FACULDADE DE TECNOLOGIA DE SÃO PAULO

DANIEL FRAZÃO RAMIRES

Automação de Ambiente Utilizando Reconhecimento de Fala

São Paulo-SP

2011

FACULDADE DE TECNOLOGIA DE SÃO PAULO

DANIEL FRAZÃO RAMIRES

Automação de Ambiente Utilizando Reconhecimento de Fala

Monografia submetida como exigência
parcial para a obtenção do Grau de
Tecnólogo em Processamento de Dados
Orientador: Prof. Dr. Silvio do Lago Pereira

São Paulo-SP
2011

Dedicatória

Dedico este trabalho aos meus pais por terem sempre me apoiado e aconselhado em cada etapa da minha vida.

Dedico também ao meu irmão que tantas vezes já buscou exemplos em mim e que hoje também me serve de exemplo para muitas coisas.

Dedico também à memória de minha amiga canina, Pepita, por toda a companhia e amor que me proporcionou em todos os dias de sua existência.

Agradecimentos

Aos meus Pais por sempre buscarem a melhor educação possível para mim e para meu irmão, por todos os incentivos e conselhos que nos deram e nos levaram a buscar cada vez o melhor para nossas vidas.

Aos amigos e colegas da Fatec, alguns por servirem de exemplos a serem seguidos, outros por diversas sugestões que me auxiliaram muito neste trabalho e a todos pelas horas que passamos lado a lado ao longo da faculdade estudando, conversando e muitas vezes rindo.

Ao professor Dr. Silvio do Lago Pereira por me ajudar nas minhas maiores dificuldades. É uma grande honra encerrar este capítulo tendo-o como orientador.

Aos vários professores da Fatec que me prepararam com conhecimentos, técnicas e ideias para que eu fosse capaz de lidar com este grande desafio.

À Karina que arrumou tempo para me ajudar com as normas e formatação.

A todos os demais, da Fatec ou de outros lugares que colaboraram direta ou indiretamente para que este trabalho pudesse ser feito.

Resumo

Este trabalho visa desenvolver um sistema de automação de ambiente utilizando o reconhecimento de fala. Neste trabalho são explicados os conceitos de automação de ambiente e de reconhecimento de fala e é apresentado um sistema em que comandos de voz são utilizados para controlar diversos aparatos em um ambiente virtual.

Palavras-chave: Reconhecimento de fala, Automação de Ambiente, Acessibilidade.

Abstract

This work aims to develop a home automation system using speech recognition. In this work are explained the concepts of home automation and speech recognition and it’s presented a system in which voice commands are used to control many devices in a virtual environment.

Keywords: Speech Recognition, Home Automation, Accessibility.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

ADC – *Analog-to-Digital Converter* – Conversor Analógico-Digital

ANN – *Artificial Neural Network* – Rede Neural Artificial

API – *Application Programming Interface* – Interface de Programação de Aplicação

C# - Linguagem de programação orientada a objetos

HMM – *Hidden Markov Models* - Modelos Ocultos de Markov

MLP – *Multi-Layer Perceptron* – Perceptron Multi-Camada

OS – *Operational System* – Sistema Operacional

PDA – *Personal Digital Assistant* – Assistente Digital Pessoal

PC – *Personal Computer* – Computador Pessoal

PLC – *Power Line Carrier* – Transporte/Comunicação por linha elétrica

SAPI – *Speech* API – API de fala do *Windows* disponível no SDK

SDK – *Speech Development Kit* – Kit de Desenvolvimento de Fala do *Windows*

SMB – *Small & Medium Business* – Pequenos e Médios Negócios

SRGS – *Speech Recognition Grammar Specification* – Especificação de Gramática de Reconhecimento de Fala

STT – *Speech To Text* – Fala Para Texto, Reconhecimento de Fala

TTS – Text To Speech – Texto Para Fala, Fala Artificial

TV – *Television* – Televisão

WAP – *Wireless Application Protocol* – Protocolo de aplicação sem fio

X-10 – Protocolo de comunicação via PLC

X-Box – Videogame da *Microsoft* que dispõe do controle Kinect

Sumário

1 - Introdução

Existem cada vez mais aparelhos eletrônicos que surgem para facilitar tanto nossas tarefas quanto nosso lazer. No entanto, é comum que tais facilidades venham acompanhadas de gama de novos conhecimentos e possibilidades que, apesar de serem geralmente fáceis de compreender e absorver, acabam se acumulando a vários outros conhecimentos sobre configurações e interfaces diversas.

Para a maioria das pessoas isso não representa mais do que um pequeno inconveniente, mas existem situações em que executar um grande número de tarefas simples se torna mais complicado. Este é o caso, por exemplo, de deficientes físicos, que podem ter dificuldade de executar tarefas que exijam a interação com um grande número de aparelhos no ambiente, devido ao excesso de locomoção exigido, ou de deficientes visuais, que podem precisar descobrir como determinado aparelho funciona, sendo que as instruções, apesar de fáceis e intuitivas, estão apresentadas de forma visual.

O objetivo deste trabalho é desenvolver um sistema de automação de ambiente utilizando o reconhecimento de fala como ferramenta para que o sistema de automação seja acessível a deficientes físicos e visuais, permitindo o controle de um ambiente com diversos aparelhos e funções, por meio de comandos de voz.

Este sistema utilizará um ambiente simulado para demonstrar o resultado dos comandos e verificar a plausibilidade do reconhecimento de fala na automação de ambientes.

Nos próximos capítulos serão abordadas as vantagens ao utilizar automação de ambiente, as arquiteturas que podem ser utilizadas para automatizar um ambiente, as dificuldades a serem enfrentadas e como o reconhecimento de fala pode ser usado para sanar algumas destas dificuldades. Serão também explicados os fundamentos da tecnologia de reconhecimento de fala, serão apresentados alguns sistemas existentes e será justificado o sistema escolhido. Após a apresentação do referencial teórico, serão descritos o sistema desenvolvido, os experimentos realizados e os resultados obtidos. Ao final, serão apresentadas as conclusões.

2 - Automação de Ambiente

Podemos definir automação de ambiente como uma tecnologia, ou uma junção de tecnologias recentes que permitem a gestão de todos os recursos de um ambiente (BARROS, 2011).

A automação de ambiente pode ser aplicada desde ambientes empresariais (como escritórios, laboratórios, oficinas, etc.) e comerciais (lojas, restaurantes, hotéis, etc.) até ambientes residenciais.

Quando aplicada em um ambiente residencial a automação de ambiente pode ser chamada de automação residencial, automação doméstica ou domótica. A palavra domótica vem do seu correspondente em Francês *domotique* e surgiu na segunda metade da década de 1980 na França, onde houve as primeiras experiências relacionadas à domótica. O termo deriva da junção da palavra do latim *domus* (casa) e Robótica que vem do checo *robota* (controle automatizado de algo) (BARROS, 2011).

Como qualquer novidade, inicialmente, a domótica é percebida pelo cliente como um símbolo de *status* e modernidade. No momento seguinte, o conforto e a conveniência por ela proporcionados passam a ser decisivos. E por fim, ela se tornará uma necessidade e um fator de economia (BARROS, 2011).

Tanto as redes domésticas quanto aquelas para fins empresariais podem ser definidas como um conjunto de dispositivos “inteligentes” que utilizam um protocolo de comunicação sobre um ou mais meios físicos para levar a cabo os objetivos pretendidos (BARROS, 2011).

Estes dispositivos podem ser classificados em sensores, atuadores, controladores, interfaces e dispositivos específicos (BARROS, 2011).

- **Sensores**: são os dispositivos que coletam dados do campo, sejam variáveis utilizadas no controle (como temperatura, velocidade, pressão, fugas de água, gás, etc.), sejam simplesmente dados para histórico e controle (como medições de tensão e corrente, etc.). Estes dispositivos são classificados como dispositivos de entrada, pois a partir deles uma informação entra no sistema. Os sensores são ideais para serem usados na garagem, cozinha, sala, despensa, *hall*, corredores, escadas, e áreas de serviço, evitando que a

lâmpada permaneça acesa quando não há pessoas presentes, o que acarreta um considerável potencial de economia de energia elétrica de até 60% (BARROS, 2011).

- **Atuadores**: são dispositivos de saída uma vez que a informação sai do sistema para o equipamento físico, para que este realize alguma tarefa. Realizam o controle de elementos como eletro válvulas (água e gás), motores (portas, rega), ligar, desligar e variar a iluminação ou o aquecimento, ventilação e ar condicionado, sirenes de alarme, etc. Podem ser magnéticos, hidráulicos, pneumáticos, elétricos, ou de acionamento misto (BARROS, 2011).

- **Controladores**: gerem a instalação e recebem a informação dos sensores transmitindo-a aos atuadores (BARROS, 2011).

- **Interfaces**: dão e recebem informações do utilizador, constando normalmente de Teclado, *display*, microfone, TV, *PC*, Telefone, Celular, *PDA*, *Internet*, *WAP*, etc (BARROS, 2011).

- Dispositivos Específicos: elementos necessários ao funcionamento do sistema como *modems* ou *routers* que permitem o envio de informação entre os diversos meios de transmissão onde viaja a mensagem (BARROS, 2011).

2.1- Vantagens da automação de ambiente

As razões de se usar um sistema de automação de ambientes incluem mais segurança e conforto para os usuários do ambiente. Pode haver também uma maior economia de água e energia caso sejam utilizadas soluções neste sentido.

Há um ganho de velocidade ao desempenhar determinadas tarefas que antes precisavam ser feitas manualmente.

A automação de ambientes permite centralizar o controle de diversos equipamentos eletrônicos. Esta facilidade pode servir tanto para o conforto de não precisar controlar cada equipamento individualmente como também para melhorar a acessibilidade de deficientes,

seja trazendo o controle de todo o ambiente para um ponto acessível a um deficiente físico ou então permitindo a um deficiente visual controlar um ambiente através de comandos de voz.

Além da centralização dos controles, a automação de ambientes permite também programar séries de ações (como por exemplo, controle simultâneo da dimerização de várias lâmpadas diferentes permitindo configurar rapidamente a iluminação do ambiente).

Essa integração e coordenação entre as diferentes funções do ambiente podem representar uma grande melhoria na segurança ao permitir controlar simultaneamente trancas, portões automáticos e câmeras e podendo também enviar aviso de detecção de movimento (SMARTHOME, 2011).

Existem diversas estruturas possíveis para a instalação de um sistema de Automação de Ambientes (controle remoto, painel na parede, comando de voz, etc.) que podem ser utilizadas da forma que seja considerada mais conveniente. Utilizando emissores infravermelhos, por exemplo, podem-se substituir todos os controles remotos da casa (SMARTHOME, 2011).

Controle Automático adiciona ainda mais conveniência ao fazer coisas acontecerem automaticamente, sem nenhum esforço sendo necessário. Exemplos incluem: ter suas luzes ligadas ao anoitecer e se apagarem na hora desejada, ter seu *home theater* inteiro ligado e sintonizado no canal desejado após pressionar um único botão no seu controle remoto (SMARTHOME, 2011).

Aplicabilidade da Automação em Residências

Cada ser humano possui suas exigências e necessidades particulares, por este motivo, as redes devem assumir preceitos que liguem inteiramente a modo de ser de cada indivíduo. Atualmente as redes domésticas não possuem padrões nem modelo a serem seguidos, restando ao seu utilizador moldá-las de acordo com a sua vontade (AUTOMATIC HOUSE, 2011).

Para estimular a melhoria dos métodos utilizados na automação de redes domésticas, é importante que haja uma utilização mais abundante desta tecnologia. Desta forma poderia haver maior interesse em pesquisas que visem melhorias, tornando a tecnologia mais robusta e confiável (AUTOMATIC HOUSE, 2011).

2.2 – Tecnologias de Automação

É possível utilizar diversas tecnologias para desenvolver uma arquitetura de automação de ambientes (SMARTHOME, 2011).

Uma tecnologia muito popular na automação doméstica é o X-10, pelo seu baixo custo e facilidade de instalação, especialmente em casas já existentes (SMARTHOME, 2011).

O sistema X-10 PLC (*Power Line Carrier*) foi originalmente desenvolvido nos anos 1970, na Escócia. Este sistema permite que produtos compatíveis troquem informações entre si através da linha elétrica existente de 110v, evitando que sejam necessários novos e custosos cabeamentos. Podem ser assinalados até 256 endereços diferentes (AUTOMATIC HOUSE, 2011).

Os primeiros produtos baseados em X-10 começaram a circular em 1979 e abrangeram uma extensa área de aplicações. A patente original expirou em dezembro de 1997 possibilitando que vários fabricantes passassem a desenvolver e fabricar novos e mais confiáveis produtos baseados em X-10 (AUTOMATIC HOUSE, 2011).

Os produtos X-10, por utilizarem o mesmo protocolo básico de transmissão, podem ser livremente usados juntos (AUTOMATIC HOUSE, 2011).

Podemos ver na Tabela 1 algumas das tecnologias que podem ser utilizadas na automação de ambientes.

Tecnologia Remota	Vantagens	Desvantagens	Aplicações Comuns
Infravermelho (Controle Remoto)	Acessível	Linha de visão	TV / outros eletrônicos que usem CR
Rede elétrica (X10)	Acessível e conecta a casa inteira	Precisa de filtro anti-ruído e acoplador de fases	Iluminação, aparelhos e segurança
Radiofreqüência	Funciona através das paredes	Problemas de distância, mais caro do que pela rede de energia	Portão de garagem, rede computacional (*Wi-Fi*)
Cabos	Velocidade e segurança	Custo alto e difícil de atualizar.	Vídeo, rede e aplicações avançadas

Tabela 1 - Tecnologias de automação de ambientes (SMARTHOME, 2011).

2.3 - Dificuldades

A automação permite unificar todos os controles possíveis do ambiente de forma a simplificar o controle de todos os aparelhos do ambiente. Muitas das opções para gerenciar o controle do ambiente utilizam um controle físico (controle remoto, painel de parede, aparelho portátil, etc.) para comandar o sistema, ou seja, controles que exigem contato direto com o usuário.

Para grande parte dos usuários, um controle físico é suficiente para suprir suas necessidades e gerar o devido conforto. No entanto, usuários que apresentem algum nível de deficiência visual ou física poderão ter dificuldades maiores ao utilizar um controle físico, em relação a algum controle sensorial (sensor de presença, reconhecimento de fala, reconhecimento de movimento (por exemplo, o Kinect do X-Box)).

Essa dificuldade se deve, no caso do deficiente visual, à necessidade de decorar a posição dos botões do controle, por exemplo, uma vez que não dispõe da capacidade de utilizar as indicações visuais do controle como referência. Em alguns casos, o controle pode dispor de uma tela plana dotada de *touchscreen* onde a posição dos botões pode mudar facilmente, o que pode ser uma grande comodidade para boa parte dos usuários, mas torna a interface completamente inacessível a deficientes visuais.

No caso de um deficiente físico, um controle físico pode estar em um local de difícil acesso podendo ser um controle pequeno e móvel (controle remoto, *smartphone*, etc) que pode entrar em pequenos espaços inacessíveis ou mesmo controles fixos (painel de parede) que podem exigir o deslocamento até o controle. Em ambos os casos isso pode significar deslocamentos desnecessários ou a necessidade de ajuda de outra pessoa, o que anula em parte a comodidade gerada pela automação de ambientes.

Uma solução para resolver o problema dos deficientes (físicos ou visuais) é utilizar o reconhecimento de fala; pois, uma vez instalados os microfones em todos os pontos necessários, não haverá a necessidade de visualizar o controle para utilizá-lo ou se deslocar até ele.

3 - Reconhecimento de Fala

3.1 - Fundamentos

Reconhecimento de fala é um processo que capta aquilo que é dito e o transforma em dados que podem ser interpretados pelo computador (CARNEGIE MELLON UNIVERSITY, 2011).

Estes dados devem então ser analisados pelo software de reconhecimento para que sejam interpretados e convertidos em texto escrito.

O texto então pode ser exibido em algum dispositivo de saída (tela, impressora, etc.) ou utilizado em algum aplicativo que esteja fazendo uso do software de reconhecimento.

Um sistema de reconhecimento de fala pode ser classificado quanto à dependência em relação ao usuário que irá utilizá-lo, a extensão do vocabulário que pode ser reconhecido por ele e a permissão ou não do uso de fala contínua no reconhecimento.

Ao classificarmos o sistema quanto à sua dependência em relação ao usuário temos:

- **Dependente do usuário**: é desenvolvido para funcionar com um único usuário falante que deve ser aquele que treinará o sistema de STT (*Speech to Text*). Estes sistemas são geralmente mais fáceis de serem desenvolvidos e são mais precisos, mas não são tão flexíveis (CARNEGIE MELLON UNIVERSITY, 2011).
- **Independente do usuário**: é desenvolvido para funcionar com qualquer falante de uma determinada língua. Estes sistemas são os mais difíceis de desenvolver e não são tão precisos, no entanto são mais flexíveis (CARNEGIE MELLON UNIVERSITY, 2011).
- **Adaptativo ao usuário**: é desenvolvido para adaptar sua operação às características dos novos usuários. Possui média dificuldade de desenvolvimento e níveis moderados de precisão e flexibilidade (CARNEGIE MELLON UNIVERSITY, 2011).

Ao classificarmos o sistema de reconhecimento de fala pela extensão de seu vocabulário não há uma definição padrão, mas podemos considerar:

- **Vocabulário pequeno ou pouco abrangente** – dezenas de palavras
- **Vocabulário médio** – centenas de palavras

- **Vocabulário extenso** – milhares de palavras
- **Vocabulário muito extenso** – dezenas de milhares de palavras

O tamanho do vocabulário de um sistema de reconhecimento de fala afeta a complexidade, processamento de requerimentos e precisão do sistema. Algumas aplicações solicitam apenas algumas poucas palavras (exemplo: comandos) enquanto outros precisam de dicionários amplos (exemplo: texto ditado) (CARNEGIE MELLON UNIVERSITY, 2011).

Na classificação do sistema pela continuidade da fala, temos dois tipos:

- **Fala discreta**: processa uma palavra de cada vez e necessita que haja uma pausa entre cada palavra dita. Esta é a forma de reconhecimento de fala mais simples de ser desenvolvida, pois o final das palavras é mais fácil de ser encontrado e uma palavra não afeta a pronúncia da outra. Além disso, porque as pronúncias de palavras são mais consistentes, elas são mais fáceis de serem reconhecidas (CARNEGIE MELLON UNIVERSITY, 2011).
- **Fala contínua**: processa a fala onde as palavras são ditas continuamente, sem separação por pausas. Fala contínua é mais difícil de manusear por causa de uma grande variedade de efeitos possíveis. Em primeiro lugar, é difícil de encontrar o início e o fim de cada palavra. Outro problema é a co-articulação dos fonemas. A produção de cada fonema é afetada pela produção dos fonemas que o cercam e, similarmente, o início e fim de cada palavra são afetados pelas palavras que a antecedem e sucedem. O reconhecimento de fala contínua também é afetado pela velocidade de fala. Uma fala rápida, por exemplo, tende a ser mais difícil de reconhecer (CARNEGIE MELLON UNIVERSITY, 2011).

Um sistema de fala contínua costuma ter um vocabulário de aproximadamente 60.000 palavras ou mais e são projetados para operar em condições mais favoráveis (como ambientes mais silenciosos e microfones de melhor qualidade) (YNOGUTI, 1999).

Devido à maior complexidade exigida dos sistemas de reconhecimento de fala para que fossem capazes de processar a fala contínua, os reconhecedores de palavra isolada chegaram a dominar o mercado até meados de 1999 (YNOGUTI, 1999). .

3.2 - Como funciona

Para converter fala em texto ou em algum comando de computador, um sistema de reconhecimento de fala precisa passar por diversas etapas. Quando falamos, são criadas vibrações no ar que se comportam como uma onda analógica (GRABIANOWSKI, 2011).

O conversor analógico-digital - ADC (*Analog-to-Digital Converter*) da placa de áudio do computador traduz essa onda analógica em dados digitais que o computador pode processar ao digitalizar o som, tirando medidas precisas da onda a intervalos frequentes. O sistema filtra o som digitalizado, para que sejam removidos ruídos indesejados, e o separa em diferentes faixas de frequência (a frequência é o comprimento de onda das ondas sonoras e nós a percebemos como diferenças na altura) (GRABIANOWSKI, 2011).

Além disso, o ADC também padroniza o som, ajustando-o a um nível de volume constante. E para ter uma ideia de como pode ser complexo esse processo todo, o som também pode ter de ser alinhado temporariamente. Como as pessoas nem sempre falam na mesma velocidade, o som deve ser ajustado para corresponder à velocidade dos modelos de som já armazenados na memória do sistema. A seguir, o sinal é dividido em segmentos menores, de até uns poucos centésimos de segundo ou até milésimos, no caso de sons consoantes oclusivos que são paradas de consoantes produzidas pela obstrução do fluxo de ar no trato vocal (fonemas /*p*/, /*b*/, /*t*/, /*d*/, /*k*/, /*g*/). O programa, então, contrapõe esses segmentos aos fonemas conhecidos do idioma desejado (GRABIANOWSKI, 2011).

Um fonema é o menor elemento de um idioma, uma representação dos sons que criamos e juntamos para formar expressões com sentido. Há 34 fonemas na língua portuguesa. Outras línguas, por sua vez, podem ter um número maior ou menor. Destes 34 fonemas, 13 são vogais, 19 consoantes e 2 semivogais. Na Tabela 2, há um conjunto de grafemas adaptado à realidade brasileira (MANOSSO, 2011).

O passo seguinte representa grande parte da dificuldade em se desenvolver um sistema de reconhecimento de fala e é o principal foco da maioria das pesquisas feitas sobre o assunto: o programa examina os fonemas dentro do contexto de outros fonemas ao redor deles. Ele analisa o resultado por um modelo estatístico complexo e os compara com uma grande coleção de palavras, frases e sentenças conhecidas. Por fim, o programa determina o que o

usuário provavelmente estava dizendo e o transforma em texto ou comandos para o computador (GRABIANOWSKI, 2011).

	fonema *	Características fonéticas
Vogais	Á	Aberta, frontal, oral, não arredondada.
	Â	Semi-aberta, central, oral, não arredondada.
	Ã	Semi-aberta, central, nasal, não arredondada.
	É	Semi-aberta, frontal, oral, não arredondada.
	Ê	Semi-fechada, frontal, oral, não arredondada.
	ẽ	Semi-fechada, frontal, nasal, não arredondada.
	Ó	Semi-aberta, posterior, oral, arredondada.
	Ô	Semi-fechada, posterior, oral, arredondada.
	Õ	Semi-fechada, posterior, nasal, arredondada.
	I	Fechada, frontal, oral, não arredondada.
	Ĩ	Fechada, frontal, nasal, não arredondada.
	U	Fechada, posterior, oral, arredondada.
	Ũ	Fechada, posterior, nasal, arredondada.
Consoantes	M	Nasal, sonora, bilabial
	N	Nasal, sonora, alveolar
	Ñ	Nasal, sonora, palatal
	B	Oral, oclusiva, bilabial, sonora
	P	Oral, oclusiva, bilabial, surda
	D	Oral, oclusiva, linguodental, sonora
	T	Oral, oclusiva, linguodental, surda
	G	Oral, oclusiva, velar, sonora
	K	Oral, oclusiva, velar, surda
	V	Oral, fricativa, labiodental, sonora
	F	Oral, fricativa, labiodental, surda
	Z	Oral, fricativa, alveolar, sonora
	S	Oral, fricativa, alveolar, surda
	J	Oral, fricativa, pós-alveolar, sonora
	X	Oral, fricativa, pós-alveolar, surda
	R	Oral, vibrante, sonora, uvular.
	R	Oral, vibrante, sonora, alveolar.
	Λ	Oral, lateral aproximante, sonora, palatal.
	L	Oral, lateral aproximante, sonora, alveolar
Semivogais	Y	Oral, palatal, sonora
	W	Oral, velar, sonora

Tabela 2 – Fonemas do português (MANOSSO, 2011).

3.3 - Estrutura interna do funcionamento

Um sistema de reconhecimento de fala com um vocabulário pequeno ou médio apresenta menor dificuldade em reconhecer a fala das poucas palavras que dispõe em seu vocabulário.

Tendo um vocabulário menor, são reduzidas, também, as chances de que uma palavra ou frase seja interpretada erroneamente. Em sistemas com vocabulários mais amplos, mesmo que estes geralmente disponham de artifícios mais complexos para reconhecer a fala, as divergências no reconhecimento são mais comuns.

Boa parte dos erros de interpretação pode ser devida à similaridade entre a frase ou palavra dita e aquela que foi reconhecida ou frequentemente a ruídos externos.

Para maximizar o acerto do reconhecimento, os sistemas que representam o estado-da-arte em reconhecimento de fala contínua baseiam-se em modelos ocultos de Markov – HMM (*Hidden Markov Models*) (MORAIS, 1997)

Um HMM é definido como um par de processos estocásticos (X,Y) sendo o processo X uma cadeia de Markov de primeira ordem que não é diretamente observável e o processo Y uma sequência de variáveis aleatórias que, no caso do reconhecimento de fala, assumem valores no espaço de parâmetros acústicos (YNOGUTI, 1999). Essa estrutura pode ser vista na Figura 1:

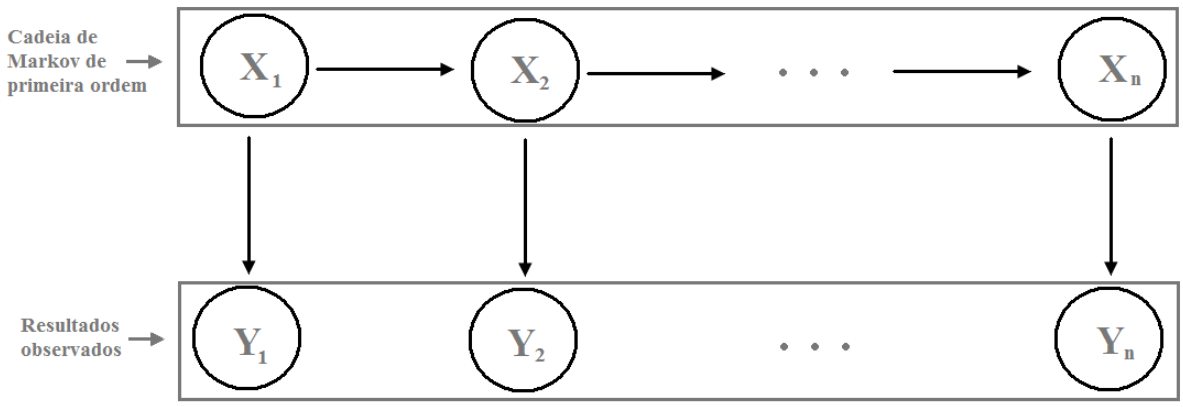

Figura 1 – Modelo Oculto de Markov (YNOGUTI, 1999).

Um HMM gera sequências de observações pulando de um estado para outro, emitindo uma observação a cada salto. Em geral, para o reconhecimento de fala, é utilizado um modelo simplificado de HMM conhecido como modelo *left-right*, ou modelo de Bakis, no qual a sequência de estados associada ao modelo tem a propriedade de, à medida que o tempo avança, o índice do estado aumenta (ou permanece o mesmo), isto é, o sistema caminha da esquerda para a direita no modelo (YNOGUTI, 1999)

Dada uma locução de entrada, um sistema de reconhecimento de fala gera hipóteses de palavras ou sequências de palavras. Destas hipóteses pode resultar uma única sequência de palavras, uma coleção de *n* melhores sequências de palavras, ou uma treliça de hipóteses de palavras parcialmente superpostas. Isto é feito num processo de busca no qual se compara uma sequência de vetores de características acústicas com os modelos das palavras que estão no vocabulário do sistema (YNOGUTI, 1999).

Em geral, o sinal de fala e suas transformações não exibem indicações claras sobre as fronteiras das palavras, de modo que a detecção destas fronteiras faz parte do processo de geração de hipóteses realizado no procedimento de busca (YNOGUTI, 1999).

Porém, para viabilizar a modelagem matemática de um HMM, são realizadas inúmeras suposições simplificadoras que limitam seu potencial efetivo (MORAIS, 1997).

Outro modelo muito utilizado no reconhecimento de fala são as redes neurais artificiais – ANN (*Artificial Neural Networks*) (MORAIS, 1997)

As ANNs são modelos matemático-computacionais inspirados no funcionamento das células neuronais e que possuem processamento altamente paralelo, executado por unidades denominadas neurônios (MENDES; OLIVEIRA, 2011).

O conhecimento é adquirido pela rede, a partir de seu ambiente, através de um processo de aprendizagem e é armazenado como peso sináptico, um valor que indica a força de conexão entre os neurônios (MENDES; OLIVEIRA, 2011).

Este aprendizado ocorre através de processos algorítmicos que alteram os pesos sinápticos de forma a representar o conhecimento adquirido (MENDES; OLIVEIRA, 2011).

Uma forma de ANN comumente utilizada em reconhecimento de fala é o *perceptron* multicamadas – MLP (*Multilayer Perceptron*) (MORAIS, 1997).

O MLP é uma arquitetura que apresenta uma camada com unidades de entrada, conectada a uma ou mais unidades intermediárias, chamadas *camadas ocultas*, e uma camada de unidades de saída conforme Figura 2.

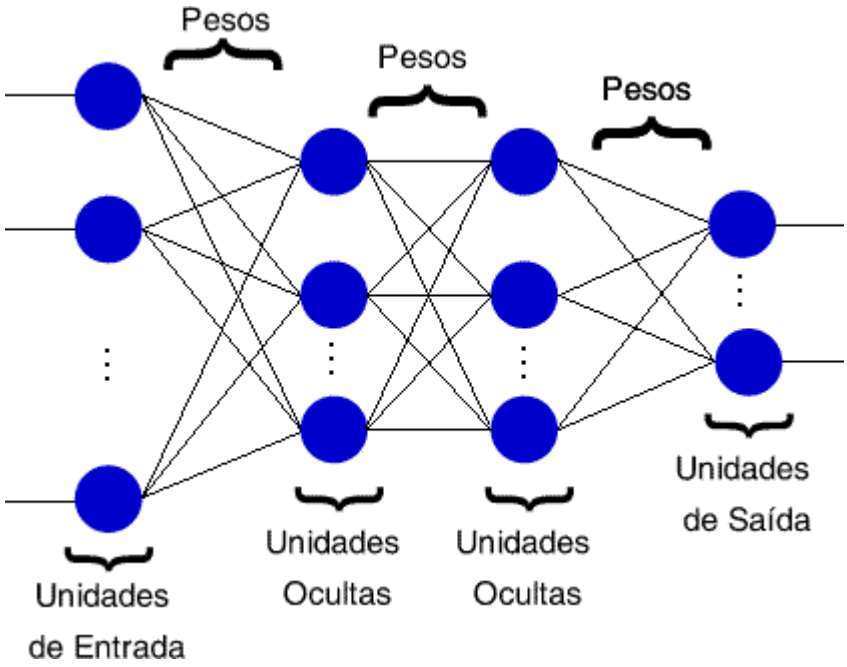

Figura 2 – Arquitetura do perceptron multi-camadas (MENDES; OLIVEIRA, 2011)

As ANNs não necessitam fazer uso de tantas suposições simplificadoras quanto as que os HMM precisam, podem aprender a generalizar superfícies complexas de decisão, tolerar ruídos e suportar paralelismo (MORAIS, 1997).

Todas estas vantagens tornam as ANNs extremamente poderosas para modelar as variabilidades acústicas da fala. Entretanto, ao contrário dos HMMs, as ANNs não têm se mostrado eficientes para o modelamento das variabilidades temporais. Com o objetivo de unir em uma única estrutura o que há de melhor nas tecnologias de redes neurais artificiais e de modelos ocultos de Markov, têm sido estudados e avaliados modelos híbridos ANN-HMM nos quais a modelagem das variabilidades acústicas é confiado à ANN enquanto o HMM responsabiliza-se pela absorção das variabilidades temporais (MORAIS, 1997).

3.4 Sistemas existentes

Windows SDK:

O *Speech Recognition* (reconhecimento de fala) no *Windows Vista* dá o poder aos usuários de interagir com seus computadores por voz. Ele foi projetado para pessoas que querem limitar

significativamente o do mouse e teclado, enquanto mantém ou aumentam sua produtividade geral (MICROSOFT, 2011).

É possível utilizar este recurso em um projeto próprio, através de uma API chamada SAPI (*Speech API*) que faz parte do SDK (*Speech Development Kit*).

Dragon Dictate* e *Dragon NaturallySpeaking:

O *Dragon Dictate* 2.5 é um sistema de reconhecimento de fala destinado a computadores que estejam rodando o sistema operacional *Mac OS*. De acordo com a Nuance (2011), este sistema possui uma maior flexibilidade na criação de documentos no *Microsoft Word* e novos meios de formatação.

A Nuance (2011) informa também que o *Dragon NaturallySpeaking* 11.5 é um modo rápido e conveniente de interagir com o PC e atinge por volta de 99% de precisão no reconhecimento de fala.

LumenVox *Speech Engine*:

O *Speech Engine* da *LumenVox* é um preciso reconhecedor de voz baseado em padrões que suporta vários idiomas e pode realizar o reconhecimento de voz em dados de áudio de qualquer fonte de áudio. No *Linux* ou *Windows*, este motor de fala independente de alto-falantes ou *hardware* alimenta soluções de fala e plataformas implantadas em empresas e ambientes de SMB (*Small and Medium Business*) no mundo inteiro (LUMENVOX, 2011).

Ele também fornece aos desenvolvedores de aplicações de fala uma plataforma eficiente de desenvolvimento e execução, permitindo linguagem dinâmica, gramática, formatação de áudio, e capacidade de criar registro para personalizar cada passo de suas aplicações. Gramáticas são inclusas com uma simples lista de palavras ou pronúncias, ou no padrão industrial *Speech Recognition Grammar Specification* (SRGS) - Especificação de Gramática de Reconhecimento de Fala (LUMENVOX, 2011).

Sistema escolhido:

O sistema de reconhecimento de fala escolhido foi o sistema SDK do *Windows* por não exigir a compra de nenhum *software* e por ser compatível com C# (a linguagem de programação escolhida para o desenvolvimento da aplicação) e ter um conteúdo mais abundante em fóruns.

Ao instalar o SDK 5.1 em um computador rodando *Windows Vista* os recursos para reconhecimento de fala ficaram imediatamente disponíveis para serem adicionados ao projeto.

4 - Sistema e Experimentos

4.1 – Descrição do Sistema

4.1.1 - Ambiente Virtual

Ao iniciar o programa, abre-se uma tela contendo botões, uma caixa de texto, uma caixa de imagem contendo o ambiente simulado e um mapa para identificar os cômodos.

O ambiente simulado, exibido na Figura 3, constitui-se de uma planta estilizada de uma residência com cinco cômodos e um corredor que serve para demonstrar visualmente os efeitos dos comandos de voz.

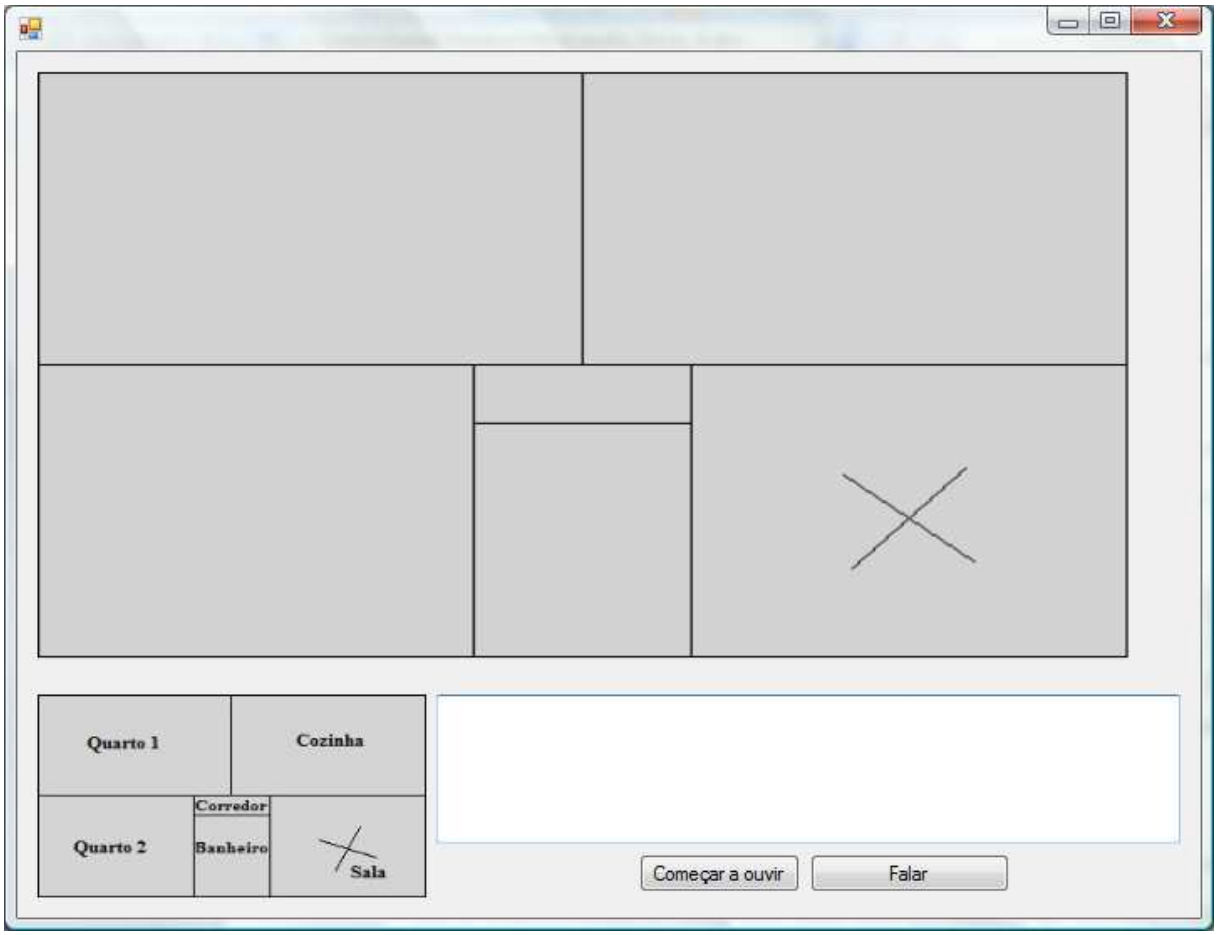

Figura 3 – Ambiente simulado (tela do aplicativo desenvolvido)

Os cômodos podem ser iluminados individualmente e existe também um ventilador funcional representado por uma cruz que fica no centro do cômodo descrito como “sala”.

Ao redimensionar a tela o ambiente também é redimensionado até que se chegue ao tamanho mínimo configurado para o elemento ou à maximização completa da tela que contém o ambiente.

4.1.2 – Controle e comandos

Ao ligar o reconhecimento de fala, podem-se pronunciar comandos para que o sistema obedeça ativando ou desativando os recursos virtuais do ambiente simulado tais como luzes e ventilador.

Estes comandos, após serem reconhecidos pelo *recognition engine*, podem ser captados no formato de texto pela API.

Ao solicitar verbalmente o acendimento das luzes de um dos cômodos virtuais, este terá sua cor de fundo trocada de cinza-claro para amarelo-claro. Veja exemplo Figura 4:

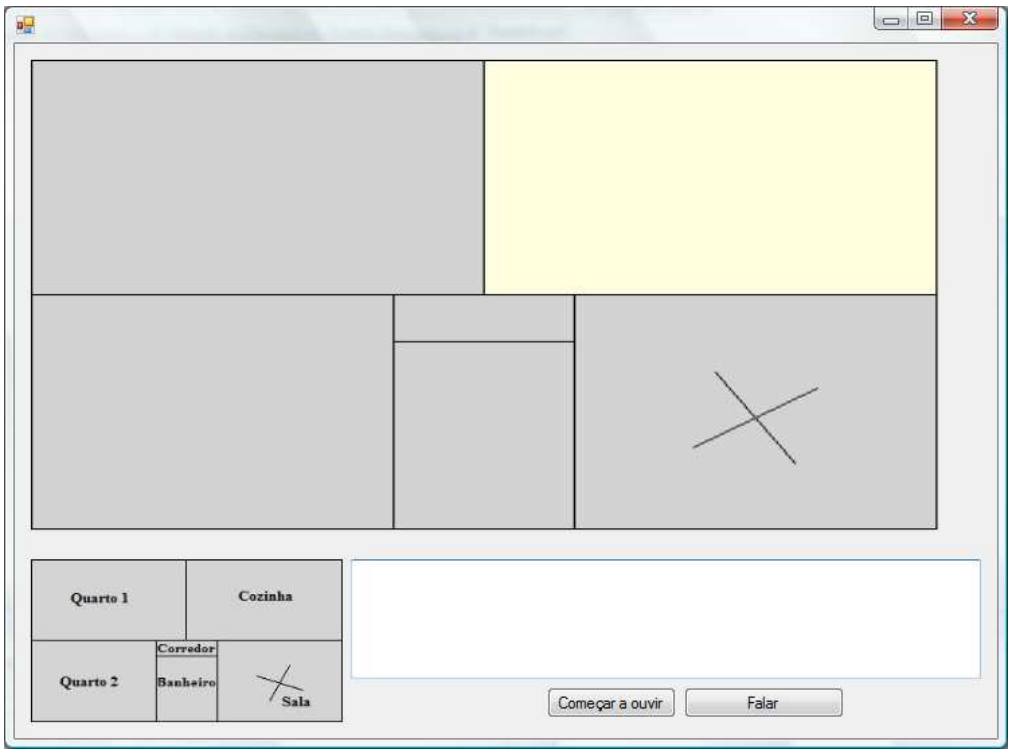

Figura 4 – Exemplo de lâmpada acesa: Cozinha

Ao solicitar verbalmente o funcionamento do ventilador, o elemento visual que representa o ventilador inicia movimento em seu eixo no sentido anti-horário até que haja solicitação verbal para que pare.

O *recognition engine* do *Windows* está disponível em algumas linguagens, no entanto o português não consta entre elas. Uma vez que foi escolhida a *recognition engine* em inglês, as palavras são reconhecidas de acordo com a gramática anglófona. Esta é a razão dos comandos utilizados no sistema estarem em inglês.

A alteração dos comandos no sistema é simples e, caso seja instalada uma *recognition engine* do *Windows* em outra língua, não haverá dificuldade em adaptar o sistema.

Os comandos configurados no sistema são:
Bedroom one – alterna as luzes do quarto 1 entre ligadas e desligadas
Bedroom two – alterna as luzes do quarto 2 entre ligadas e desligadas
Kitchen – alterna as luzes da cozinha entre ligadas e desligadas
Room – alterna as luzes da sala entre ligadas e desligadas
Fan – alterna entre as diferentes velocidades do ventilador: rápido, lento e desligado
Shower – alterna as luzes do banheiro entre ligadas e desligadas
Way – alterna as luzes do corredor entre ligadas e desligadas
Reset – apaga todas as luzes
Acender Luzes – acende todas as luzes
Message [“mensagem”] – exibe uma *message box* contendo como texto a [“mensagem”]

O comando “Acender Luzes” está em português para demonstrar uma técnica que permite que a *recognition engine* em inglês possa ser usada para comandos em português.

Esta técnica se baseia em dizer o comando desejado em português e analisar a resposta do reconhecimento de fala. Caso uma resposta seja razoavelmente constante, ou seja, em ao menos 80% das vezes, o mesmo comando implica numa mesma resposta, então o comando poderá ser utilizado.

O comando “Acender Luzes”, por exemplo, é reconhecido pela *recognition engine* como “*A Senior* Luzes”. Apesar de “luzes” ser uma palavra em português, esta foi aprendida e entrou

na gramática do sistema, mas só foi reconhecida graças ao fato das pronúncias, tanto em inglês quanto em português, desta palavra serem muito similares.

4.1.3 – Estrutura geral

O sistema foi feito utilizando C# e o SDK 5.1 que possui APIs que permitem tanto produzir fala artificial ou TTS (*Text to Speech*) e reconhecer fala ou STT (*Speech to Text*).

A tela do sistema dispõe de duas imagens, uma abriga o ambiente virtual e outra um pequeno mapa com os nomes dos cômodos representados, há também uma caixa de texto e dois botões: “Começar a ouvir” e “Falar”

O acionamento do botão “Começar a ouvir” dá início ao processo de reconhecimento, ligando a *recognition engine* caso esteja desligada e iniciando o processo de escuta dos comandos que forem ditos.

O botão “Falar”, quando acionado, utiliza o texto disponível na caixa de texto para sintetizar fala.

4.1.4 – Ciclo percepção x ação

Conforme pode ser acompanhado na Figura 5, uma vez iniciado no programa o processo de escuta dos comandos, cada evento de reconhecimento (1) irá direcionar o resultado do reconhecimento (2) para a API.

O programa confere se o resultado recebido pela API é um dos comandos programados através de comparação simples de *strings* (3).

Uma vez identificado um comando são executadas as ações referentes àquele comando (4).

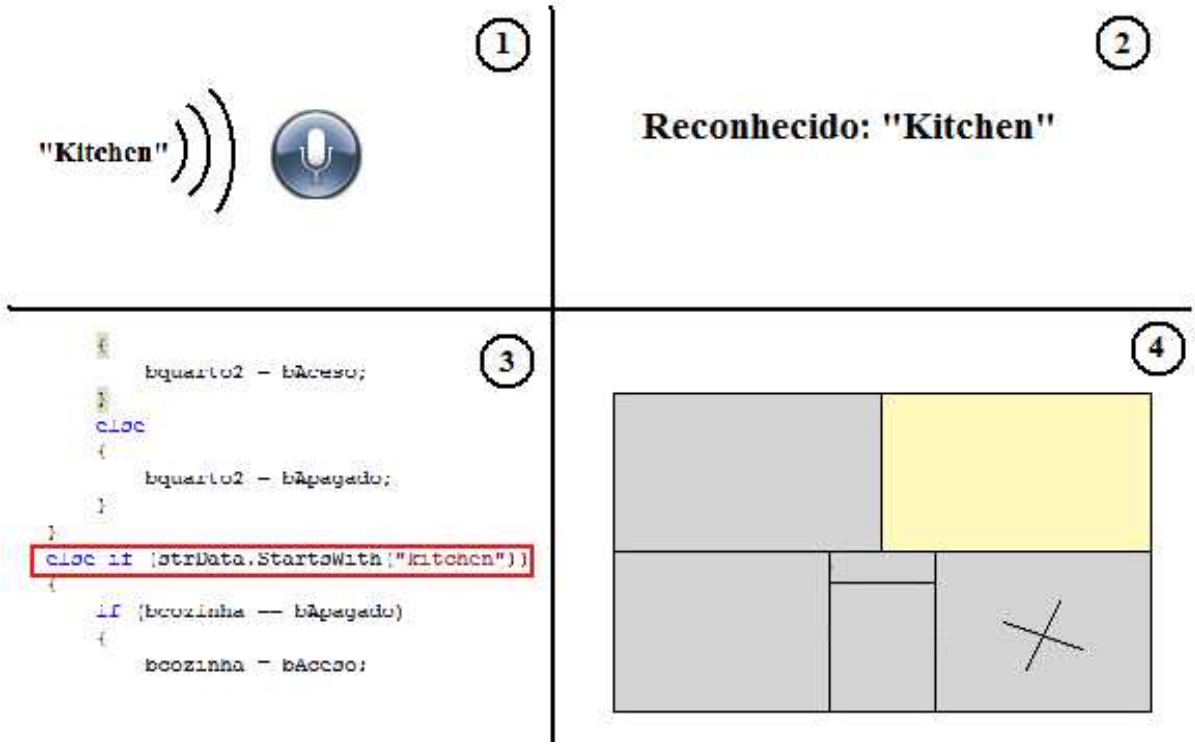

Figura 5 – Ciclo Percepção x Ação

4.1.5 – Exemplos de código

Para utilizar os recursos da API de fala do SDK, basta adicioná-la ao projeto e declarar as linhas de "*using*" correspondentes à síntese (1) e ao reconhecimento de fala (2).

```
using System.Speech.Synthesis; // (1)
using System.Speech.Recognition; // (2)
```

A utilização do recurso de TTS é extremamente simples de ser utilizada. No exemplo abaixo estão codificados a declaração do sintetizador de voz (3) e a utilização deste sintetizador para pronunciar uma frase de exemplo (4).

```
SpeechSynthesizer synth = new SpeechSynthesizer(); // (3)
synth.Speak("It's not another hello world."); // (4)
```

Para utilizar o recurso de STT são necessários mais passos. No exemplo abaixo estão codificados e explicados os passos necessários para criar um sistema simples que entenda comandos:

É necessário declarar no início uma *recognition engine* (5).

```
SpeechRecognitionEngine recognitionEngine =
    new SpeechRecognitionEngine(
```

```
        new System.Globalization.CultureInfo("en-US")); // (5)
```

No botão que aciona o reconhecimento de fala, é preciso selecionar o componente de entrada de áudio (6), carregar uma gramática de reconhecimento (7), adicionar um *handler* para tratar o evento de reconhecimento de fala (8) e iniciar o processo de reconhecimento assíncrono de fala (9).

```
recognitionEngine.SetInputToDefaultAudioDevice(); // (6)

recognitionEngine.LoadGrammar(new DictationGrammar()); // (7)

recognitionEngine.SpeechRecognized += new EventHandler
<SpeechRecognizedEventArgs>(recognitionEngine_SpeechRecognized); // (8)

recognitionEngine.RecognizeAsync(RecognizeMode.Multiple); // (9)
```

A seguir é necessário criar a rotina que será chamada pelo *handler* de reconhecimento (10).

```
public void recognitionEngine_SpeechRecognized(object sender,
        SpeechRecognizedEventArgs e) {
}// (10)

MessageBox.Show("Texto reconhecido: " + e.Result.Text); // (11)
```

Para executar comandos é preciso utilizar, dentro da rotina do *handler*, o texto resultante do reconhecimento através do “e.Result.Text” (11). Um exemplo de comando simples seria exibir uma mensagem sempre que é dito o comando “*Hello*” (12)

```
if (e.Result.Text.StartsWith("message "))
{
        MessageBox.Show(“Hello my friend!”);   // (12)
}
```

Um exemplo mais dinâmico de comando poderia ser um comando que escreve numa mensagem aquilo que for dito após o comando *“message”* (13).

```
if (e.Result.Text.StartsWith("message "))
{
        MessageBox.Show(e.Result.Text.Substring(7));  // (13)
}
```

4.2 – Descrição dos experimentos

Experimento 1:

Numa fase prévia, antes da programação dos comandos, foram testados três tipos de microfone diferentes: um microfone embutido de *notebook*, um *headfone* e um microfone de mesa para testar qual apresentaria maior desempenho.

Os microfones foram testados através de 40 palavras ditadas para cada um e foi medida a quantidade de erros pela quantidade de palavras.

Neste teste se destacou o *headfone* com 82% de acerto, seguido pelo microfone embutido com 77% de acerto e por último o microfone de mesa com 65%.

Experimento 2:

O objetivo deste experimento foi testar o reconhecimento de cada comando utilizado no programa. O experimento foi feito com 4 usuários sendo 1 o usuário padrão que treinou a *recognition engine* e outros 3 usuários convidados identificados pelo número.

Para cada usuário, cada um dos dez comandos foi repetido 10 vezes ao longo do experimento e foi anotado o número de erros. A Tabela 3 apresenta os resultados obtidos.

Comandos	Erros usuário padrão	Erros usuário 1	Erros usuário 2	Erros usuário 3
Bedroom one	0	1	0	0
Bedroom two	0	1	0	0
Kitchen	0	3	0	0
Room	1	4	3	2
Fan	*1*	*3*	*5*	*5*
Shower	*0*	*2*	*0*	*2*
Way	*1*	*3*	*2*	*0*
Reset	*0*	*1*	*0*	*0*
Acender Luzes	*2*	*5*	*1*	*2*
Message	0	1	0	1

Tabela 3 – Resultado do teste dos comandos

Experimento 3:

Em diversos momentos, na maioria dos casos numa pausa entre testes informais, o reconhecedor de fala do *Windows* foi deixado em estado de espera, mas, devido a conversas próximas ao microfone, era ativado acidentalmente ao interpretar erroneamente o comando

para que fosse ativado: “*Start listening.*” – que significa “Comece a ouvir”. A ocorrência pode ser percebida através do som padrão que é emitido ao ativar o reconhecimento, no entanto parte dos casos foi percebida através de textos aleatórios sendo digitados pelo reconhecimento de fala.

Foi decidido elaborar um experimento para verificar os riscos potenciais ao abandonar o sistema de reconhecimento de fala exposto a um ambiente onde ocorrem conversações.

O experimento consistiu em deixar o reconhecedor de fala ligado, porém inativo e iniciar um monólogo aleatório até que alguma ação indesejada ocorresse. Este experimento foi feito 15 vezes.

A ativação ocorreu com uma média de 10 segundos de conversação. Em onze casos a ação indesejada foi a digitação dos reconhecimentos aleatórios, em dois casos o reconhecedor de fala maximizou uma das janelas abertas, em outro abriu o *Microsoft Word* e em outro abriu o menu iniciar.

Experimento 4:

Estando o usuário a uma distância de dois metros de um microfone fixo, situação considerada comum no uso diário de automação de ambientes, não houve perda significativa de qualidade no reconhecimento. Para quantificar esta informação foi desenvolvido um experimento no microfone embutido do *notebook* (representando um microfone fixo) nas distâncias de dois palmos, a distância comum de uso deste microfone, e dois metros.

À distância de dois palmos, o sistema atingiu 75% de assertividade no reconhecimento de fala contínua para texto ditado e 87% para os comandos configurados no programa. À distância de dois metros, os valores baixaram para 68% de assertividade no reconhecimento de fala contínua para texto ditado e 84% para os comandos configurados no programa.

5 – Conclusões

O sistema desenvolvido, conforme a proposta deste trabalho, apresenta um ambiente simulado e a possibilidade de controlar este ambiente através de comandos de fala.

Para possibilitar este tipo de controle é utilizada uma ferramenta de reconhecimento de fala que converte a fala em texto de forma que os comandos possam ser facilmente reconhecidos.

As funções controláveis apresentadas no ambiente simulado são as luzes dos cômodos e a velocidade do ventilador.

Para controlar este ambiente foram testados três tipos de microfones de forma a descobrir qual seria mais adequado: o microfone do *notebook*, um *headfone* e um microfone de mesa.

Os resultados indicaram o *headfone* como a melhor opção, uma vez que apresentou 82% de acerto contra 77% do microfone embutido e 65% do microfone de mesa.

Devido à grande diferença observada, conclui-se que a qualidade dos microfones utilizados é essencial para o bom funcionamento do reconhecimento de fala.

O reconhecimento dos comandos utilizados no programa foi testado para verificar a assertividade de cada um.

A assertividade geral do experimento foi de 87%, havendo 52 falhas de reconhecimento em meio a 400 testes de comando.

A assertividade com o usuário padrão (o usuário principal do sistema) foi de 95%, havendo apenas cinco falhas em 100 comandos.

O comando que se destaca pela menor assertividade é o “Acender Luzes” com apenas 75%, havendo 10 falhas em 40 comandos. Este é o único comando em português e, uma vez que a *recognition engine* utilizada reconhece apenas a pronúncia fonética do inglês, este comando é reconhecido internamente pelo programa como “A senior luzes”, o que reduz a assertividade do comando em relação a outros.

Apesar do pequeno sucesso em fazer a *recognition engine* em inglês do *Windows* reconhecer comandos em português, não haveria dificuldade em, tendo uma *recognition engine* em português(pt-br), alterar levemente o sistema para que funcione completamente em nossa língua.

Outro experimento verificou os riscos potenciais ao abandonar o sistema de reconhecimento exposto a um ambiente onde ocorrem conversações próximas ao microfone.

Foi detectado neste experimento um risco baixo relacionado ao reconhecimento de comandos aleatórios pelo sistema, mas que, apesar de baixo, este risco se revelou comum.

Tendo em vista este risco, sugere-se aproveitar uma característica da *recognition engine* do SDK: reconhecer a fala mesmo com o reconhecedor de fala do *Windows* inativa.

Ao detectar o comando *"Start listening"*, o comando que ativa o reconhecimento de comandos do *Windows*, a API pode ser usada para bloquear a ativação do reconhecedor do *Windows*, deixando disponíveis para reconhecimento os comandos programados no programa.

A distância em relação ao microfone, um fator que contaria no caso de um ambiente real, foi testada para verificar a assertividade do reconhecimento em uma distância considerada comum no uso diário de automação de ambientes.

Comparando com a distância padrão de utilização do microfone, a perda de assertividade foi de 7% para texto ditado, baixando de 75% para 68%, já para os comandos do programa houve uma perda menor: apenas 3%, baixando de 87% para 84% de assertividade.

Levando em conta estas conclusões, um sistema de automação de ambientes que utilize reconhecimento de fala pode ser instalado de forma satisfatória, desde que sejam levados em conta: a qualidade dos microfones utilizados, o nível de treinamento do sistema em relação ao usuário, a seleção de comandos que apresentem maior assertividade e um bom planejamento do posicionamento dos microfones para que nenhuma área fique descoberta.

Referências Bibliográficas

AUTOMATIC HOUSE. **O que Podemos Automatizar.** Disponível em: <http://www.automatichouse.com.br/AutomaticHouse/WebSite/Automacao/Conforto.aspx>. Acesso em: 10 nov. 2011.

BARROS, Auriza de. **Edifícios Inteligentes e a Domótica:** Proposta de um Projecto de Automação Residencial utilizando o protocolo X-10. 2010. 105 f. Monografia (Graduação) - Curso de Licenciatura em Engenharia de Construção Civil, Universidade Jean Piaget de Cabo Verde, Cidade da Praia, 2010. Disponível em: <http://bdigital.cv.unipiaget.org:8080/jspui/bitstream/123456789/279/1/Auriza%20de%20Barros.pdf>. Acesso em: 10 nov. 2011.

CARNEGIE MELLON UNIVERSITY. **What is speech recognition?** Disponível em: <http://www.speech.cs.cmu.edu/comp.speech/Section6/Q6.1.html>. Acesso em: 10 nov. 2011.

GRABIANOWSKI, Ed. **Como funciona o reconhecimento de voz.** traduzido por HowStuffWorks Brasil. Disponível em: <http://informatica.hsw.uol.com.br/reconhecimento-de-voz.htm>. Acesso em: 10 nov. 2011.

LUMENVOX. **LumenVox Speech Engine.** Disponível em: <http://www.lumenvox.com/products/speech_engine/>. Acesso em: 10 nov. 2011.

MANOSSO, Radamés. **Fonemas da língua portuguesa.** Disponível em: <http://www.radames.manosso.nom.br/gramatica/fonemas.htm>. Acesso em: 10 nov. 2011.

MENDES, Daniele Quintela; OLIVEIRA, Marcio Ferreira da Silva (Org.). Tutorial de Redes Neurais: Aplicações em Bioinformática. Disponível em: <http://www.lncc.br/~labinfo/tutorialRN/>. Acesso em: 20 nov. 2011.

MICROSOFT. **Windows Speech Recognition.** Microsoft Accessibility. Disponível em: <http://www.microsoft.com/enable/products/windowsvista/speech.aspx>. Acesso em: 10 nov. 2011.

MORAIS, Edmilson da Silva. **Reconhecimento automático de fala continua empregando modelos híbridos ANN +HMM.** 1997. 124 f. Dissertação (Mestrado) - Curso de Engenharia Eletrica, Faculdade de Engenharia Elétrica e de Computação da Unicamp, Campinas, 1997. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=vtls000126018&fd=y>. Acesso em: 10 nov. 2011.

NUANCE. **Dragon Dictate.** Disponível em: <http://www.nuance.com/for-individuals/by-product/dragon-for-mac/dragon-dictate/index.htm>. Acesso em: 10 nov. 2011.

SMARTHOME. **What is Home Automation?** Disponível em: <http://www.smarthome.com/homeautomation.html>. Acesso em: 10 nov. 2011.

YNOGUTI, Carlos Alberto. **Reconhecimento de Fala Contínua Usando Modelos Ocultos de Markov.** 1999. 152 f. Tese (Doutorado) - Curso de Engenharia Elétrica, Faculdade de Engenharia Elétrica e de Computação da Unicamp, Campinas, 1999. Disponível em: <http://www.decom.fee.unicamp.br/lpdf/teses_pdf/Tese-Doutorado-Carlos_Alberto_Ynoguti.pdf>. Acesso em: 10 nov. 2011.

APÊNDICE A – Código fonte do programa – Designer

```
namespace Papagaio
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.textoTextBox = new System.Windows.Forms.TextBox();
            this.falarButton = new System.Windows.Forms.Button();
            this.ouvirButton = new System.Windows.Forms.Button();
            this.comandosButton = new System.Windows.Forms.Button();
            this.casaPictureBox = new System.Windows.Forms.PictureBox();
            this.degreesTimer = new System.Windows.Forms.Timer(this.components);
            this.comandosPanel = new System.Windows.Forms.Panel();
            this.label6 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.mapaPictureBox = new System.Windows.Forms.PictureBox();
            ((System.ComponentModel.ISupportInitialize)(this.casaPictureBox)).BeginInit();
            this.comandosPanel.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.mapaPictureBox)).BeginInit();
            this.SuspendLayout();
            //
            // textoTextBox
            //
            this.textoTextBox.Anchor =
((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right)));
            this.textoTextBox.Location = new System.Drawing.Point(250, 383);
            this.textoTextBox.Multiline = true;
            this.textoTextBox.Name = "textoTextBox";
            this.textoTextBox.Size = new System.Drawing.Size(324, 89);
            this.textoTextBox.TabIndex = 0;
            //
            // falarButton
            //
            this.falarButton.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Right)));
            this.falarButton.Location = new System.Drawing.Point(353, 478);
            this.falarButton.Name = "falarButton";
            this.falarButton.Size = new System.Drawing.Size(119, 23);
```

```
            this.falarButton.TabIndex = 1;
            this.falarButton.Text = "Falar";
            this.falarButton.UseVisualStyleBackColor = true;
            this.falarButton.Click += new System.EventHandler(this.falarButton_Click);
            //
            // ouvirButton
            //
            this.ouvirButton.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Right)));
            this.ouvirButton.Location = new System.Drawing.Point(251, 478);
            this.ouvirButton.Name = "ouvirButton";
            this.ouvirButton.Size = new System.Drawing.Size(96, 23);
            this.ouvirButton.TabIndex = 4;
            this.ouvirButton.Text = "Começar a ouvir";
            this.ouvirButton.UseVisualStyleBackColor = true;
            this.ouvirButton.Click += new System.EventHandler(this.ouvirButton_Click);
            //
            // comandosButton
            //
            this.comandosButton.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Right)));
            this.comandosButton.Location = new System.Drawing.Point(478, 478);
            this.comandosButton.Name = "comandosButton";
            this.comandosButton.Size = new System.Drawing.Size(99, 23);
            this.comandosButton.TabIndex = 5;
            this.comandosButton.Text = "Exibir comandos";
            this.comandosButton.UseVisualStyleBackColor = true;
            this.comandosButton.Visible = false;
            this.comandosButton.Click += new System.EventHandler(this.comandosButton_Click);
            //
            // casaPictureBox
            //
            this.casaPictureBox.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
                        | System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right)));
            this.casaPictureBox.Location = new System.Drawing.Point(12, 12);
            this.casaPictureBox.MinimumSize = new System.Drawing.Size(339, 339);
            this.casaPictureBox.Name = "casaPictureBox";
            this.casaPictureBox.Size = new System.Drawing.Size(562, 365);
            this.casaPictureBox.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.casaPictureBox.TabIndex = 15;
            this.casaPictureBox.TabStop = false;
            //
            // degreesTimer
            //
            this.degreesTimer.Interval = 10;
            this.degreesTimer.Tick += new System.EventHandler(this.degreesTimer_Tick);
            //
            // comandosPanel
            //
            this.comandosPanel.Controls.Add(this.label6);
            this.comandosPanel.Controls.Add(this.label4);
            this.comandosPanel.Controls.Add(this.label5);
            this.comandosPanel.Controls.Add(this.label3);
            this.comandosPanel.Controls.Add(this.label2);
            this.comandosPanel.Controls.Add(this.label1);
            this.comandosPanel.Location = new System.Drawing.Point(144, 120);
            this.comandosPanel.Name = "comandosPanel";
            this.comandosPanel.Size = new System.Drawing.Size(328, 100);
            this.comandosPanel.TabIndex = 16;
            this.comandosPanel.Visible = false;
            //
            // label6
            //
            this.label6.Location = new System.Drawing.Point(0, 80);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(329, 18);
            this.label6.TabIndex = 18;
            this.label6.Text = "More [frase] - Concatena a frase ditada na caixa de texto.";
            //
```

```
            // label4
            //
            this.label4.Location = new System.Drawing.Point(0, 65);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(329, 15);
            this.label4.TabIndex = 17;
            this.label4.Text = "Say [frase] - Repete a frase ditada com uma voz sintetizada.";
            //
            // label5
            //
            this.label5.Location = new System.Drawing.Point(0, 2);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(329, 18);
            this.label5.TabIndex = 16;
            this.label5.Text = "Para utilizar os comandos falados o programa precisa estar ouvindo.";
            //
            // label3
            //
            this.label3.Location = new System.Drawing.Point(0, 50);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(329, 15);
            this.label3.TabIndex = 15;
            this.label3.Text = "Message [frase] - Joga a frase ditada numa caixa de menságem.";
            //
            // label2
            //
            this.label2.Location = new System.Drawing.Point(0, 35);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(329, 15);
            this.label2.TabIndex = 14;
            this.label2.Text = "Show [frase] - Escreve a frase ditada na caixa de texto.";
            //
            // label1
            //
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label1.Location = new System.Drawing.Point(0, 20);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(329, 15);
            this.label1.TabIndex = 13;
            this.label1.Text = "Comandos:";
            //
            // mapaPictureBox
            //
            this.mapaPictureBox.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
            this.mapaPictureBox.Image = global::Papagaio.Properties.Resources.Mapa;
            this.mapaPictureBox.Location = new System.Drawing.Point(12, 383);
            this.mapaPictureBox.Name = "mapaPictureBox";
            this.mapaPictureBox.Size = new System.Drawing.Size(232, 121);
            this.mapaPictureBox.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.mapaPictureBox.TabIndex = 21;
            this.mapaPictureBox.TabStop = false;
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(586, 516);
            this.Controls.Add(this.mapaPictureBox);
            this.Controls.Add(this.comandosPanel);
            this.Controls.Add(this.casaPictureBox);
            this.Controls.Add(this.comandosButton);
            this.Controls.Add(this.ouvirButton);
            this.Controls.Add(this.falarButton);
            this.Controls.Add(this.textoTextBox);
            this.MinimumSize = new System.Drawing.Size(602, 552);
            this.Name = "Form1";
            ((System.ComponentModel.ISupportInitialize)(this.casaPictureBox)).EndInit();
            this.comandosPanel.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)(this.mapaPictureBox)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();
```

```
        }

        #endregion

        private System.Windows.Forms.TextBox textoTextBox;
        private System.Windows.Forms.Button falarButton;
        private System.Windows.Forms.Button ouvirButton;
        private System.Windows.Forms.Button comandosButton;
        private System.Windows.Forms.PictureBox casaPictureBox;
        private System.Windows.Forms.Timer degreesTimer;
        private System.Windows.Forms.Panel comandosPanel;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.PictureBox mapaPictureBox;
    }
}
```

APÊNDICE B – Código fonte do programa – Formulário

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using SpeechLib;
using System.Speech.Synthesis;
using System.Speech.Recognition;
using System.Threading;


namespace Papagaio
{
    public partial class Form1 : Form
    {
        //Fala
        //SpeechLib
        SpSharedRecoContext objRecoContext;
        ISpeechRecoGrammar grammar;
        string strData = "No recording yet";
        SpLexicon addlex = new SpLexicon();

        //SDK
        SpeechRecognitionEngine recognitionEngine = new SpeechRecognitionEngine(new
            System.Globalization.CultureInfo("en-US"));

        //Desenho
        Pen pBordas = Pens.Black;
        Brush bApagado = Brushes.LightGray;
        Brush bAceso = Brushes.LightYellow;

        Brush bquarto1 = Brushes.LightGray;
        Brush bquarto2 = Brushes.LightGray;
        Brush bcozinha = Brushes.LightGray;
        Brush bbanheiro = Brushes.LightGray;
        Brush bsala = Brushes.LightGray;
        Brush bcorredor = Brushes.LightGray;

        Point[] quarto1 = { new Point(0, 0), new Point(250, 0), new Point(250, 175), new Point(0, 175),
new Point(0, 0) };
        Point[] quarto2 = { new Point(0, 175), new Point(200, 175), new Point(200, 350), new Point(0,
350), new Point(0, 175) };
        Point[] cozinha = { new Point(250, 0), new Point(500, 0), new Point(500, 175), new Point(250,
175), new Point(250, 0) };
        Point[] banheiro = { new Point(200, 210), new Point(300, 210), new Point(300, 350), new
Point(200, 350), new Point(200, 210) };
        Point[] sala = { new Point(300, 175), new Point(500, 175), new Point(500, 350), new Point(300,
350), new Point(300, 175) };
        Point[] ventilador = { new Point(0, 0), new Point(0, 0), new Point(0, 0), new Point(0, 0) };
        Point[] corredor = { new Point(200, 175), new Point(300, 175), new Point(300, 210), new
Point(200, 210), new Point(200, 175) };

        Double position = 0;
        Int16 speed = 0;

        public Form1()
        {
            InitializeComponent();
            degreesTimer.Start();
        }

        private void falarButton_Click(object sender, EventArgs e)
        {
            //Síntese de voz:
            SpeechSynthesizer synth = new SpeechSynthesizer();
            synth.Speak(textoTextBox.Text);
```

```
        }


        private void RecoSystemSpeech()
        {
            try
            {
                objRecoContext = new SpeechLib.SpSharedRecoContext();

                recognitionEngine.SetInputToDefaultAudioDevice();

                recognitionEngine.LoadGrammar(new DictationGrammar());

                // Add a handler for the speech recognized event.
                recognitionEngine.SpeechRecognized += new EventHandler <SpeechRecognizedEventArgs>
(recognitionEngine_SpeechRecognized);

                recognitionEngine.RecognizeAsync(RecognizeMode.Multiple);

                System.Threading.Thread.Sleep(600);

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }

        }

        // Handle the SpeechRecognized event.
        public void recognitionEngine_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
        {
            string strData;

            strData = e.Result.Text.ToLower();

            //MessageBox.Show("Recognized text: " + strData);

            if (strData.Length > 0)
            {
                if (strData.StartsWith("show "))
                {
                    textoTextBox.Text = strData.Substring(4);
                }
                else if (strData.StartsWith("message "))
                {
                    MessageBox.Show(strData.Substring(7));
                }
                else if (strData.StartsWith("say "))
                {
                    SpVoice voice = new SpVoice();
                    voice.Speak(strData.Substring(3), SpeechVoiceSpeakFlags.SVSFDefault);
                }
                else if (strData.StartsWith("more "))
                {
                    textoTextBox.Text = textoTextBox.Text + strData.Substring(4);
                }
                else if (strData.StartsWith("a senior luzes"))
                {
                    bquarto1 = bAceso;
                    bquarto2 = bAceso;
                    bsala = bAceso;
                    bcorredor = bAceso;
                    bcozinha = bAceso;
                    bbanheiro = bAceso;

                    draw();
                }
                else if (strData.StartsWith("reset"))
                {
                    bquarto1 = bApagado;
                    bquarto2 = bApagado;
                    bsala = bApagado;
                    bcorredor = bApagado;
```

```
    bcozinha = bApagado;
    bbanheiro = bApagado;

    draw();
}
else if (strData.StartsWith("bedroom one"))
{
    if (bquarto1 == bApagado)
    {
        bquarto1 = bAceso;
    }
    else
    {
        bquarto1 = bApagado;
    }
}
else if (strData.StartsWith("bedroom two"))
{
    if (bquarto2 == bApagado)
    {
        bquarto2 = bAceso;
    }
    else
    {
        bquarto2 = bApagado;
    }
}
else if (strData.StartsWith("kitchen"))
{
    if (bcozinha == bApagado)
    {
        bcozinha = bAceso;
    }
    else
    {
        bcozinha = bApagado;
    }
}
else if (strData.StartsWith("keaton"))
{
    if (bcozinha == bApagado)
    {
        bcozinha = bAceso;
    }
    else
    {
        bcozinha = bApagado;
    }
}
else if (strData.StartsWith("fan"))
{
    if (speed == 2)
    {
        speed = 1;
    }
    else if (speed == 1)
    {
        speed = 0;
    }
    else
    {
        speed = 2;
    }
}
else if (strData.StartsWith("fed"))
{
    if (speed == 2)
    {
        speed = 1;
    }
    else if (speed == 1)
    {
        speed = 0;
    }
```

```
                    else
                    {
                        speed = 2;
                    }
                }
                else if (strData.StartsWith("room"))
                {
                    if (bsala == bApagado)
                    {
                        bsala = bAceso;
                    }
                    else
                    {
                        bsala = bApagado;
                    }
                }
                else if (strData.StartsWith("shower"))
                {
                    if (bbanheiro == bApagado)
                    {
                        bbanheiro = bAceso;
                    }
                    else
                    {
                        bbanheiro = bApagado;
                    }
                }
                else if (strData.StartsWith("way"))
                {
                    if (bcorredor == bApagado)
                    {
                        bcorredor = bAceso;
                    }
                    else
                    {
                        bcorredor = bApagado;
                    }
                }
            }
        }


        public static void RecognitionEvent(int i, object o, SpeechLib.SpeechRecognitionType srt,
SpeechLib.ISpeechRecoResult isrr)
        {
            //Pelo SpeechLib estou tentando usar a seguinte sintaxe:
            string strText = isrr.PhraseInfo.GetText(0, -1, true);
            Console.WriteLine("recognized: " + strText);
        }

        private void ouvirButton_Click(object sender, EventArgs e)
        {
            {
                try
                {
                    RecoSystemSpeech();
                }
                catch (Exception ex)
                {
                    System.Windows.Forms.MessageBox.Show("Exception caught when initializing SAPI." + "
This application may not run correctly.\r\n\r\n" + ex.ToString(), "Error");
                }
            }

        }

        private void comandosButton_Click(object sender, EventArgs e)
        {
            comandosPanel.Visible = !comandosPanel.Visible;
        }

        void draw()
        {
            int x = 525; //500
```

```
            int y = 367; //350

            Bitmap bitmap = new Bitmap(x, y);

            Graphics image = Graphics.FromImage(bitmap);

            image.FillPolygon(bquarto1, quarto1);
            image.FillPolygon(bquarto2, quarto2);
            image.FillPolygon(bcozinha, cozinha);
            image.FillPolygon(bsala, sala);
            image.FillPolygon(bbanheiro, banheiro);
            image.FillPolygon(bcorredor, corredor);

            image.DrawLines(pBordas, quarto1);
            image.DrawLines(pBordas, quarto2);
            image.DrawLines(pBordas, cozinha);
            image.DrawLines(pBordas, sala);
            image.DrawLines(pBordas, banheiro);
            image.DrawLines(pBordas, ventilador);

            casaPictureBox.Image = bitmap;

        }

        private void degreesTimer_Tick(object sender, EventArgs e)
        {
            position = position + speed;

            //Após o incremento
            if (position >= 360)
            {
                position = 0;
            }

            ventilador = ventiladorPoints(40, position);

            draw();
        }

        Point[] ventiladorPoints(int size, double position)
        {
            //Declara 5 pontos
            Point[] points = {new Point(0,0), new Point(0,0), new Point(0,0), new Point(0,0), new
Point(0,0)};

            //Cada lado tem uma defasagem em graus dependendo do número de lados
            double degreesPerSide = (2 * Math.PI) / 4;

            double degree;

            //Define as coordenadas de cada ponto de acordo com o grau e o tamanho
            points[2] = new Point(400, 267);

            for (int i = 0; i <= 3; i++)
            {
                //Define o grau de cada ponto
                degree = degreesPerSide * i + (position / 180 * Math.PI);

                if (i == 0)
                {
                    //Define as coordenadas de cada ponto de acordo com o grau e o tamanho
                    points[0] = new Point(Convert.ToInt32(Math.Sin(degree) * size + 400),
                                          Convert.ToInt32(Math.Cos(degree) * size + 267)
                                         );
                }
                if (i == 1)
                {
                    //Define as coordenadas de cada ponto de acordo com o grau e o tamanho
                    points[3] = new Point(Convert.ToInt32(Math.Sin(degree) * size + 400),
                                          Convert.ToInt32(Math.Cos(degree) * size + 267)
                                         );
                }
                if (i == 2)
                {
```

```
                    //Define as coordenadas de cada ponto de acordo com o grau e o tamanho
                    points[1] = new Point(Convert.ToInt32(Math.Sin(degree) * size + 400),
                                          Convert.ToInt32(Math.Cos(degree) * size + 267)
                                         );
                }
                if (i == 3)
                {
                    //Define as coordenadas de cada ponto de acordo com o grau e o tamanho
                    points[4] = new Point(Convert.ToInt32(Math.Sin(degree) * size + 400),
                                          Convert.ToInt32(Math.Cos(degree) * size + 267)
                                         );
                }

            }

            return points;
        }

    }
}
```