura do sistema, e a classe "Gerente_Acoes", que tem o papel de coordenar todas as interações do sistema e/ou usuários.

Por outro lado, o ciclo da interação inicia-se pelo pacote de Voz para texto. Este, como sugere o nome, é responsável por converter o áudio capturado via *Hardware* em texto. O diagrama de classes do Pacote de Voz para Texto está representado na Figura 6.

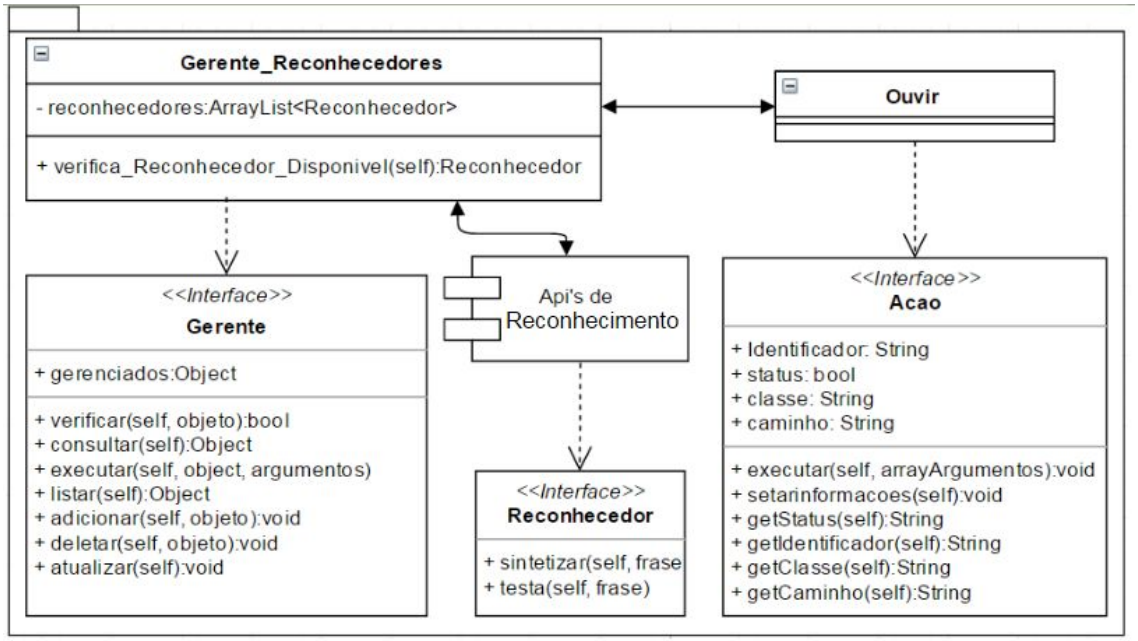

Figura 6. Pacote de Voz para texto.

A estrutura deste pacote visa a disponibilidade da função em tempo de execução do sistema. Ao definir a *interface* Reconhecedor buscamos, independente da API escolhida, garantir sempre o reconhecimento da voz. Porém para isso, como mostra a Figura 6, foi necessário implementar a *interface* "Gerente" via "Gerente_Reconhecedores", para gerenciar as API's implementadas. Deste modo, a ação "Ouvir" sempre receberá uma instância de classe que implemente a *interface* "Reconhecedor", garantindo a disponibilidade da função. Desta forma, podemos implementar diversas API's e até definir critérios de implementação para sua utilização sem qualquer alteração no sistema. Na Figura 6 encontram-se elementos em comum com a Figura 5, como a *interface* Gerente, "Acao" e a classe "Ouvir".

Neste mesmo contexto, no pacote Texto para fala mudam apenas as implementações, ou seja, ao invés de converter voz em texto converte-se texto para voz, como mostra a Figura 7.

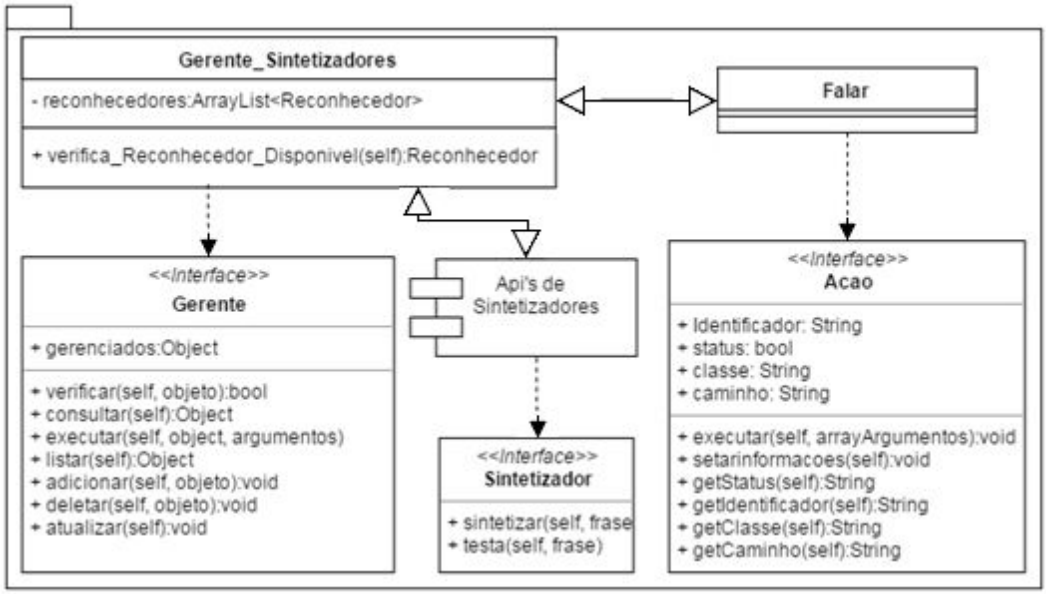

Figura 7. Pacote de Texto para fala.

Por fim, o pacote de banco de dados, que faz a interação efetiva com o banco de dados visando a conversão dos objetos para dados relacionais, está representado na Figura 8.

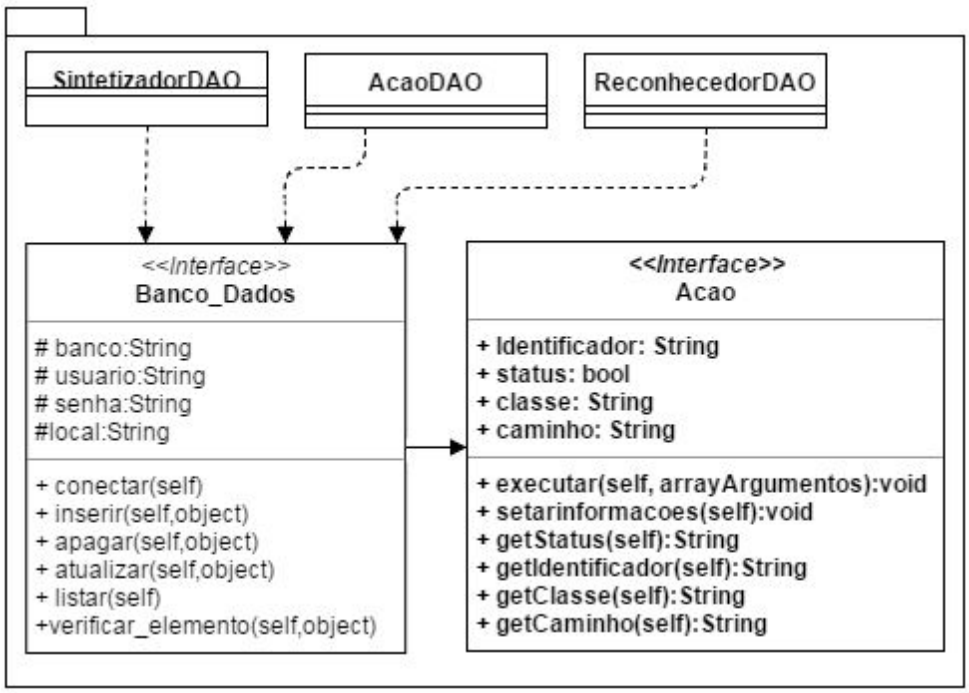

Figura 8. Pacote de Banco de dados.

A interação com o banco de dados também é classificada como ação. Por este motivo, antes de efetivamente ser instanciado implementa-se a *interface* "Acao".

4.2. O ambiente de desenvolvimento

Para realizar o desenvolvimento do trabalho estudamos a possibilidade de uso do Arduino, interfaceando com o ambiente físico, somado a um computador *desktop*, para realizar o processamento do *software*. Deste modo, teríamos mais facilidade no

desenvolvimento do *software*, pois o mesmo seria escrito em alto nível, e o Arduino faria a interação com o ambiente físico, operando em modo escravo. Porém nesse caso teríamos a limitação de mobilizar dois equipamentos para utilização do sistema, bem como o custo do próprio *desktop* e Arduino.

Como solução, propusemos o uso do Banana Pi, o qual possibilita a interação com o mundo físico, permite uma programação de alto nível, possui custo relativamente baixo e contém um microfone embutido. Seu poder de processamento é limitado, se comparado ao *desktop*, porém isso não impacta na realização do projeto proposto.

Uma vez definida a plataforma sobre a qual foi desenvolvido o sistema, foram necessários testes para a escolha da API de reconhecimento de voz. Dentre muitas API's testadas, a *Web Speech* API teve a melhor taxa de reconhecimento, porém funcionou apenas no navegador Chrome. Além disso, para sua utilização, foi necessário o uso de um *Mouse*, o qual precisa ser acionado para iniciar a captura do áudio.

A API CMUSphinix apresentou-se completa e robusta, não necessitando de Internet, porém, implementando-a no *Hardware*, apresentou-se limitada uma vez que a mesma exigiu muito processamento e não suportou a arquitetura ARM do *Hardware*, sendo necessária a recompilação da mesma na arquitetura utilizada.

Por fim, como solução definitiva, a Google *Speech Recognition* apresentou-se estável e eficiente. Vale ressaltar que foram necessárias diversas adaptações de *drivers*, dentre elas a do PyAudio, responsável pela captura do áudio, que em sua implementação no Banana Pi lançava *Exceptions* ao fim de cada interação, finalizando a aplicação. Como solução, realizou-se modificações no processo de captura do áudio, sendo que esta proposta foi submetida via plataforma *GitHub* para os desenvolvedores do *driver*.

A configuração final definida para o sistema ficou, portanto: a placa Banana Pi com o sistema operacional Raspbian - versão Linux customizada para o Raspberry Pi tendo assim maior aproveitamento do *hardware*; uso da linguagem Python - linguagem base do *hardware* - e da API *Speech Recognition* implementando a API Google *Speech Recognition.* Conforme ilustra a Figura 9.

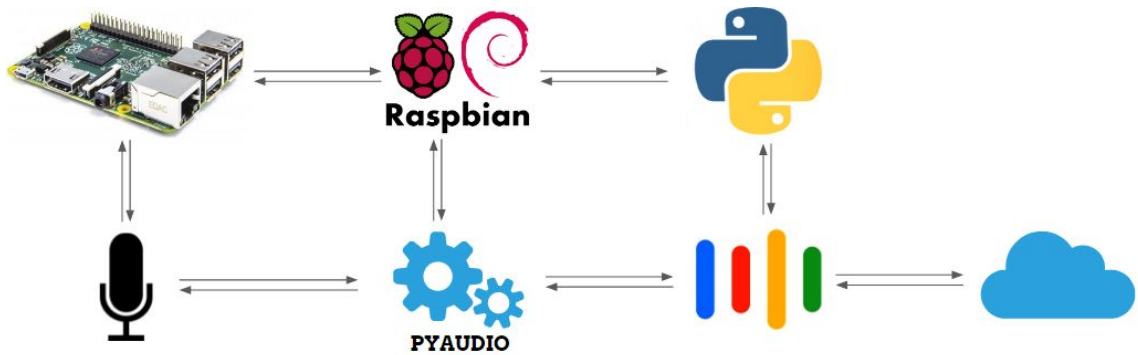

Figura 9. Ilustração do ambiente.

Na Figura 10 somamos a arquitetura do software proposto com o ambiente definido.

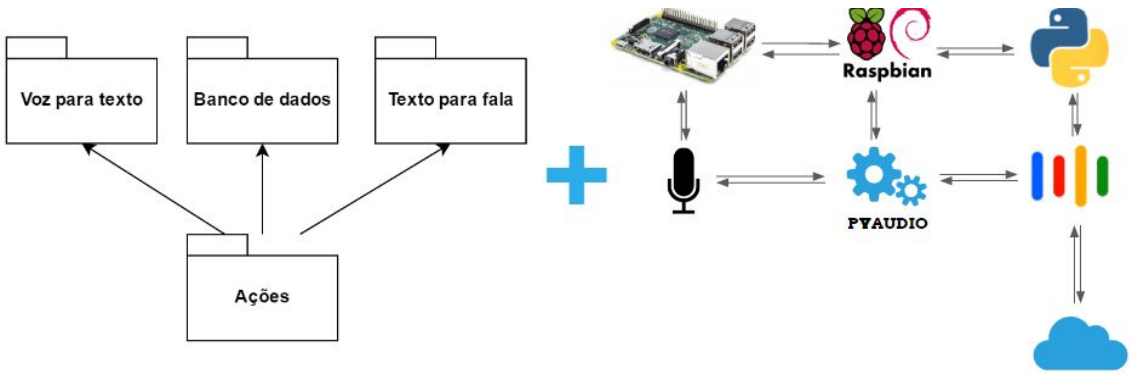

Figura 10. Plataforma desenvolvida.

5. Estudo de caso

O estudo de caso em questão visa auxiliar pessoas com limitações físicas e para comodidade em atividades diárias. As áreas de automação residencial e de Internet das Coisas (IoT) dispõem de uma série de soluções para implementação em uma residência.

A fim de comprovar o conceito da arquitetura desenvolvida, selecionou-se três ações possíveis, todas relativas a ações do tipo "ligar" e "desligar" equipamentos eletrônicos. Tais equipamentos podem ser uma luz, um ventilador, bem como um atuador capaz de abrir e fechar uma porta. A Figura 11 ilustra tais possíveis ações.

Figura 11. Ilustração de Sistema de Automação Residencial.

Para cada *hardware* da Figura 11 definiu-se para representá-lo um *led*, uma referência e as ações a serem executadas. A Tabela 1 ilustra todas as ações cadastradas no sistema.

Tabela 1. Ações cadastradas no banco de dados

Hardware	Referência	Ação	*Led*
Lâmpada	Luz	Ligar	Vermelho
		Desligar	
Porta	Porta	Ligar	Verde
		Desligar	
Ventilador	Ventilador	Ligar	Branco (azul)
		Desligar	

Em nosso estudo de caso, as ações físicas do sistema foram tratadas via interação com *leds*, a fim de obter-se uma realimentação visual das ações físicas propostas. A Figura 12 apresenta uma foto do experimento em execução.

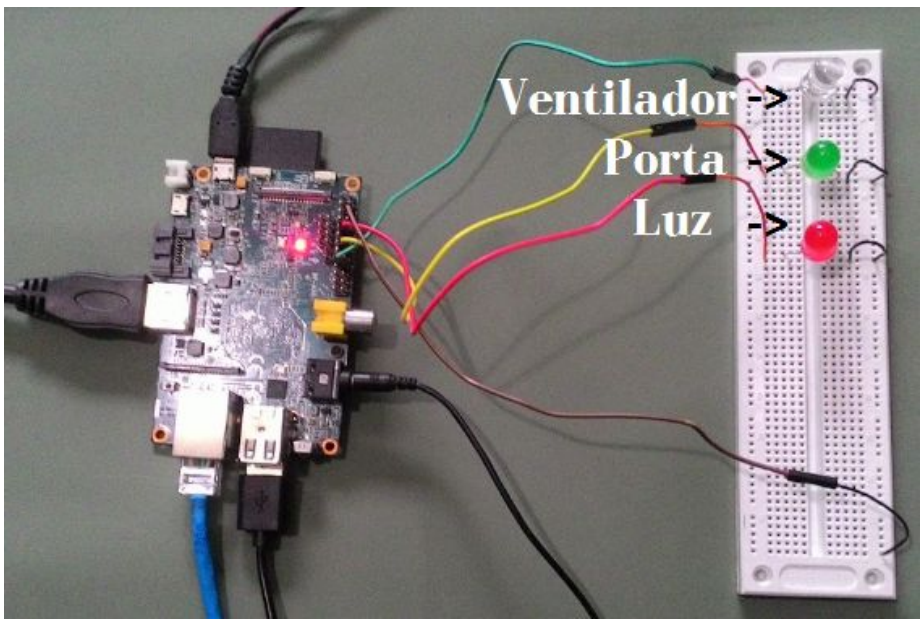

Figura 12. Foto do Sistema em Operação.

Uma vez definida a configuração de ações a serem tratadas, em seguida são elaboradas as classes responsáveis pelas execuções das mesmas. As ações escolhidas foram: Acender e apagar uma lâmpada, ligar e desligar um ventilador e abrir e fechar uma porta, sendo que, na prática, cada uma das ações comanda um led de uma determinada cor.

Caso o sistema fosse instalado em uma residência, os próximos passos seriam projetar e montar os dispositivos necessários para o acionamento de cargas de maior potência.

Dentro da arquitetura implementada é possível acoplar qualquer funcionalidade apenas se preocupando com as ações envolvidas, mesmo que estas utilizem *drivers* e

API's. Deste modo, após finalizada a arquitetura, os casos de uso propostos foram implementados sem dificuldades.

6. Resultados e Discussões

As dificuldades encontradas na realização deste projeto tiveram relação com o crescimento modesto da comunidade do Banana Pi para soluções mais complexas, o que tornou a implementação das API's de reconhecimento de voz o maior desafio. Em sua maioria, no contexto Linux, as API's foram projetadas para processadores 64 e/ou 32 bits acarretando instabilidades em sua utilização.

Com relação à API de reconhecimento de voz definiu-se a API *Speech Recognition* implementando a API Google *Speech Recognition*, a qual se mostrou simples, eficaz e com baixo uso de processamento. Uma limitação identificada desta solução é que a mesma tem a necessidade de utilização de Internet.

Os testes foram realizados de forma empírica, onde foi verificada a quantidade de palavras corretamente interpretadas em função da distância do microfone. Buscou-se efetuar os testes empregando-se sempre a mesma entonação e clareza ao dar o comando de voz.

O microfone utilizado foi o do próprio Banana Pi, e os resultados estão descritos na Tabela 2.

Tabela 2. Testes empíricos de acertos de palavras proferidas.

Distância em m²	Acertos
1	8/10
4	8/10
9	6/10

Verificou-se a partir dos resultados da Tabela 1 que para uma boa utilização do sistema é necessário aplicá-lo em pequenos espaços como cômodos de uma casa. No caso do protótipo com *leds*, percebeu-se que o *feedback* visual auxilia na compreensão do funcionamento do sistema.

Para cada comando acontece um atraso na resposta do sistema de, em média, 4 segundos. Tal atraso deve-se à velocidade de acesso à Internet bem como a limitações do *hardware*.

É válido lembrar que a API escolhida (*Google Speech Recognition*) precisa de conexão com a Internet para reconhecer os comandos do usuário. De toda forma, ela reconhece todos os idiomas disponíveis pelo Google, bastando reconfigurá-la.

7. Conclusões

O trabalho alcançou o objetivo proposto, implementando uma arquitetura de *software Open Source,* de baixo custo, a qual suporta a interação com um usuário via voz e interage com o ambiente físico. Desta forma, com o uso da mesma é possível automatizar e programar tarefas cotidianas ou específicas. Além disso, novas

funcionalidades podem ser acrescidas sem a necessidade de alterações em sua arquitetura.

Como exemplos de possíveis aplicações podem-se citar: na indústria, em processos onde a tecnologia de toque em tela (*touch screen*) não seja aplicável, substituindo-se o comando pela voz; em um ambiente ligado ao conceito de Internet das Coisas, disponibilizando uma *interface* com usuário; em casas inteligentes, para conforto cotidiano, otimizando e automatizando as atividades diárias; bem como auxiliar no dia a dia de pessoas com necessidades específicas - como no caso de movimentos limitados.

Alguns aspectos da solução proposta podem ser aperfeiçoados em futuras versões do *software*, como a utilização de uma API de reconhecimento de voz que não utilize a Internet, o que proporcionaria maior flexibilidade e independência da infraestrutura. Também é necessário investigar a causa real do *delay* a cada comando realizado e testar a solução em outros *hardwares*. Além disso, o emprego de um módulo de potência que permita o uso de relés, assim permitindo o acionamento de cargas de maior potência, como lâmpadas e motores. Para o usuário final, poder-se-ia implementar uma *interface* para gestão do *software*.

O trabalho também possibilitou o emprego de conhecimentos adquiridos ao longo do curso de Análise e Desenvolvimento de Sistemas, tais como metodologia de pesquisa, coleta de dados, análise dos requisitos, gerência de projetos, desenvolvimento de diagramas e arquitetura, programação orientada a objetos e banco de dados.

8. Referências Bibliográficas

ARDUINO. **Arduino Uno**. Disponível em: <http://arduino.cc/en/Main/arduinoBoardUno>. Acesso em: 22 mar. 2016.

AMARAL, M., BARRIVIERA, R., TEIXEIRA, E.C. **Reconhecimento de Voz para Automação Residencial baseado em Agentes Inteligentes**. 2009. Disponível em: https://www.researchgate.net/publication/41908455_Reconhecimento_de_Voz_para_Automacao_Residencial_baseado_em_Agentes_Inteligentes. Acesso em: 30 abr. 2016.

BERSCH, R. **INTRODUÇÃO À TECNOLOGIA ASSISTIVA.** 2013. apud RADABAUGH, 1993. Disponível em: <http://www.assistiva.com.br/Introducao_Tecnologia_Assistiva.pdf>. Acesso em: 13 mar. 2016.

BORGES, A. **PROJETO DOSVOX.** 2002. Disponível em: <http://intervox.nce.ufrj.br/dosvox/>. Acesso em: 19 mar. 2016.

Brito, E. **Site lista cinco coisas que você pode fazer com o Raspberry Pi.** 2012. Disponível em:<http://www.techtudo.com.br/noticias/noticia/2012/12/site-lista-cinco-coisas-que-voce-pode-fazer-com-o-raspberry-pi.html>. Acesso em: 10 mar. 2016.

CAMARATA, K. et al. **A Physical Computing Studio: Exploring Computational Artifacts and Environments**. International Journal of Architectural Computing. Vol. 1. no 2: 169-190. 2003. Disponível em: <https://wiki.cc.gatech.edu/designcomp/images/4/4f/IJAC-p169-physcomp.pdf>. Acesso em 15 mar. 2016.

CHIELE, M.R., ZERBETTO, A. **Desenvolvimento de uma Cadeira de Rodas Controlada por Voz.** 2010. Disponível em: <http://www.eletrica.ufpr.br/anais/cba/2010/Artigos/64854_1.pdf>. Acesso em: 18 mar. 2016.

CMUSphinx Wiki. **CMUSphinx**. Disponível em: <http://cmusphinx.sourceforge.net/wiki/>. Acesso em: 14 abril 2016.

COMPUTAÇÃO Física BR. **Sobre Computação Física**. 2012. Disponível em: <https://computacaofisicabr.wordpress.com/o-que-e-computacao-fisica/>. Acesso em: 10 mar. 2016.

Futurecom. **Entenda os *wearable devices* – os dispositivos vestíveis**. 2014. Disponível em: <http://blog.futurecom.com.br/en/entenda-os-wearable-devices-os-dispositivos-vestiveis/>. Acesso em: 10 abr. 2016.

GUIMARÃES, R. de F.R. et al. **GVOICER:** Sistema de Reconhecimento de Voz para Controle de Aplicações de Realidade Virtual. 2005. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/warv/2005/007.pdf>. Acesso em: 18 mar. 2016.

GUILHOTO, P. J. dos S., ROSA, S.P.S. de S. **Reconhecimento de Voz**. 2001. Disponível em: <https://student.dei.uc.pt/~guilhoto/downloads/voz.pdf>. Acesso em: 18 mar. 2016.

LUCENA, G.G. **Automação residencial por comandos de voz utilizando microcontrolador.** 2006. 82 f. TCC (Graduação) - Curso de Engenharia da Computação, Centro Universitário de Brasília, Brasília, 2006. Disponível em: <http://repositorio.uniceub.br/bitstream/123456789/3264/2/20114845.pdf>. Acesso em: 13 mar. 2016.

MARIN, H. de F., CUNHA, I.C.K.O. Perspectivas atuais da Informática em Enfermagem. **Reben,** São Paulo, v. 3, n. 59, p.354-357, 28 maio 2006. Bimestral. Disponível em: <http://www.scielosp.org/pdf/reben/v59n3/a19v59n3.pdf>. Acesso em: 19 mar. 2016.

MIDIAVOX (Org.). **URAs - Unidades de Resposta Audível em poucas palavras:** Conceitos que podem ajudar a decidir. 2003. Disponível em: <http://www.midiavox.com.br/downloadDoc.php?d=arqDocumentacao&f=Microsoft_Word__ura_white_paper.doc.pdf.> Acesso em: 18 mar. 2016.

MUDRA. **Página Oficial do projeto Mudra**. Disponível em: <http://www.projectmudra.com/> . Acesso em: 11 out. 2016.

ORACLE. **Java *Speech* API**. Java *Speech* API *Frequently Asked Questions*. Disponível em: <http://www.oracle.com/technetwork/java/jsapifaq-135248.html#what> Acesso em: 14 abril 2016.

PFLEEGER, Shari Lawrence. **Engenharia de *software*: teoria e prática**. 2. ed. São Paulo: Pearson Education, 2013. 537 p.

PRESSMAN, R. S. **Engenharia de *software*:** uma abordagem profissional. 7. ed. São Paulo: Bookman, 2011.

RASPBERRY PI. **Página Oficial do Raspberry Pi**. Disponível em: <http://www.raspberrypi.org/> . Acesso em: 22 mar. 2016.

RIBEIRO, C.J.S. **BIG DATA: os novos desafios para o profissional da informação.** Rio de Janeiro: Itec, v. 1, n. 1, 2014. Semestral. Disponível em: <http://www.ies.ufpb.br/ojs2/index.php/itec/article/view/19380/11156>. Acesso em: 12 mar. 2016.

SINGER, T. **Tudo Conectado: Conceitos e Representações da Internet das Coisas.** Práticas internacionais em rede Salvador: Sim Social, 10 out. 2012. Disponível em: <http://www.simsocial2012.ufba.br/modulos/submissao/Upload/44965.pdf>. Acesso em: 10 mar. 2016.

SOMMERVILLE, I. **Engenharia de software.** 9. ed. São Paulo: Pearson / Prentice Hall, 2011.

TOTALIP (Org.). **Total IP explica:** PABX, URA e DAC. 2013. Disponível em: <http://totalip.com.br/total-ip-explica-pabx-ura-e-dac/>. Acesso em: 18 mar. 2016.

UPTON, E.; HALFACREE, G. **Raspberry Pi - Manual do Usuário**. São Paulo: Novatec Editora, 2013.

UPTON, E. **OTTO: A HACKABLE CAMERA POWERED BY RASPBERRY PI.** Disponível em: <https://www.raspberrypi.org/blog/otto-a-hackable-camera-powered-by-raspberry-pi/>. Acesso em: 11 out. 2016.

UPTON, Liz. **Robot *volcanology*.** Disponível em: <https://www.raspberrypi.org/blog/robot-volcanology/>. Acesso em: 11 out. 2016.

ZAMBARDA, P. **“Internet das Coisas”: entenda o conceito e o que muda com a tecnologia.** 2014. Disponível em: <http://www.techtudo.com.br/noticias/noticia/2014/08/internet-das-coisas-entenda-o-conceito-e-o-que-muda-com-tecnologia.html>. Acesso em: 17 mar. 2016.

W3C. ***Web Speech* API *Specification***. Disponível em: <https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html#introduction>. Acesso em: 22 mar. 2016.