UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

JUAN CARLOS URIBE PORTO

UM TUTORIAL DE CONTROLE DE VOZ PARA ROS E RESULTADOS

FORTALEZA

2023

JUAN CARLOS URIBE PORTO

UM TUTORIAL DE CONTROLE DE VOZ PARA ROS E RESULTADOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Fabrício Gonzalez Nogueira

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

P881t Porto, Juan Carlos Uribe.
Um tutorial de controle de voz para ROS e resultados / Juan Carlos Uribe Porto. – 2023.
43 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza, 2023.
Orientação: Prof. Dr. Fabrício Gonzalez Nogueira.

1. Controle de voz. 2. Python. 3. ROS. I. Título.

CDD 621.3

JUAN CARLOS URIBE PORTO

UM TUTORIAL DE CONTROLE DE VOZ PARA ROS E RESULTADOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Fabrício Gonzalez Nogueira
(Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Bismark Claure Torrico
Universidade Federal do Ceará (UFC)

Eng. Bruno Luiz Faustino
Universidade Federal do Ceará (UFC)

Dedico à minha mãe.

AGRADECIMENTOS

À Éris.

Ao Prof. Dr. Fabrício Gonzalez Nogueira por me orientar no meu trabalho de conclusão de curso.

Aos professores Drª. Natália Maria Cordeiro Barroso, Dr. Jonatan Floriano da Silva e Dr. Raimundo Alves Leitao Junior por me orientarem durante a graduação.

Aos colegas Engª. Juliana Carvalho, Eng. Filipe Virgolino, Eng. Danilo Soares, Eng. Aaron Marques e Eng. Enzo Furlan, todos me ajudaram em algum ponto do percurso.

Ao Físico Danilo Augusto que conheci na graduação e se tornou um bom amigo.

À minha professora e grande amiga Yuka Ito.

Às minhas amigas Juliane Souza e Sabrina Souza por me suportarem por tanto tempo.

À Cinthya Sampaio, minha heroína, salvou minha vida.

À professora Lingwei Zhang.

Ao Dr. Epitácio pela oportunidade de estágio.

Ao Eng. Renan Moura por todo suporte que me deu em meu estágio.

Ao NUTEC, instituição na qual estagiei.

Aos membros da banca avaliadora Prof. Dr. Bismark Claure Torrico e ao Eng. Bruno Luiz Faustino.

"La vanité et le bonheur sont incompatibles."

(Choderlos de Laclos)

RESUMO

O âmbito deste trabalho é o sistema de comandos por voz, mostrando as dificuldades de sua implementação. Visto que a utilização desta tecnologia é muito grande atualmente devido a popularização dos sistemas integrados aos smartphones, computadores, caixas de som e televisões. Sendo também uma ferramenta importante para acessibilidade de pessoas com deficiencia. Para essa finalidade será utilizado os sistema operacional conhecido como sistema ROS (*Robot Operating System*), que é uma plataforma de *software* amplamente utilizada em robótica. Ele fornece uma infraestrutura para a comunicação entre os componentes do robô, além de oferecer uma ampla variedade de ferramentas para desenvolvimento e controle de robôs. Nesse contexto, a implementação de um sistema de controle de voz para o ROS pode ser uma solução interessante para permitir que os usuários controlem os robôs de forma eficiente e intuitiva. Também será ultilizada a linguagem *Python*, ela será usada para criar o sistema de reconhecimento de voz, integrar e automatizar junto ao ROS, permitindo que os usuários possam interagir com um robô de forma natural e intuitiva.

Palavras-chave: Controle por voz. ROS. Python.

ABSTRACT

The scope of this work is the voice command system, showing the difficulties of its implementation. Since the use of this technology is currently very large due to the popularization of systems integrated into smartphones, computers, speakers and televisions. It is also an important tool for accessibility for people with disabilities. For this purpose, the operating system known as ROS (Robot Operating System) will be used, which is a software platform widely used in robotics. It provides an infrastructure for communication between robot components, as well as offering a wide range of tools for robot development and control. In this context, the implementation of a voice control system for ROS can be an interesting solution to allow users to control robots efficiently and intuitively. The Python language will also be used, it will be used to create the voice recognition system, integrate and automate it with ROS, allowing users to interact with a robot in a natural and intuitive way.

Keywords: Voice Control. ROS. Python.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
LTS	Long Term Service
ROS	Robot Operational System

LISTA DE SÍMBOLOS

A	Ampere
V	Volt

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, a robótica tem se desenvolvido rapidamente e se tornado uma das áreas mais fascinantes da tecnologia. A robótica está cada vez mais presente em nossas vidas, seja em fábricas, hospitais, residências ou até mesmo em missões espaciais. Um dos desafios enfrentados pelos robôs é a interação com o ambiente e com os seres humanos, o que exige interfaces intuitivas e fáceis de usar. Nesse contexto, o controle de voz tem se tornado uma alternativa interessante e promissora para controlar robôs.

O *Robot Operational System (ROS)* é um sistema operacional de código aberto para robôs que permite a criação de sistemas complexos de robôs que podem ser controlados remotamente. *Python* é uma das linguagens de programação suportadas para trabalhar com o ROS e permite o desenvolvimento de uma variedade de aplicações para robótica, incluindo o controle de voz.

Este trabalho tem como objetivo apresentar o desenvolvimento de um sistema de controle de voz em *Python* para o ROS. Serão abordados conceitos de processamento de linguagem natural, comunicação entre *Python* e ROS. O sistema proposto permitirá ao usuário controlar um robô usando comandos de voz de forma intuitiva e natural.

Espera-se que o sistema desenvolvido possa contribuir para o avanço da robótica e da interação homem-máquina. Além disso, o trabalho também pretende explorar as possibilidades da linguagem Python para o desenvolvimento de aplicações em robótica e demonstrar a viabilidade do uso de controle de voz em sistemas robóticos.

1.1 Motivação

A robótica tem se expandido rapidamente nas últimas décadas, tornando-se uma área em constante desenvolvimento. Com o crescente uso de robôs em várias aplicações, a necessidade de interfaces intuitivas e fáceis de usar também cresceu. A capacidade de controlar um robô por meio de voz é uma alternativa promissora e que pode melhorar significativamente a interação homem-máquina, uma vez que a voz é uma forma natural e universal de comunicação.

O controle de voz tem sido usado em vários tipos de aplicação, desde assistentes pessoais até sistemas de controle de robôs. O ROS, como um sistema operacional de robôs de código aberto, é amplamente utilizado em muitos tipos de robôs e fornece uma plataforma para o desenvolvimento de *software* de controle de robôs. O Python, por sua vez, é uma linguagem de programação simples para trabalhar com ROS, permitindo o desenvolvimento de aplicativos de robótica eficientes.

O desenvolvimento de um sistema de controle de voz em *Python* para ROS pode melhorar significativamente a eficiência e a precisão do controle de robôs. A utilização de comandos de voz pode permitir o controle simultâneo de várias tarefas e pode melhorar a interação homem-máquina, permitindo que os usuários controlem o robô de forma mais intuitiva. Além disso, o controle de voz pode ser usado para fornecer feedback em tempo real, permitindo que o usuário avalie a eficácia do controle e faça ajustes instantaneamente.

O desenvolvimento de um sistema de controle de voz em *Python* para ROS tem o potencial de melhorar significativamente a eficiência e a eficácia do controle de robôs. O uso de comandos de voz pode tornar o controle de robôs mais efetivo, melhorando a interação homem-máquina. Portanto, este trabalho tem como objetivo contribuir para o avanço da robótica e demonstrar a viabilidade do uso de controle de voz em sistemas robóticos.

1.2 Objetivos

- Controlar o robô por voz (*Raspberry PI*).
- Estudo sobre reconhecimento de voz.
- Estudo de controle em Python.
- Controle para ROS.

Objetivo é desenvolver um programa que controle um robô em tempo real através do ROS sendo excutado no sitema *Ubuntu*, sendo o *Ubuntu* compativel com o *Raspberry PI*. O sistema será capaz de reconhecer comandos de voz e transmiti-los para o robô, permitindo que ele execute tarefas complexas de forma autônoma. Sendo feito desenvolvimento de um algoritmo de reconhecimento de fala que possa ser usado para detectar e interpretar comandos de voz em tempo real, integração do algoritmo de reconhecimento de fala com o ROS, permitindo que os comandos de voz sejam transmitidos para os nós do ROS e executados pelo robô.

1.3 Premissa

Primeiro será desenvolvido um programa em *Python* que seja capaz de reconhecer o sinal de voz e transcrever. Depois um segundo programa que receba e processe essa informação e converta em um comando do ROS. Por último serão feitos os testes de validação no *Raspberry PI*.

2 FUNDAMENTAÇÃO TEORICA

Neste capítulo serão trata alguns tópicos importantes para o desenvolvimento deste trabalho.

2.1 *Python*

Python é uma linguagem de programação de alto nível, criada em 1991 pelo programador Guido van Rossum. Ela é uma linguagem interpretada, o que significa que o código-fonte é executado diretamente por um interpretador, sem necessidade de compilação prévia (INSTITUTE, 2023).

A linguagem *Python* tem sido é bastante utilizada em diferentes áreas, como programação web, desenvolvimento de jogos, análise de dados, inteligência artificial, entre outras. Uma das razões para sua popularidade é sua sintaxe simples e legível, que facilita a compreensão do código mesmo por pessoas que não têm conhecimentos avançados em programação (INSTITUTE, 2023).

Python é uma linguagem multiplataforma, o que significa que ela pode ser executada em diferentes sistemas operacionais, como *Windows*, *macOS* e *Linux*, além de poder ser integrada com outras linguagens de programação. *Python* também é uma linguagem de código aberto, o que significa que o código-fonte está disponível para que qualquer pessoa possa modificar e distribuir livremente. Isso torna mais fácil a colaboração entre desenvolvedores e possibilita o surgimento de novas soluções (SANTANA, 2023).

Uma das principais vantagens de *Python* é a grande comunidade de desenvolvedores que contribuem com bibliotecas, tutoriais e fóruns de discussão. Sendo assim mais fácil aprender a linguagem e resolver problemas que surgem durante o desenvolvimento.

Python oferece uma série de bibliotecas e ferramentas que permitem o desenvolvimento rápido e fácil de aplicações de controle de voz. A biblioteca Speech-Recognition permite o processamento de áudio em tempo real e o reconhecimento de diferentes idiomas e sotaques (ZHANG, 2014).

A linguagem *Python* está constantemente evoluindo, o que traz novas funcionalidades e melhorias de desempenho.

2.2 *Linux*

Linux é um sistema operacional de código aberto baseado em *Unix*, que é usado em uma variedade de plataformas, incluindo servidores, *desktops*, dispositivos móveis e até mesmo em carros autônomos. Ele foi criado em 1991 por Linus Torvalds e tem sido um importante contribuinte para a evolução da tecnologia de *software* livre e de código aberto (MUNDI, 2021).

Um dos recursos do *Linux* é sua flexibilidade. Ele é personalizável e pode ser adaptado para atender a diferentes necessidades de computação. Ele também oferece uma variedade de distribuições, como *Ubuntu, Debian* e *Fedora*, cada uma com suas próprias características e foco em diferentes tipos de usuários (TEXEIRA, 2022).

Outra vantagem do *Linux* é sua comunidade de desenvolvedores e usuários. Como é um projeto de código aberto, qualquer pessoa pode contribuir com melhorias e correções de bugs. Isso significa que o *Linux* tem uma grande base de usuários dedicados que trabalham juntos para melhorar o sistema (ALECRIM, 2022). Em 2023, o *Linux* alcançou uma parcela de 3% dos usuários de desktop, representando sua maior participação até aquela data (STATCOUNTER, 2023).

O *Linux* é seguro. Como é de código aberto, ele permite que a comunidade de desenvolvedores trabalhe juntos para identificar e corrigir falhas de segurança mais rapidamente. Além disso, ele é menos suscetível a ataques de malware, pois sua arquitetura é mais segura do que a de outros sistemas operacionais (ASSISCITY, 2022).

O *Linux* também é escalável e pode ser adaptado para rodar em diferentes tipos de dispositivos, desde computadores pessoais até dispositivos móveis e até

mesmo em sistemas de controle de carros autônomos (MUNDI, 2021).

O *Linux* é um sistema operacional gratuito. Qualquer pessoa pode baixar e usar o sistema operacional sem pagar por ele. Isso é importante para organizações sem fins lucrativos e empresas que precisam economizar dinheiro em licenças de *software*.

No entanto, como qualquer sistema operacional, o *Linux* também tem suas desvantagens e limitações. Por exemplo, nem todos os programas são compatíveis com o sistema operacional, o que pode limitar suas opções de *software*. Além disso, pode haver problemas de compatibilidade de *hardware* com alguns dispositivos.

Apesar disso, o *Linux* é uma opção estável para um sistema operacional estável, seguro e altamente personalizável.

2.3 *Ubuntu*

O *Ubuntu* é um sistema operacional baseado em *Linux*, que tem como objetivo fornecer uma experiência simples e intuitiva para os usuários. Desenvolvido pela empresa *Canonical Ltd.*, o *Ubuntu* é uma das distribuições mais populares e largamente utilizadas no mundo do *software* livre, ocupando a oitava posição no Distrowatch (DISTROWATCH, 2023).

Uma das principais características do *Ubuntu* é a sua filosofia de *software* livre e código aberto. Isso significa que o sistema operacional é distribuído gratuitamente, permitindo que qualquer pessoa o utilize, modifique e compartilhe conforme suas necessidades. Isso promove a liberdade e a transparência, permitindo que os usuários tenham controle total sobre o sistema (UBUNTU, 2023).

Além disso, o *Ubuntu* é conhecido por sua facilidade de uso. Ele apresenta uma interface gráfica amigável e intuitiva, chamada de *Gnome*, que permite que os usuários naveguem facilmente pelo sistema operacional e acessem seus aplicativos e arquivos. O *Ubuntu* também é compatível com uma grande variedade de *hardware*, há parecerias com Dell, Lenovo e HP fazendo acessível para diferentes tipos de dispositivos, desde computadores desktop até servidores e dispositivos móveis (UBUNTU, 2023).

Duas das características importantes do *Ubuntu* é a sua estabilidade e segurança. A *Canonical* se dedica a fornecer atualizações regulares de segurança e correções de bugs, garantindo que o sistema operacional esteja protegido contra ameaças e vulnerabilidades. Além disso, o Ubuntu conta com uma ampla comunidade de usuários e desenvolvedores que oferecem suporte e colaboram para a melhoria contínua do sistema (UBUNTU, 2023).

O *Ubuntu* também se destaca por sua vasta biblioteca de *software*. Através de sua loja de aplicativos, os usuários podem acessar milhares de programas gratuitos e de código aberto para atender às suas necessidades. Essa variedade de *software* abrange desde aplicativos de escritório, como processadores de texto e planilhas, até ferramentas de desenvolvimento, reprodutores de mídia e jogos (UBUNTU, 2023).

Uma das versões mais conhecidas do *Ubuntu* é a *Long Term Service (LTS)*, que oferece suporte a longo prazo, com atualizações de segurança e correções de bugs por até cinco anos. Essa versão é ideal para usuários e empresas que buscam estabilidade e confiabilidade em seus sistemas (SPINUPWP, 2023).

O *Ubuntu* é um sistema operacional baseado em Linux, gratuito e de código aberto, que oferece uma experiência simples, estável e segura para os usuários. Com sua filosofia de *software* livre, interface amigável e vasta biblioteca de *software*, o *Ubuntu* atrai uma grande base de usuários.

2.4 ROS

O ROS é um conjunto de ferramentas e bibliotecas de código aberto que facilitam o desenvolvimento de *software* para robôs. Criado em 2007 pela Willow Garage, o ROS se tornou uma das plataformas mais populares e largamente utilizadas na comunidade de robótica (ROS, 2023b).

O ROS não é um sistema operacional tradicional, mas sim um framework flexível que fornece uma estrutura para facilitar a comunicação e a integração entre diferentes componentes de um robô. Ele é executado em cima de um sistema operacional real, como o Linux, fornecendo recursos como gerenciamento de *hardware*, controle de dispositivos e comunicação em tempo real (VARGAS IOHAN G.; TAVARES, 2013).

Uma vantagem do ROS é sua arquitetura distribuída. Ele permite que os desenvolvedores dividam o sistema do robô em módulos independentes chamados de "nós"que podem ser executados em diferentes computadores ou processadores. Esses nós podem se comunicar uns com os outros por meio de um sistema de troca de mensagens, permitindo a criação de sistemas complexos e altamente modularizados.

Além disso, o ROS oferece uma variedade de bibliotecas e ferramentas para auxiliar no desenvolvimento de *software* para robótica. Isso inclui bibliotecas para manipulação de imagens, processamento de dados, controle de movimento, simulação e muito mais. Essas bibliotecas ajudam os desenvolvedores a acelerar o processo de desenvolvimento, reduzindo a necessidade de escrever código do zero.

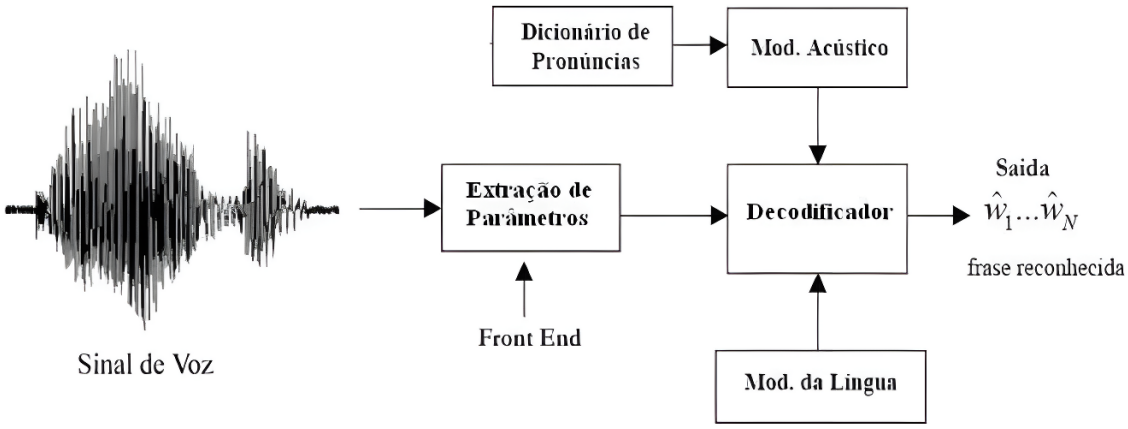
Uma das principais vantagens do ROS é a sua comunidade . Há uma quantidade de pacotes de *software* disponíveis desenvolvidos pela comunidade, que abrangem uma gama de funcionalidades e aplicações. Essa comunidade também fornece suporte, tutoriais e documentação que facilita a aprendizagem e o compartilhamento de conhecimento entre os desenvolvedores.

O ROS serve para projetos que envolvem robôs móveis, manipuladores robóticos e sistemas autônomos. Com sua arquitetura modular, recursos avançados e comunidade ativa, o ROS é um forte impulsionador da inovação na área de robótica.

2.5 Reconhecimento de voz

Reconhecimento de voz é um campo da inteligência artificial que permite que os computadores interpretem e compreendam a linguagem falada pelos seres humanos. O reconhecimento de voz tem aplicações em uma gama de setores e indústrias. Um dos exemplos mais comuns é a assistência virtual em dispositivos móveis e smart speakers, como a *Siri* da *Apple* e o *Google Assistant*. Esses assistentes pessoais usam o reconhecimento de voz para entender os comandos e perguntas dos usuários e fornecer respostas relevantes e úteis. O reconhecimento de voz é um processo complexo que envolve várias etapas.

Figura 1 – Blocos de um sistema de reconhecimento de voz.



Fonte: SILVA (2010)

Amostragem

Amostragem de audio é a primeira das etapas onde através de um microfone ou outro dispositivo de entrada o áudio do usuário é capturado, do qual é feita uma amostra em intervalos regulares para obter uma representação digital.

Algumas caracteríticas do sinal voz de acordo com (PLANNERER, 2005) são a largura de banda do sinal é de 4 Khz, o sinal é períodico e tem frequência fundamental entre 80 Hz e 350Hz, há picos na distribuição espectral de energia em

$$(2n-1)*500 \quad Hz; \quad n = 1,2,3... \tag{2.1}$$

e o envelope do espectro de potência do sinal mostra uma diminuição com o aumento da frequência.

O sinal de áudio capturado pode conter ruídos de fundo, variações de volume e outros artefatos indesejados. Nesta etapa, o sinal de áudio é pré-processado para remover o ruído e normalizar o volume, a fim de melhorar a qualidade do sinal e facilitar a análise posterior. O Pré-processamento consiste em pre-ênfase, segmentação, janelamento e transformada de Fourier.

Pré-ênfase

Na pré-ênfase é aplicado um filtro passa altas de primeira ordem com o intuito de reduzir os efeito dos pulsos glotais e impedância de radiação, visando assim evitar perda de dados durante o processo de segmentação(MORENO, 1996). A pré-ênfase realça as características da fala, especialmente aquelas presentes nas altas frequências, melhorando a inteligibilidade da fala em ambientes ruidosos ou com baixa qualidade de gravação.

Segmentação

Ao reconhecer sinais de voz, é essencial determinar com exatidão os pontos de início e término das palavras. Em outras palavras, é necessário distinguir as partes do sinal que contêm informações de voz das partes que não contêm, com o objetivo de reduzir o tempo de processamento. Consequentemente, o sinal de voz é segmentado em quadros relativamente pequenos, nos quais são assumidas características de quase-estacionariedade. Após a segmentação do sinal, é registrado como um vetor de atributos para ser processado (GORDILLO, 2013).

Janelamento

O janelamento é uma técnica utilizada para suavizar a descontinuidade causada pela segmentação. A janela de Hamming é a mais comumente empregada nesse contexto (PERICO ALISSON; SHINOHARA, 2014).

Transformada de Fourier

No reconhecimento de voz, o sinal é transformado em suas componentes de frequência, possibilitando assim a diferenciação entre as vozes de diferentes indivíduos. Essa transformação para o domínio de frequência permite a observação do número de vezes que ocorrem mudanças na amplitude do sinal, sendo por isso utilizada a transformada de Fourier. Sendo representada por

$$X(k) = \sum_{k=0}^{N-1} x[n] e^{\frac{-j2\Pi kn}{N}} \tag{2.2}$$

onde N representa o número de amostras (GORDILLO, 2013).

Extração de parâmetros

A proxima etapa é a extração de parâmentros como mostrado na figura 1, é onde são extraídas do sinal de áudio características acústicas relevantes. Essas características podem incluir a intensidade do som, a frequência fundamental, a duração dos fonemas e outros atributos que ajudam a distinguir os diferentes sons da fala. Existem várias técnicas de extração, uma delas sendo o banco de filtros para decompor o sinal de áudio em diferentes bandas de frequência, também há *Perceptual Linear Prediction* que é uma técnica que leva em consideração as características do sistema auditivo humano. Ela modela o processo de audição humana para capturar os elementos mais perceptíveis e relevantes para o reconhecimento de fala. *Mel Frequency Cepstral Coefficients* é uma das técnicas mais populares na extração de características da fala. Ela se baseia na análise das frequências do espectro de áudio e na aplicação da transformada de cosseno discreta aos valores dos log-mel espectros. O resultado é um conjunto compacto de coeficientes cepstrais, que capturam informações relevantes sobre a forma da onda sonora. Sendo *Mel Frequency Cepstral Coefficients* mais efetiva que *Perceptual Linear Prediction* (HONIG, 2005).

Dicionário fonético

A transformação de uma sequência de caracteres em uma sequência de fonemas é um requisito fundamental para serviços que abrangem reconhecimento e síntese de voz. Um dicionário é composto por um conjunto de palavras acompanhadas de suas transcrições fonéticas correspondentes. As sugestões de conversão, que seguem critérios fonológicos pré-definidos, são fornecidas ao sistema com base no idioma para o qual o aplicativo é direcionado (SILVA, 2010).

Modelagem de linguagem

Para realizar o reconhecimento de voz, que se encontra representado na figura 1, é necessário ter um modelo de linguagem que represente as palavras e frases possíveis que podem ser faladas. Esse modelo pode ser baseado em gramáticas ou em modelos estatísticos, como Modelos de Markov Ocultos (GORDILLO, 2013).

Pós-processamento

Após a decodificação, é feito um pós-processamento para refinar e melhorar os resultados. Isso pode incluir a correção de erros, a aplicação de regras gramaticais ou até mesmo a análise contextual para melhorar a precisão e a compreensão da transcrição.

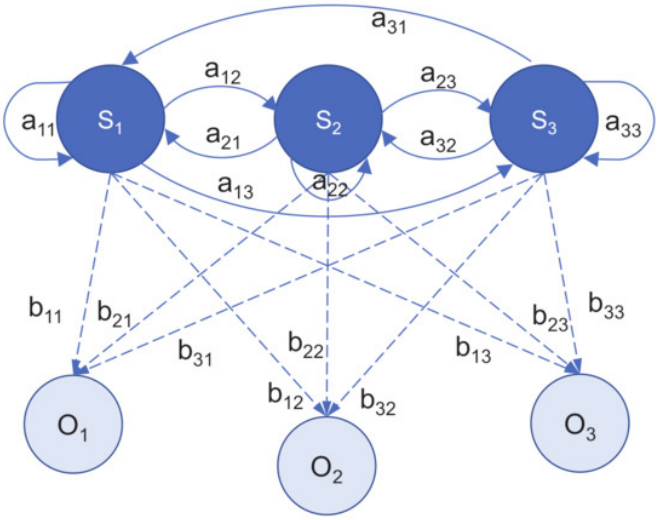
2.6 Modelos Ocultos de Markov

Os modelos ocultos de Markov (HMM) são um sistema estocástico duplo, consistindo em um processo não detectável, mas pode ser inferido por meio de outro processo estocástico (SOUZA, 2013).

Os processos ocultos são compostos por um conjunto de estados interligados por transições com probabilidades, enquanto os processos observáveis (não ocultos) consistem em um conjunto de saídas ou observações, onde cada uma pode ser emitida por cada estado, de acordo com uma função de densidade de probabilidade específica (OLIVEIRA LUIZ EDUARDO SOARES DE; MORITA, 2010).

Atualmente, os modelos de Ocultos de Markov (HMM) emergiu como a abordagem predominante no campo do reconhecimento de fala. Esses modelos estocásticos têm se mostrado especialmente adequados para caracterizar a variabilidade presente em sinais que variam ao longo do tempo. A principal vantagem do HMM é que ele é apropriado para lidar com sinais corrompidos por ruídos, como fala ou escrita, além de sua fundamentação teórica sólida. A existência de algoritmos consistentes permite ajustar automaticamente os parâmetros do modelo por meio de procedimentos iterativos (OLIVEIRA LUIZ EDUARDO SOARES DE; MORITA, 2010).

Figura 2 – Um modelo oculto de Markov com três estados e três estados de observação.



Fonte: ZHANG Mingchi; CHEN (2021)

Os modelos ocultos de Markov possuem uma fundamentação matemática sólida, garantindo a convergência para um ponto ótimo. Além disso, ele apresenta uma capacidade de treinamento eficiente, otimizando automaticamente os parâmetros a partir dos dados disponíveis. Outro aspecto importante é a utilização de técnicas de decodificação que permitem descrever uma sequência de entrada em termos da melhor sequência de estados possível.

Na Figura 2, encontra-se representado um modelo oculto de Markov em uma máquina de estados finita, que transita entre seus estados a cada unidade de tempo.

2.7 Programas Existentes

SpeechRecognition

O *SpeechRecognition* é uma biblioteca experimental que permite a integração de reconhecimento de fala em aplicações e projetos de programação.

Desenvolvida para ser versátil, a biblioteca SpeechRecognition suporta uma variedade de serviços de reconhecimento de fala, incluindo reconhecimento baseado em nuvem, bem como reconhecimento offline. Isso significa que você pode escolher entre diferentes provedores, como Google Speech Recognition, IBM Watson, Microsoft

Bing, CMU Sphinx, entre outros, de acordo com suas necessidades e preferências (ZHANG, 2014).

A biblioteca fornece uma sintaxe simples para implementar a funcionalidade de reconhecimento de fala em seu código. O processo envolve a captura de áudio do microfone, o envio desse áudio para o serviço de reconhecimento escolhido e a interpretação e manipulação dos resultados obtidos.

Pocketsphinx

O *Pocketsphinx* é um motor para reconhecimento de fala contínua para *Python*, mas o seu desenvolvimento já descontinuado, mas ainda sim ele está disponível com erros que são considerados clássicos (HUGGINS-DAINES, 2023).

3 DESENVOLVIMENTO

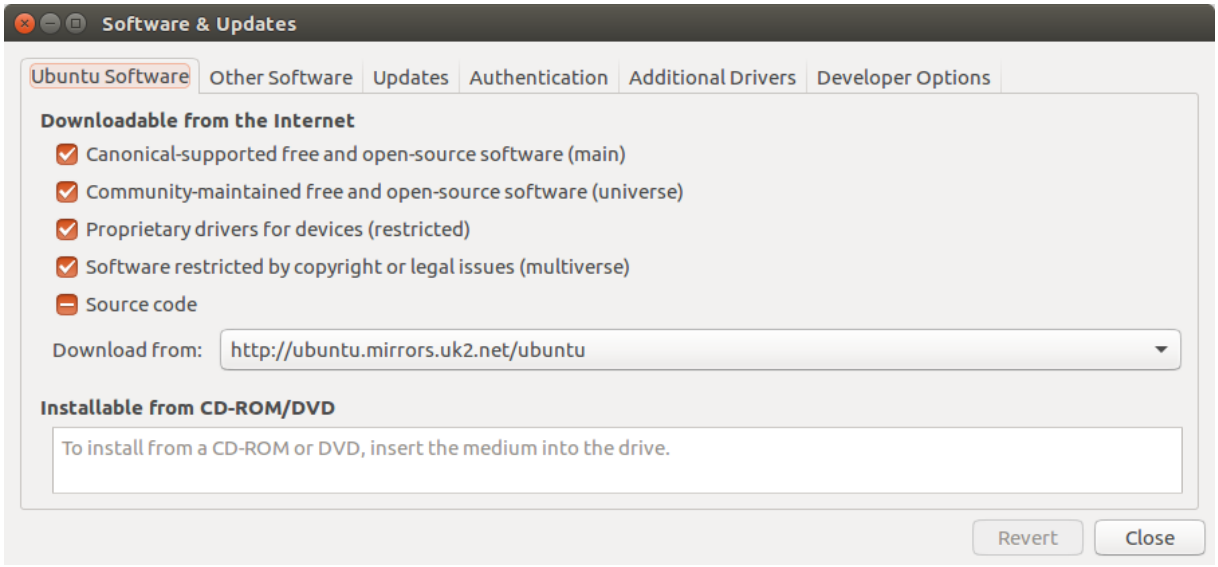
3.1 Instalação do ROS

Nesse capítulo serão tratadas as etapas para o desenvolvimento do programa.

Primeiro passo é a instalação do ROS em ambiente Linux Ubuntu de acordo com o site do projeto (ROS, 2023a), a versão escolhida pela compatibilidade foi ROS Noetic.

1. Configurar os repositórios do Ubuntu para permitir "*Universe*", "*Restricted*" e "*Multiverse*".

Figura 3 – Aba *Software* Ubuntu



Fonte: Ubuntu

2. Configurar o source.list para aceitar *software* de packages.ros.org.

```
1    sudo sh -c 'echo "deb http://packages.ros.org/ros/
       ubuntu \$(lsb\_release -sc) main" > /etc/apt/
       sources.list.d/ros-latest.list'
```

3. Configurar as Keys.

```
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

4. Fazer Update.

```
sudo apt update
```

5. Instalação.

```
sudo apt install ros-noetic-desktop-full
```

6. Configuração do ambiente.

```
echo "source /opt/ros/noetic/setup.bash" $\gg$  \textasciitilde /.bashrc
source ~/.bashrc
```

7. Dependências para construir pacotes.

```
sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
```

8. Inicializar Rosdep.

```
sudo apt install python3-rosdep
```

9. Criar uma área de trabalho ROS.

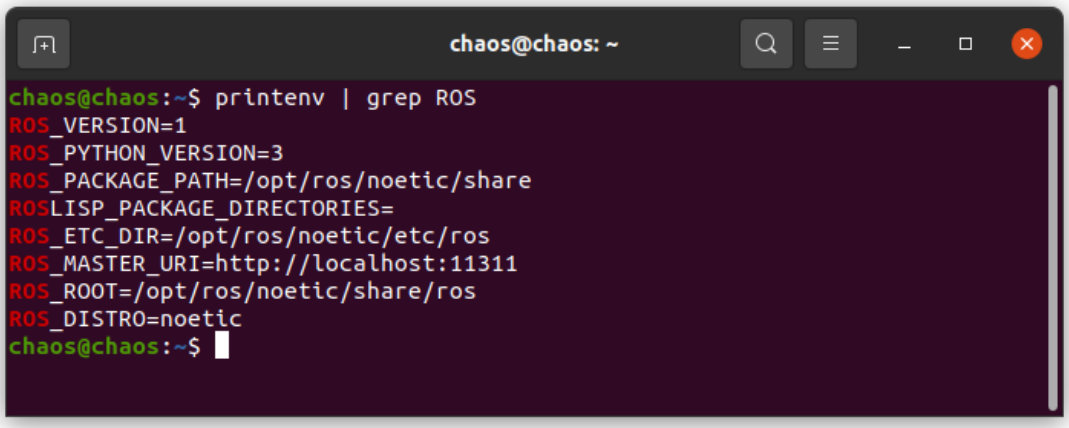
```
\$ mkdir -p \textasciitilde /catkin\_ws/src
\$ cd ~/catkin\_ws/
\$ catkin\_make
```

Após isso foi usado o comando

```
\$ printenv | grep ROS
```

que retornou os sequintes configrações

Figura 4 – Configurações do ROS



Fonte: Autoria própria.

Próximo passo é a instalação da biblioteca *SpeechRecognition*.

```
pip install SpeechRecognition
```

O código de reconhecimento de voz foi desenvolvido baseado no código básico do *SpeechRecognition* encontrado no *site* (LOGOS, 2021).

Também foi instalada a biblioteca de automação *pyautogui*.

```
pip install pyautogui
```

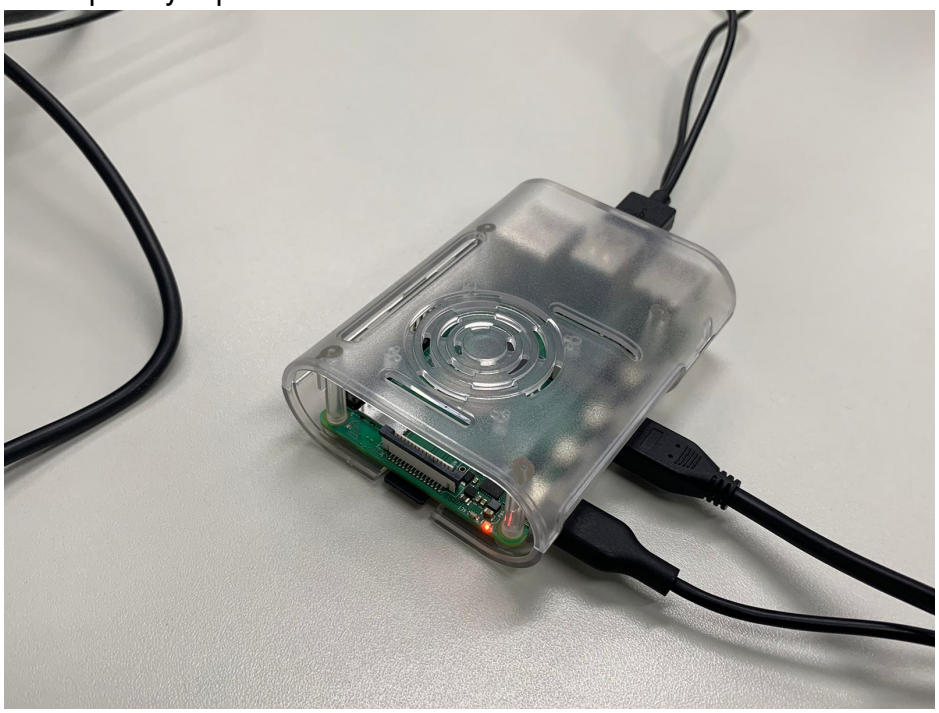
O programa de reconhecimento de voz se encontra no apêndice A.

O Próximo passo foi desenvolvimento do programa de automatização que faz a comunicação entre o Programa de reconhecimento de voz e o ROS, que também se encontra no apêndice A.

3.2 Instalando o *Ubuntu* no *Raspberry PI*

Foi necessária a instalação do *Ubuntu Server 20.04* no *Raspberry PI*, pois era a versão recomendada pelo ROS.

Figura 5 – Raspberry e periféricos.



Fonte: Autoria própria

Foram conectadas as entradas um fonte de 5V e 3A para que possa alimentar os demais periféricos, o cabo HDMI para ligar o *Raspberry* ao monitor, o teclado e

mouse.

As etapas para instalação foram as seguintes:

1. Foi baixada a imagem do *Ubuntu Server 20.04* para *Raspberry PI 4*. A imagem oficial pode ser encontrada no *site* do *Ubuntu*.

2. Após o *download*, foi verifiada a integridade do arquivo da imagem utilizando seu algoritmo de *hash* o SHA256.

3. Foi inserido um cartão microSD no seu computador.

4. Foi feita a formatação do cartão microSD, através do gerenciador de discos do *Ubuntu*.

5. Usando o gerenciador de discos foi escrita a imagem no cartão microSD.

6. Após a gravação o cartão microSD foi removido do computador.

7. Foi inserido o cartão microSD no slot do *Raspberry PI*.

8. Quando ligado e conectado o *Raspberry*, o processo de inicialização começa e a tela de configuração do *Ubuntu Server 20.04* foi exibida.

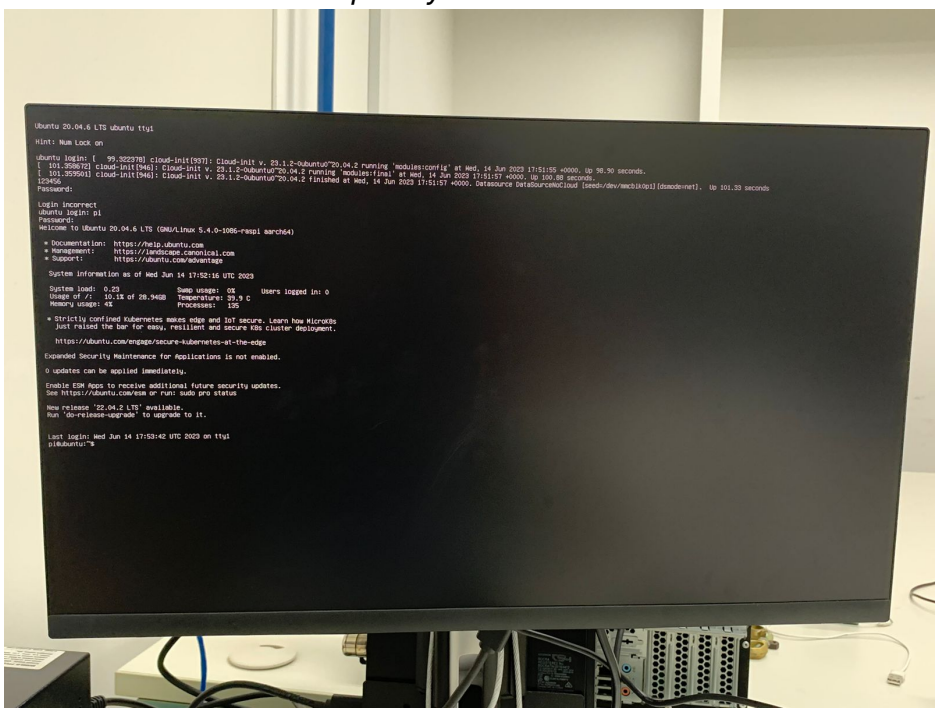
9. Configurar senha.

10. Instalação de pacotes necessários.

11. Quando concluído o *Raspberry* foi reiniciado.

Após isso foi feita a instalação do ROS no *Raspberry*.

Figura 6 – *Ubuntu Server* no *Raspberry*.



Fonte: Autoria própria

3.3 Custos do Projeto

Com intuito de concretizar este projeto, é preciso adquirir certos materiais, cujo custo totaliza R$ 1057,48.

Tabela 1 – Custos do projeto

Descrição	Preço
Cabo HDMI	R$ 19,90
Raspberry PI 4	R$ 938,99
Cartão microSD 32 GB	R$ 35,90
Microfone USB	R$ 22,79
Fonte 5V 3A	R$ 39,90
Total	R$ 1057,48

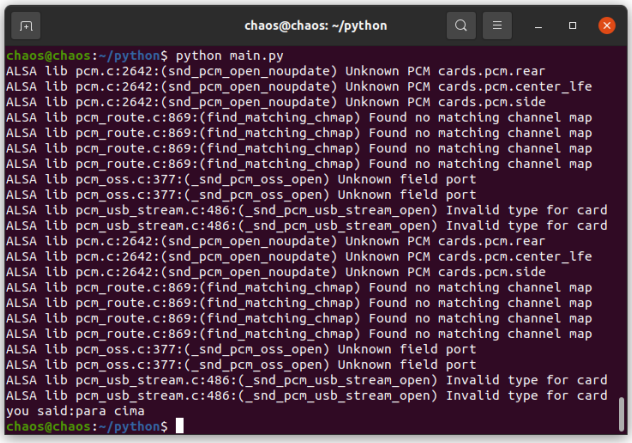
Fonte: Autoria própria.

4 RESULTADOS

4.1 Programa 1

Primeiro foi testado o código 1 do apêndice A e o resultado foi o desejado.

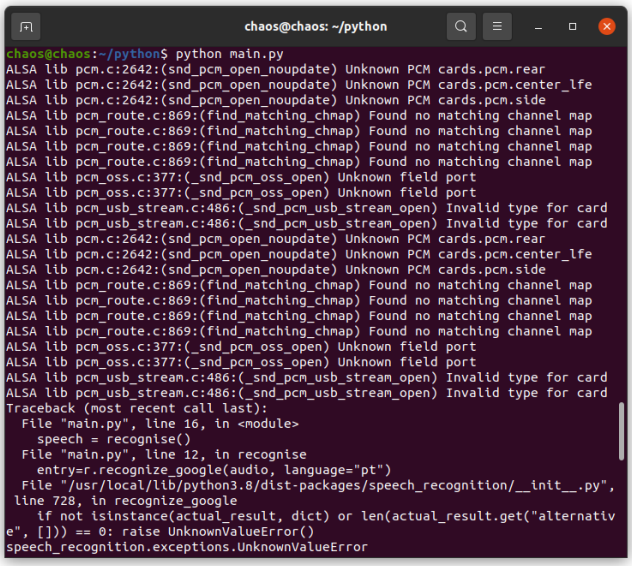
Figura 7 – Execução do programa de reconhecimento de voz



Fonte: Autoria própria

Apesar de ter sido alcançado o resultado desejado, houve uma série de erros na detecção de voz devido a ruídos simples.

Figura 8 – erro no reconhecimento de voz



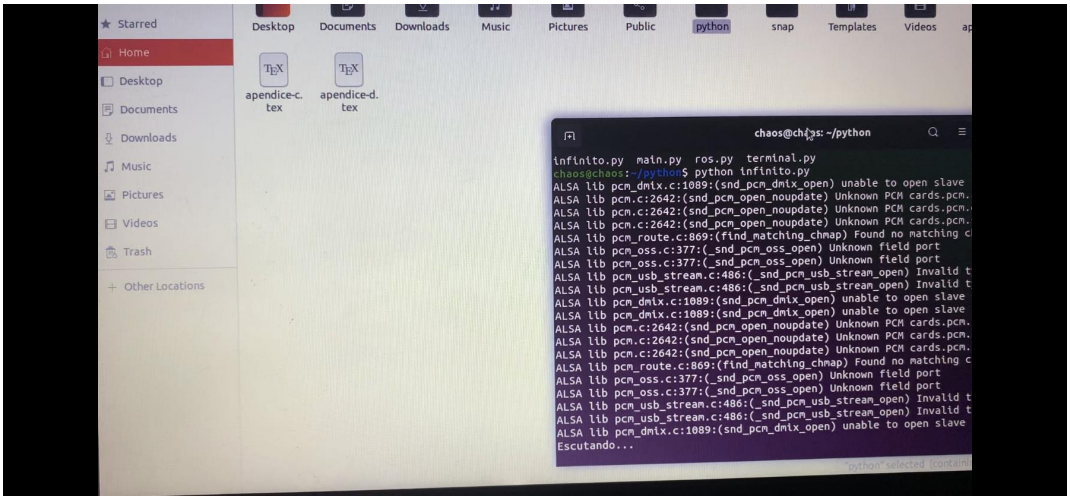
Fonte: Autoria própria

O programa além de apresentar frequentes erros, também apresenta atrasos, travamentos e uma notável hipersensibilidade a ruídos.

4.2 Programa 2

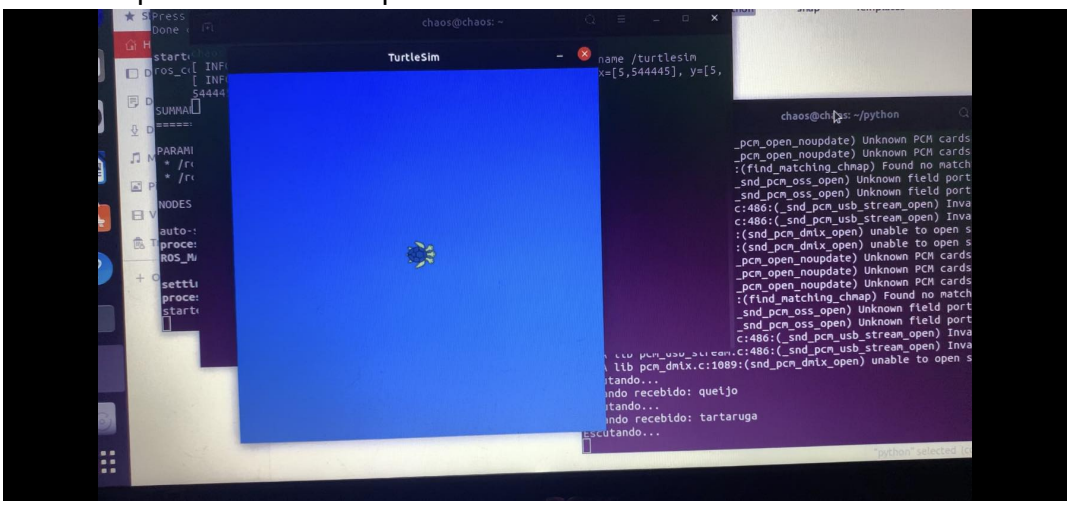
Depois foi testado o programa de comando do ROS, e o resultado foi o seguinte.

Figura 9 – Antes do comando para iniciar o *roscore*



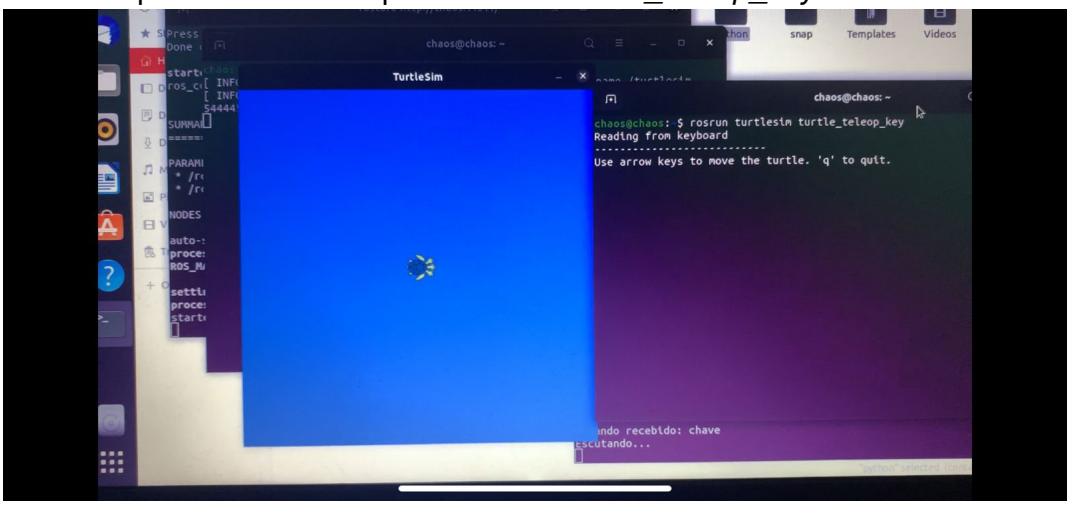
Fonte: Autoria própria

Figura 10 – Após do comando para iniciar o *turtlesim*



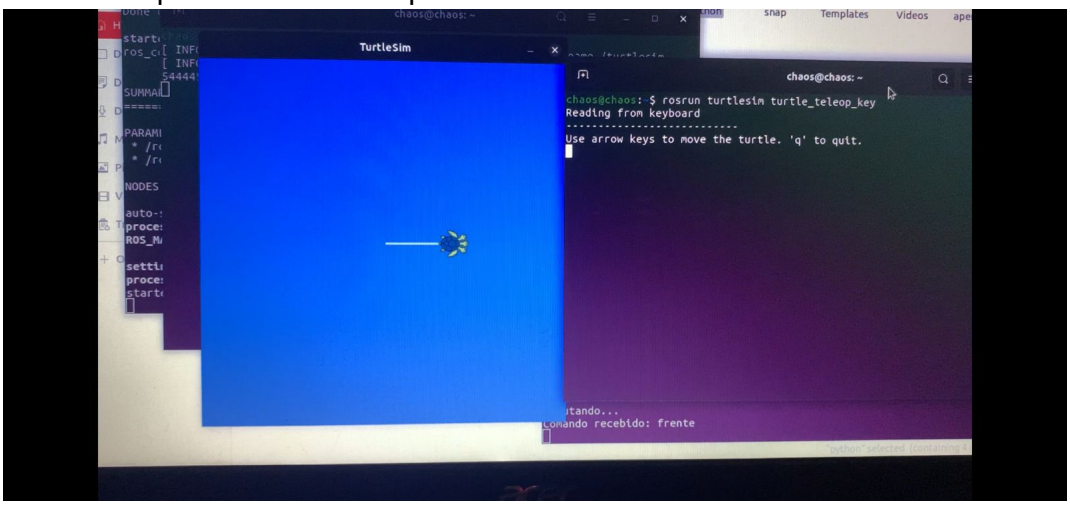
Fonte: Autoria própria

Figura 11 – Após do comando para iniciar o *turtle_teleop_key*



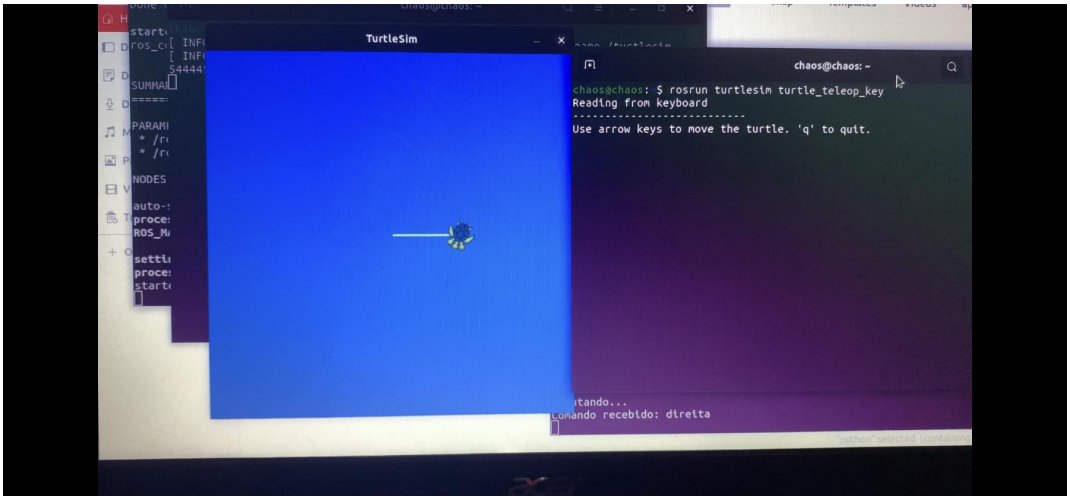
Fonte: Autoria própria

Figura 12 – Depois do comando para frente



Fonte: Autoria própria

Figura 13 – Depois do comando para direita



Fonte: Autoria própria

Embora o resultado desejado tenha sido alcançado, o acionamento do ROS pelo comando resultou na herança de todos os defeitos do programa 1 para o programa 2, tornando-o extremamente ineficiente.

Houve também problema com a palavra ’ROS’, sendo confundida com a palavra ’Ross’ e também ’cross’, por isso foi substituida pelo comando ’queijo’ no programa, já que ele está configurado para reconhecer palavras da lingua portuguesa. O vídeo do teste está disponível em: <https://youtu.be/-I02cx-GhOk>.

O programa foi submetido a um teste no qual foram executados comandos aleatórios, e em 200 tentativas, ele demonstrou responsividade em 43% delas.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho mostrou que ao utilizar o ROS como *framework* de desenvolvimento, os desenvolvedores se beneficiam da arquitetura modular e distribuída do sistema, o que facilita a integração do reconhecimento de voz com outros componentes do robô. O ROS possui uma vasta biblioteca de pacotes e ferramentas que podem ser aproveitados para simplificar o desenvolvimento e acelerar o tempo de implementação.

A linguagem Python oferece uma ampla gama de bibliotecas e *Application Programming Interface (API)* para o processamento de voz e o reconhecimento de fala, como o *SpeechRecognition*, o *pocketsphinx* e o *Google Cloud Speech-to-Text*. Essas ferramentas permitem que os desenvolvedores capturem, processem e interpretem a fala em tempo real, convertendo-a em comandos ou informações úteis para o robô.

O reconhecimento de voz ainda apresenta desafios, como a variação na qualidade da fala, a detecção de diferentes idiomas e sotaques, e a necessidade de lidar com ruídos e ambientes acústicos complexos. Esses desafios requerem técnicas avançadas de processamento de sinais e aprendizado de máquina para melhorar a precisão e a robustez do reconhecimento de voz.

Em resumo, o reconhecimento de voz para ROS baseado na linguagem *Python* oferece uma solução promissora e acessível para a interação homem-máquina em sistemas robóticos. Com a combinação do poder do ROS, a flexibilidade do *Python* e as técnicas avançadas de processamento de voz, é possível criar robôs inteligentes capazes de compreender e responder a comandos verbais, abrindo novas possibilidades para a robóticas.

Como trabalho futuro, seria interessante explorar maneiras de reduzir o atraso, inicializar o programa junto com o sistema operacional e também integrá-lo com uma inteligência artificial.

REFERÊNCIAS

ALECRIM, E. **Comunidade Linux é a mais rápida em corrigir bugs reportados, aponta Google**. 2022. Disponível em: <https://tecnoblog.net/noticias/2022/02/21/comunidade-linux-e-a-mais-rapida-em-corrigir-bugs-reportados-aponta-google/>. Acesso em: 20 jun 2023.

ASSISCITY. **Por que Linux é considerado o sistema operacional mais seguro?** 2022. Disponível em: <https://www.assiscity.com/comportamento/por-que-o-linux-e-considerado-o-sistema-operacional-mais-seguro--119081.html>. Acesso em: 20 jun 2023.

DISTROWATCH. **Page Hit Ranking**. 2023. Disponível em: <https://distrowatch.com/>. Acesso em: 05 maio 2023.

GORDILLO, C. D. A. **Reconhecimento de Voz Contínua Combinando os Atributos MFCC e PNCC com Métodos de Robustez SS, WD, MAP e FRN.** 2013. 101 f. Dissertação (Mestrado em Engenharia Elétrica) — Centro Técnico Científico, Programa de Pós–graduação em Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 3 2013.

HONIG, F. e. a. Revising perceptual linear prediction (plp). Universitat Erlangen-Nurnberg, 2005.

HUGGINS-DAINES, D. **PocketSphinx 5.0.1**. 2023. Disponível em: <https://pypi.org/project/pocketsphinx/>. Acesso em: 06 jun 2023.

INSTITUTE, P. **Python® – the language of today and tomorrow**. 2023. Disponível em: <https://pythoninstitute.org/about-python>. Acesso em: 19 jun 2023.

LOGOS, C. **Como Criar O Seu Próprio Assistente Virtual com Python em 2021**. 2021. Disponível em: <https://www.youtube.com/watch?v=HJuhrBm8AC8&>. Acesso em: 05 maio 2023.

MORENO, P. J. **Speech recognition in noisy environments**. Tese (PhD thesis) — Carnegie Mellon University, 1996.

MUNDI, O. **Hoje na História: 1991 - Sistema operacional de código aberto Linux é criado na Finlândia**. 2021. Disponível em: <https://operamundi.uol.com.br/hoje-na-historia/1260/hoje-na-historia-1991-sistema-operacional-de-codigo-aberto-linux-e-criado-na-finlandia>. Acesso em: 22 jun 2023.

OLIVEIRA LUIZ EDUARDO SOARES DE; MORITA, M. E. Introdução aos modelos escondidos de markov. Pontifícia Universidade Católica do Paraná, 2010.

PERICO ALISSON; SHINOHARA, C. S. S. C. D. **SISTEMA DE RECONHECIMENTO DE VOZ PARA AUTOMATIZAÇÃO DE UMA PLATAFORMA ELEVATÓRIA.** 2014. 96 f. Curitiba: [s.n.], 2014.

PLANNERER, B. **An Introduction to Speech Recognition**. 2005. Disponível em: <https://www.scribd.com/document/52115407/

An-Introduction-to-Speech-Recognition-B-Plannere#>. Acesso em: 13 jun. 2023.

ROS. **Getting Started**. 2023. Disponível em: <https://www.ros.org/blog/getting-started/>. Acesso em: 20 jun 2023.

ROS. **ROS – Robot Operating System**. 2023. Disponível em: <https://www.ros.org/>. Acesso em: 05 jun 2023.

SANTANA, B. **Python® – the language of today and tomorrow**. 2023. Disponível em: <https://www.hostinger.com.br/tutoriais/python-o-que-e>. Acesso em: 20 jun 2023.

SILVA, C. P. A. d. **Um Software de Reconhecimento de Voz para Português Brasileiro.** 2010. 74 f. Dissertação (Mestrado em Engenharia Elétrica) — Instituto de Tecnologia, Programa de Pós–graduação em Engenharia Elétrica: Telecomunicações, Universidade Federal do Pará, Belém, 3 2010.

SOUZA, D. M. d. **Modelos ocultos de Markov: uma Abordagem em Controle de Processos.** 2013. 72 f. Mestrado em Engenharia Civil — Departamento de Estatística; Universidade Federal de Juiz de Fora, Juiz de Fora, 2013.

SPINUPWP. **What Does Ubuntu "LTS" Mean?** 2023. Disponível em: <https://spinupwp.com/doc/what-does-lts-mean-ubuntu/>. Acesso em: 05 maio 2023.

STATCOUNTER. **Desktop Operating System Market Share Worldwide - June 2023**. 2023. Disponível em: <https://gs.statcounter.com/os-market-share/desktop/worldwide>. Acesso em: 17 jul 2023.

TEXEIRA, S. **Hoje na História: 1991 - Sistema operacional de código aberto Linux é criado na Finlândia**. 2022. Disponível em: <https://www.cpt.com.br/artigos/por-que-escolher-o-linux-quais-sao-suas-vantagens>. Acesso em: 22 jun 2023.

UBUNTU. **Ubuntu for desktops**. 2023. Disponível em: <https://ubuntu.com/desktop>. Acesso em: 25 mar 2023.

VARGAS IOHAN G.; TAVARES, D. M. Estudo de caso usando o framework robot operating system. 2013. Disponível em: <https://www.enacomp.com.br/biblioteca-depublicacoes/get-file.php?id=16384>. Acesso em: 15 jun 2023.

ZHANG, A. e. a. **SpeechRecognition**. 2014. Disponível em: <https://pypi.org/project/SpeechRecognition/>. Acesso em: 13 maio 2023.

ZHANG MINGCHI; CHEN, X. L. W. Introdução aos modelos escondidos de markov. 2021. Disponível em: <https://www.mdpi.com/2076-3417/11/7/3138>. Acesso em: 22 jun. 2023.

APÊNDICE A – CÓDIGOS-FONTES UTILIZADOS PARA RECONHECIMENTO DE VOZ

Código-fonte 1 – Reconhecimento de voz em *(*Python)

```
1  import speech_recognition as sr
2
3  def recognise():
4    r = sr.Recognizer()
5
6    with sr.Microphone() as source:
7      r.adjust_for_ambient_noise(source)
8
9      while True:
10       audio = r.listen(source)
11
12       entry=r.recognize_google(audio, language="pt")
13       return"voce disse {}".format(entry)
14
15
16 speech = recognise()
17 print(speech)
```

Código-fonte 2 – Reconhecimento de voz para ROS

```
import speech_recognition as sr
import pyautogui

def recognise():
    r = sr.Recognizer()

    with sr.Microphone() as source:
        r.adjust_for_ambient_noise(source)

        while True:
            print("Escutando...")

            try:
                audio = r.listen(source)
                entry = r.recognize_google(audio, language=
                   "pt")
                print("Comando recebido: {}".format(entry))

                if 'terminal' in entry:
                    pyautogui.hotkey("ctrl", "alt", "t")
                elif 'queijo' in entry:
                    pyautogui.PAUSE = 2
                    pyautogui.hotkey("ctrl", "alt", "t")
                    pyautogui.write("roscore")
                    pyautogui.press("enter")
                elif 'tartaruga' in entry:
                    pyautogui.PAUSE = 2
                    pyautogui.hotkey("ctrl", "alt", "t")
                    pyautogui.write("rosrun turtlesim
                       turtlesim_node")
                    pyautogui.press("enter")
```

```
30                  elif 'chave' in entry:
31                      pyautogui.PAUSE = 2
32                      pyautogui.hotkey("ctrl", "alt", "t")
33                      pyautogui.write("rosrun turtlesim
                            turtle_teleop_key")
34                      pyautogui.press("enter")
35                  elif 'frente' in entry:
36                      pyautogui.PAUSE = 2
37                      pyautogui.press("up")
38                  elif 'atras' in entry:
39                      pyautogui.PAUSE = 2
40                      pyautogui.press("down")
41                  elif 'direita' in entry:
42                      pyautogui.PAUSE = 2
43                      pyautogui.press("right")
44                  elif 'esquerda' in entry:
45                      pyautogui.PAUSE = 2
46                      pyautogui.press("left")
47                  elif 'fechar' in entry:
48                      pyautogui.PAUSE = 2
49                      pyautogui.press("q")
50
51              except sr.UnknownValueError:
52                  print("Nao foi possivel reconhecer o  udio
                        ")
53              except sr.RequestError:
54                  print("Nao foi possivel se conectar ao
                        servi o de reconhecimento de voz")
55
56  try:
57      recognise()
58  except KeyboardInterrupt:
```

```
59      print("Programa interrompido")
```