Universidade do Minho
Escola de Engenharia

Fábio André da Costa Barbosa

Aplicação de Língua Gestual Portuguesa num robô virtual com suporte em ROS

dezembro de 2019

Universidade do Minho
Escola de Engenharia

Fábio André da Costa Barbosa

Aplicação de Língua Gestual Portuguesa num robô virtual com suporte em ROS

Dissertação de Mestrado

Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação do

Professor Doutor Agostinho Gil Teixeira Lopes

dezembro de 2019

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Aplicação de Língua Gestual Portuguesa num robô virtual com suporte em ROS

No mundo atual, verifica-se cada vez mais, o uso de novas tecnologias para tarefas do dia-a-dia ou então para resolução de problemas. Na última década tem-se verificado uma exploração exponencial em diversas áreas, incluindo robótica, que tem como um dos seus focos evoluir e dinamizar a interação entre o Homem e a Máquina, através da reprodução de sons para assim comunicar por uma Língua.

Os indivíduos surdos e/ou mudos possuem deficiência na audição e/ou fala, tendo as- sim uma desvantagem no mundo social já que a comunicação por voz é predominante no quotidiano, havendo poucos que conseguem entendê-los. Dado isso, o objetivo do projeto é apoiar essas pessoas desenvolvendo uma plataforma robótica capaz de habilitar os robôs a conversar em Língua Gestual. O objetivo é criar uma infraestrutura capaz de traduzir Língua Portuguesa para Língua Gestual Portuguesa, desenvolvendo um robô virtual.

No projeto usou-se um midleware open-source que é focado no desenvolvimento de robôs, designado em português, de sistema operativo de robôs. Este tem vindo a ser cada vez mais habitual entre quem desenvolve robôs, por fornecer bibliotecas e ferramentas prontas a serem usadas. Com este middleware consegue-se integrar bem com vários simu- ladores, facilitando a criação de um robô humanoide virtual. Neste projeto desenvolveu-se uma prova de conceito, mas, pensando na escalabilidade do sistema, foi desenvolvida uma base de dados com MongoDB, para o sistema guardar as propriedades de cada gesto e uma interface gráfica para ser mais fácil de usar e visualizar o robô a traduzir.

Finalmente, o sistema proposto foi avaliado por um grupo de pessoas bilingues, que permitiu detetar algumas limitações do sistema e validar a aplicabilidade deste em edifícios públicos. Face aos testes, este projeto provou obter grande sucesso nos objetivos propostos.

Palavras Chave: IHM (Interação Homem-Máquina), Língua Gestual, MongoDB, ROS, Tradutor linguísticos.

ABSTRACT

Portuguese Sign Language Application in a virtual robot with ROS support

Our world is increasingly using new technologies for day-to-day tasks or for problem solving. In the last decade there has been an exponential exploration in several areas, including robotics, which has as one of its focuses evolve and dynamize the interaction between Man and Machine, through the reproduction of sounds to communicate in a language.

Deaf and/or mute individuals are deficient in hearing and/or speech, thus having a disadvantage in the social world since voice communication is predominant in everyday life, with few who can understand them. Given that, the goal of the project is to support these people by developing a robotic platform capable of enabling robots to talk in Sign Language. The goal is to create an infrastructure capable of translating Portuguese to Portuguese Sign Language, developing a virtual robot.

An open-source middleware was used for the project, which focuses mainly on robot development, robotic operating system, that has become common among those who develop robots, by providing libraries and tools ready to be used. This middleware can integrate well with several simulators, facilitating the creation of a virtual humanoid robot. In this project was developed a proof of concept but, thinking about the scalability of the system, a database was developed with MongoDB, for the system to save the properties of each gesture and a graphical interface to be easier to use and to visualize the robot translating.

Finally, the proposed system was evaluated by a group of bilingual people, which al- lowed to detect some limitations of the system and to validate its applicability in public buildings. In the face of the tests, this project proved to achieve great success in the proposed objectives.

Keywords: HMI (human-machine interaction), Language Translator, MongoDB, ROS, Sign Language.

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

ABREVIAÇÕES

ASIMO	*Advanced Step in Innovative Mobility*
ASLAN	*Antwerp's Sign Language Actuating Node*
CEATEC	*Cutting-Edge IT & Electronics Comprehensive Exhibition*
GDL	Graus de Liberdade
IG	Interface Gráfica
JSON	*JavaScript Object Notation*
LG	Língua Gestual
LGP	Língua Gestual Portuguesa
LP	Língua Portuguesa
NA	Não Aplicável
ROS	*Robotic Operating System*
SO	Sistema Operativo

1 INTRODUÇÃO

Neste capítulo será feito um enquadramento do projeto onde serão descritas as áreas às quais este projeto pertence, serão também explicados a motivação e os objetivos. Por fim, será apresentada a estrutura do relatório.

1.1 Enquadramento

O ramo da robótica está, nos dias de hoje, a ser aplicado em várias áreas, apoiando engenheiros, técnicos e profissionais da linha de montagem, assim como também cirurgiões, pessoas com deficiência, entre outros. A robótica procura auxiliar na resolução de problemas. A interligação entre robótica e saúde tem tido especial atenção, por exemplo na utilização de braços robóticos em cirurgias de alto risco. Neste trabalho vai-se recorrer também à área de robótica para resolver problemas na área de saúde, mais especificamente em pessoas com deficiência auditiva e/ou oral, como surdos e/ou mudos.

A Língua Gestual é a principal forma de comunicação das pessoas surdas/mudas, no entanto como não é ensinada às pessoas ouvintes, são poucos os que as compreendem resultando num desentendimento entre ambos os lados que pode ser ajudado com recurso à robótica. Tal como estão a ser feitos tradutores para as diversas línguas orais também se podem fazer para as línguas gestuais, de modo a poderem ser utilizados para melhorar a comunicação entre ouvintes e surdos/mudos.

1.2 Motivação

O interesse deste trabalho é o de ajudar pessoas com deficiência na audição e/ou fala, pois são pessoas que não têm muito apoio na comunidade portuguesa. Como a fala é o método mais comum para comunicação, torna-se complicada a integração destas pessoas na sociedade. Há uma falta de comunicação entre pessoas com deficiência e sem ela, dado que normalmente os ouvintes não aprendem língua gestual. Esta solução tenta, portanto, apoiar na comunicação de ambos, em locais onde não existam interpretes, mas que é na mesma crucial haver alguém para os interpretar, como por exemplo, em locais públicos, como na receção de hospitais, tribunais, estações, centrais de autocarros, entre muito outros.

Além de desenvolver uma solução face ao problema descrito, o projeto pretende aprofundar os

conhecimentos em robótica, em especial abordar infraestruturas que estão a ser populares no mercado, como o caso de *Robotic Operating System* (ROS). Podendo haver a possibilidade de aplicar o projeto resultante num robô verdadeiro e no futuro ser possível inseri-lo no mundo real.

1.3 Objetivos

Pretende-se com este projeto criar um sistema para auxiliar pessoas com necessidades, especificamente pessoas com deficiência na audição e/ou fala, sendo estas surdas e/ou mudas. A ideia base seria a de desenvolver um robô real com braços e dedos, que pudesse comunicar em língua gestual. Como o foco do projeto não é desenvolver *hardware* para implementar um robô físico e não se sabe quais as especificações que teria, iniciou-se o trabalho por uma solução de simulação em ROS que fosse genérica o suficiente para ser aplicada no futuro a qualquer robô.

Para o projeto responder ao problema que lhe é apresentado, três requisito específico tem que ser cumpridos:

- Desenvolver num ambiente de simulação (*Gazebo*) a parte superior de um robô humanoide com braços, mãos e dedos;
- Utilizar uma plataforma robótica *standard* que seja universal (ROS) para movimentar os membros do robô simulado;
- Desenvolver uma aplicação para definir os movimentos para um conjunto de cinco palavras predefinidas.

O objetivo do projeto é promover os robôs a serem bilingues (relativamente à Língua Gestual Portuguesa e Língua Portuguesa), sendo assim capazes de comunicar tanto verbalmente como por gestos. Dado que, já existem no mercado várias ferramentas para reprodução e aquisição da fala, incluindo português, faltando assim, de momento uma ferramenta para fazer o mesmo para gestos.

1.4 Estrutura da dissertação

Esta dissertação encontra-se divida em vários capítulos, somando um total de cinco, onde cada um explora um tópico diferente, explicando o seguinte:

- O capítulo “1. Introdução” descreve o projeto proposto, dando primeiro um enquadra- mento do projeto, depois explica tópicos essenciais para o projeto como a motivação e objetivos.
- O capítulo “2. Revisão Bibliográfica” apresenta alguns exemplos de projetos realiza- dos por outros

desenvolvedores, quer por estudantes universitários e/ou empresas, quer por centros de investigação com interesse nesse tema que implementaram um projeto semelhante ao que irá ser apresentado neste documento. Na segunda parte do capítulo irão ser apresentados os Fundamentos teóricos referentes aos componentes usados para o projeto, sendo esses a Língua Gestual Portuguesa (LGP) e ROS.

- O capítulo "3. Trabalho realizado" apresentará todo o processo de desenvolvimento, descrevendo pormenorizadamente as metodologias. Irá ser descrito o processo de criação do modelo robótico genérico que se assemelha às dimensões e movimentos de um ser humano, o desenvolvimento da componente operacional do projeto, onde o comportamento principal da aplicação corre, a criação e população da base de dados com as propriedades de cada gesto e, por fim, a implementação da Interface Gráfica (IG) para facilitar o uso da aplicação para traduzir uma frase ou palavras.
- O capítulo "4. Análise de resultados" explicará os cenários criados para testar o projeto, como também mostrará os resultados adquiridos de cada cenário de teste realizado.
- O capítulo "5. Conclusões e perspetivas futuras" descreverá as conclusões tiradas após uma análise do capítulo anterior, indicará as limitações do projeto e aspetos a melhorar no futuro.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo será descrito o estudo feito na área do projeto, onde serão apresentados alguns projetos, em que os seus objetivos vão ao encontro com os deste projeto, sendo esses o desenvolvimento de uma plataforma capaz de habilitar um robô a falar LG.

Depois serão apresentados os fundamentos teóricos relativos aos tópicos deste projeto, que são LGP e ROS.

2.1 Estado da Arte

Nesta secção vão ser descritos três robôs com a habilidade de utilizar Língua Gestual (LG). É de notar que estes projetos foram desenvolvidos no estrangeiro e, como tal, foram desenvolvidos para a sua LG nativa. Isto porque cada país tem a sua própria LG.

Dois dos projetos referenciados são robôs completos, ou seja, humanoides que estão a ser desenvolvidos por empresas. O terceiro foi uma investigação de alunos universitários, que desenvolveram um braço capaz de produzir gestos de LG. Vai ser dado maior ênfase aos robôs que já estão a ser aplicados em locais públicos para comunicar com as pessoas no dia-a-dia. Será deixado para o final o braço robótico desenvolvido por alunos.

2.1.1 Toshiba: Aiko Chihira

A Toshiba apresentou em 2014, um robô inovador numa exposição no Japão, chamado *Cutting-Edge IT & Electronics Comprehensive Exhibition* (CEATEC). Este robô foi projetado para ter o máximo de fluidez de movimentos nas mãos e braços de forma a possibilitar o robô a falar Língua Gestual Japonesa. O robô chama-se Aiko Chihira. Ao longo do processo de desenvolvimento da Aiko, houve várias equipas a contribuir, como a própria Toshiba, a universidade de Osaka, os Institutos de Tecnologia de Shibaura e Shonan, como também a empresa aLab Inc [1-3].

O objetivo da empresa com o robô Aiko, era criar um robô que fosse capaz de ser um companheiro para pessoas idosas ou com demência, oferecer tele-consultoria, quer por fala como por LG, como também permitir aos profissionais de saúde ou familiares a possibilidade de monitorizar essas pessoas [4].

2.1.2 HONDA: ASIMO

Em 2000, Honda teve uma visão de desenvolver um dos robôs mais avançados do mundo com várias habilidades sem ajuda ou monitorização humana, como caminhar independentemente, subir degraus, compreender gestos, comandos por voz e até mesmo, imitar outros comportamentos humanos. Em 2002, Honda finalizou o seu projeto e mostrou ao mundo o seu robô, chamado ASIMO, representado na figura abaixo, figura 1. Este nome é um acrónimo com significado de *Advanced Step in Innovative Mobility*. Recentemente, em 2014, ASIMO foi melhorado para conter mais e melhores funcionalidades, como ser capaz de comunicar através de LG. Como o centro de investigação da Honda reside no Japão, implica que ASIMO aprendeu Língua Gestual Japonesa [5-8].

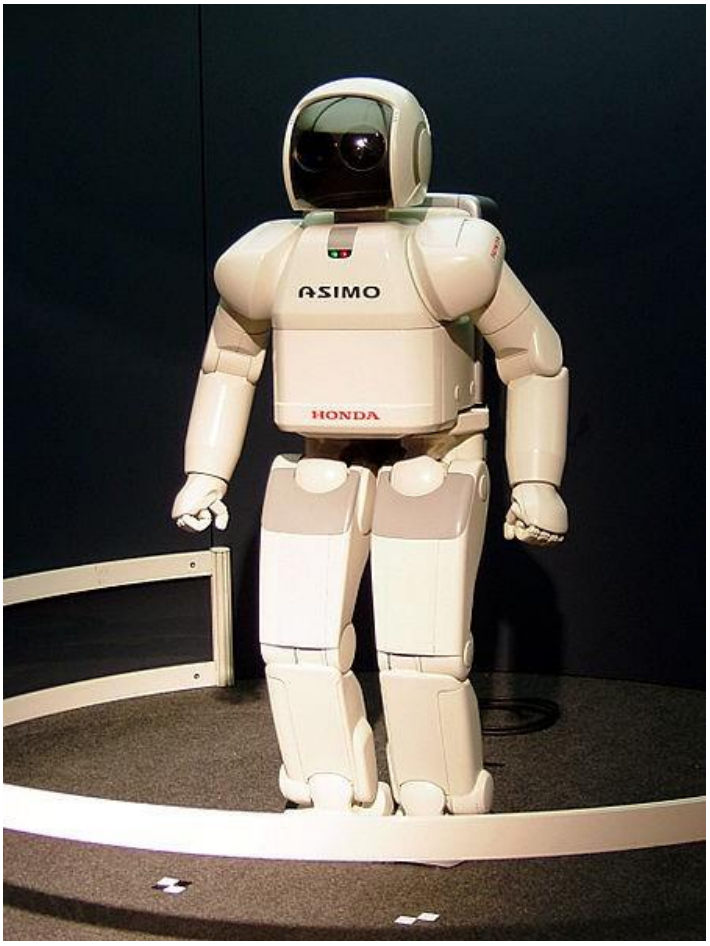


Figura 1: Robô Honda Asimo [9]

2.1.3 Antwerp's Sign Language Actuating Node, ASLAN

O nome que foi dado ao robô é ASLAN, que é um acrônimo para *Antwerp's Sign Language Actuating*

Node. O robô idealizado e desenvolvido por três alunos da universidade Antwerp. A origem do projeto deu-se em 2014 por alunos no seu mestrado quando tomaram conhecimento do défice que existia no seu país de suporte e pessoas para traduzir LG. Os três alunos, Guy Fierens, Stijn Huys e Jasper Slaets idealizaram a criação de um robô capaz de traduzir a tempo real língua nativa oral para LG, tendo em atenção que entre os outros requisitos existentes, o projeto deveria ser barato, portátil e capaz de ser construído a partir de uma impressora 3D [10,11].

O objetivo deste robô não é tirar o lugar aos intérpretes, mas sim apoiá-los, de maneira a que os robôs estejam presentes onde forem precisos, ou seja, quando não haja disponibilidade de um intérprete estar presente. A escolha de desenvolver um robô físico em vez de um simulador ou uma aplicação deve-se ao facto de um objeto físico facilitar a comunicação, já que as pessoas surdas/mudas estão mais habituadas a interagir com outros fisicamente [12,13].

O robô ASLAN pode ouvir em Língua oral e traduzir o discurso para LG em tempo real. Atualmente, a mão funciona de forma independente, por isso só pode executar gestos com uma mão, mas a equipa já está desenvolvendo um outro braço para que os dois braços possam trabalhar em conjunto para poder replicar todos os gestos da LG. Em LG, a comunicação só é bem feita quando se expõe a ideia por gestos com as mãos, postura corporal e expressões faciais, deste modo os pesquisadores estão a procurar soluções para que o robô contenha dois braços e também uma face expressiva [14,15].

2.2 Fundamentos Teóricos

Este projeto é multidisciplinar, por isso neste capítulo serão descritas as áreas que o projeto envolve, fornecendo ao leitor todo o conhecimento que este necessita para entender os passos e razões que irão ser discutidos no próximo capítulo "3. Trabalho realizado". Os tópicos principais que se irão discutir serão dois: o primeiro é destinado à área que o projeto pretende ajudar, LGP; e o segundo será para falar da infraestrutura que o projeto usa, ROS.

2.2.1 Língua Gestual Portuguesa

Este tópico fala da história de LG, descrevendo origem dela, de forma a perceber o seu objetivo e quem esta ajuda. Depois vai ser explicado como a Língua funciona, mais especificamente LGP e identificar os aspetos que a distinguem de LP.

História

A Língua Gestual foi criada para estabelecer um meio de comunicação para pessoas que possuíam deficiência na audição e/ou fala. *Charles Michel de L'Épée* é reconhecido por muitos como sendo o "criador" da LG, apesar de já existir LG antes dele, pelos grandes feitos que conseguiu alcançar; nomeadamente o Instituto Nacional de Surdos-Mudos em Paris, sendo a primeira escola para surdos do mundo. Passou a ser atribuído aos surdos o estatuto de humanos por reconhecimento da existência da sua língua e a constatação da dificuldade que um surdo tinha a exprimir-se oralmente [16,17].

Em Portugal o primeiro instituto para surdos-mudos foi criado no ano de 1823, em Lisboa, no Palácio Conde de Mesquitela. No entanto, só anos mais tarde, no ano letivo de 1993/94 é que a LGP é reconhecida como língua materna e natural dos surdos profundos e introduzida como parte integrante das matérias curriculares no Instituto de "surdos-mudos" Jacob Rodrigues Pereira (IJRP), criado em 1922 [18].

Desde 1990 que os surdos são ensinados através do Modelo Bilingue que visa a aprendizagem da sua língua materna, LGP, e de uma segunda língua, a língua da comunidade ouvinte circundante, a LP, primeira na vertente escrita e eventualmente na vertente oral.

Os surdos aprendem LG de forma natural e espontânea sendo mais difícil para eles a aprendizagem das línguas orais. Tal como os ouvintes têm dificuldades em falar com ou- vintes estrangeiros por terem uma língua diferente, os surdos sentem igual dificuldade a falar com surdos estrangeiros, já que todos os países têm a sua própria língua gestual.

Conceitos fundamentais para o projeto

A LGP é um sistema linguístico composto predominantemente por símbolos arbitrários. Possui ainda aspetos contrastivos em que a alteração de uma unidade fonológica altera o sentido da palavra.

Existem três tipos de gestos que são utilizados: os gestos icónicos, que apresentam ele- mentos de semelhança com a realidade; os gestos referenciais, ato de apontar diretamente para o referente ou espaço que o representa, e os gestos arbitrários, que não apresentam uma relação com a realidade.

Em LGP, a postura que envolve as configurações da mão, local de articulação e orientação, é muito importante, pois LG usa gestos estáticos e dinâmicos. Os gestos estáticos são usados para interpretar as letras e os números. Na figura abaixo, figura 2, pode-se encontrar a interpretação do gesto para cada letra do abecedário como também os números no intervalo de um a nove.

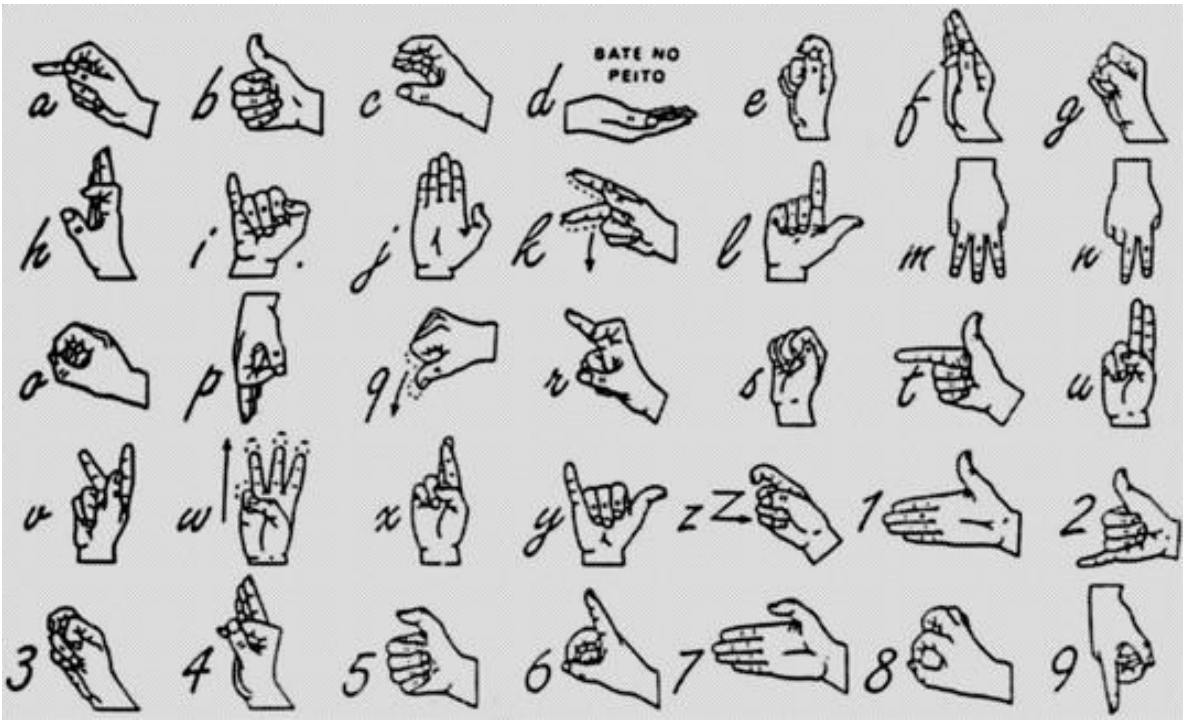


Figura 2: Abecedário e números em LGP [19]

Para o caso de palavras e frases é preciso muitos mais movimentos e preocupação nos gestos a fazer, dado haver gestos com definições específicas para certos contextos. Abaixo são apresentadas duas palavras para mostrar que os movimentos podem ser simples, fazendo um movimento em profundidade com punho cerrado, e outro mais complexo onde já existe mais movimentos com os braços e dedos.

Figura 3: Gesto LGP: "Casa"

Figura 4: Gesto LGP: "Jardim"

Frases em LGP

A construção frásica de LGP é diferente de LP, de maneira que não basta traduzir palavra a palavra para se criar uma frase em LGP. Para além que, em LGP, não é só o uso das mãos e braços que completam uma frase, mas também o uso da expressão facial. Por esta razão, LP é considerada como uma segunda Língua por quem aprendeu LGP antes, sendo tão difícil para eles aprender LP, como aprender outra Língua qualquer.

Existem muitas Línguas Gestuais espalhadas pelo mundo, cada uma com os seus próprios gestos, no entanto, mesmo dentro de Portugal existem os chamados "regionalismos", em que, tal como acontece com LP, cada região pode ter a sua própria variante de uma determinada palavra. Tornando LGP uma língua tão rica como qualquer outra.

Em LGP existem dois gestos distintos para caracterizar os géneros masculino e feminino nos seres humanos, no entanto nos animais, como por exemplo o "gato", existe um gesto que caracteriza o masculino, mas é necessário um conjunto de gestos para caracterizar o feminino, por exemplo no caso de "gata" os gestos utilizados são: "mulher" + "gato".

Neste projeto foi usada a frase "Bom dia, em que posso ajudar?", em LP. Ao traduzir para LGP a frase ficou repartida pela utilização dos gestos "Bom dia" + "eu" + "ajudar" + "posso", nesta específica ordem, pois só assim, com esta sequência, a construção frásica em LGP está correta. O grande destaque é que,

enquanto em LP a ordem palavras segue sujeito -> verbo -> objeto, em esta ordem depende do contexto da frase, mas regra geral a ordem segue-se por objeto -> sujeito -> verbo. O que acontece de maneira geral é que os verbos não são conjugados, aparecendo no tempo infinitivo impessoal e por isso, quando se quer, por exemplo, indicar uma ação continua, o gesto que indica o verbo é repetido várias vezes [20,21].

2.2.2 Sistema Operativo de Robótica (ROS)

Esta secção foca-se na descrição do Sistema Operativo de Robótica (ROS). Pretende-se explicar quais os objetivos e problemas que levaram a criar a infraestrutura, depois vão ser abordados tópicos mais técnicos que pretendem explicar o fluxo de operação e conceitos essenciais.

História e Âmbito

Nos últimos anos, a área de robótica teve um crescimento e uma ramificação exponencial. No mundo atual pode-se verificar que ela se integra cada vez mais nas outras áreas, tornando-se numa área de desenvolvimento de multiplataformas e multidisciplinar. Exemplos disto são a existência de veículos autónomos, quer por terra, tanto em pavimento liso como terrenos acidentados [22], quer náuticos [23] ou aéreos [24]; robôs humanoides ou que apresentam apenas alguns dos membros do ser humano a trabalhar em ambientes industriais [25,26] ou hospitalares [27] e até mesmo em ambiente agrícola [28].

Devido à grande diversidade de robôs e de áreas onde ela se aplica, surgiu uma necessidade de automatizar certos passos, um deles seria a configuração de um robô físico no *software*. A ideia seria criar uma ferramenta que pudesse uniformizar e abstrair esse processo, assim os desenvolvedores teriam mais tempo disponível no desenvolvimento do *software*. Em 2007 a infraestrutura foi oficializada pela *Willow Garage* e atualmente o responsável pela manutenção é a empresa Open *Source Robotics Foudation* [29].

ROS é uma infraestrutura que opera por cima de um sistema operativo, como o Windows ou Linux, tendo o segundo mais popularidade e maior manutenção. Uma boa descrição sucinta é a dada pela própria wiki do ROS, que é a seguinte: "ROS é um sistema meta-operacional de código aberto para o seu robô. Ele fornece os serviços que você esperaria de um sistema operacional, incluindo abstração de *hardware*, controle de dispositivo de baixo nível, implementação de funcionalidades de uso comum, passagem de mensagens entre processos e gerenciamento de pacotes. Ele também fornece ferramentas e bibliotecas para obter, construir, escrever e executar código em vários computadores." [30, 31]. Devido à facilidade de usar e automatizar, ROS tornou-se uma infraestrutura *standard* e flexível para os desenvolvedores, para quando querem escrever

software destinado a robôs [32].

Prós e Contras do ROS

Existe várias infraestruturas como o ROS, tais como Player, YARP, Orocos, MRPT, entre outros. O que torna ROS especial em comparação com as outras é que possuí características como:

- Promoção de implementação modelar, que promove os seus desenvolvedores a implementar código modular, onde cada módulo foca-se numa só operação do sistema. Desta forma, o sistema é mais tolerável a falhas, porque caso um dos processos falhar, o robô não irá interromper abruptamente, pois outras partes estão a funcionar corretamente e podem ser capazes de recuperar o processo que falhou [33];
- Comunidade é bastante ativa, colaborativa e a continua a crescer [34];
- Promoção do uso de licença de livre acesso, como BSD, que permite a reutilização em código de aplicações no meio comercial ou fechado [35,36]. Isto permite que muitos desenvolvedores possam, em conjunto, resolver inúmeros problemas simultaneamente e isso acelera o desenvolvimento de robôs [37];
- Prototipagem fácil de *software*, corrige o problema de ser muito difícil preparar e testar com o robô físico, utilizando simuladores e ficheiros de *debug* como ROS *bags* [38];
- Oferece muitas ferramentas básicas já integradas na plataforma, contendo *software*s que ajudam o desenvolvedor a depurar, visualizar dados/arquitetura do sistema e simular cenários [39].

Pode-se verificar que existem vários pontos que promovem o uso de ROS, mais do que outras infraestruturas rivais. Mesmo assim, ROS também traz algumas desvantagens que são importantes de referenciar, tais como:

- Ser muito novo, tem apenas 10 anos;
- Focar-se em certos aspetos e, como tal, ter que abdicar de outros como a segurança e a escalabilidade dos sistemas;
- Ser multiplataformas, mas ter um favoritismo para um determinado sistema operativo, Ubuntu;
- Ser dependente de versões. ROS já tem algumas versões, mas estas dependem de uma versão específica do SO, isto significa que cada versão do ROS corresponde a uma determinada versão do sistema operativo, por isso fazer um *downgrade* ou um *update* de um geralmente requer o mesmo para o outro;
- Ser incapaz de suportar sistemas de missões críticas, dado que ROS foi criado para apoiar investigadores e estudantes a desenvolverem projetos para provar conceitos [38];

- Não oferecer uma boa ferramenta para modelar robôs. Os desenvolvedores têm que usar principalmente ficheiros xml como URDF e XACRO;
- Ser complexo e exigir algum tempo para aprender;
- Só operar num PC, o que quer dizer que não se aplica a microcontroladores.

Conceitos fundamentais

ROS foi projetado para ser um sistema com mínimo acoplamento, pois o sistema é partido por várias tarefas, sendo cada uma delas da responsabilidade de um Nó específico. Os Nós comunicam entre si enviando mensagens, que são informação estruturada que guarda dados importantes do sistema, em tempo real. As mensagens são publicadas num determinado tópico, que é como um "canal" que deixa transmitir uma informação específica. Um Nó que esteja interessado em receber um determinado tipo de dados, inscreve-se para o respetivo tópico, chamam-se a este tipo de Nós de subscritores. Um Nó que envia informação chama-se editor [40]. Abaixo, figura 5, é apresentado um sistema exemplo, que demonstra a arquitetura do ROS.

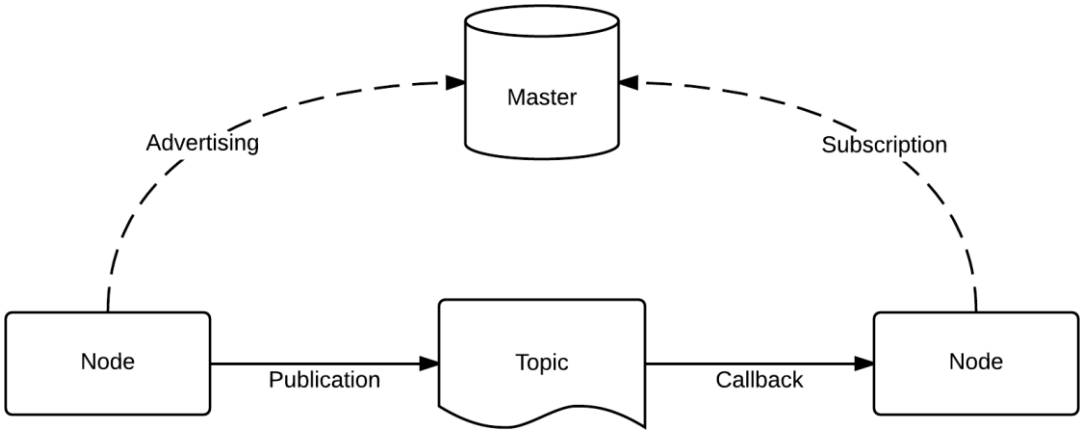


Figura 5: Arquitetura ROS [41]

Sendo ROS um sistema que segue uma topologia centralizada, implica que existe um componente "ROS Master" que controla e gera todos os processos existentes na infraestrutura em execução. Este fornece serviços de nomeação e registo para todos os processos ativos no sistema, também rastreia os subscritores e publicadores de tópicos e serviços [40]. O sistema é composto por vários processos em que, cada um dos Nós é responsável por uma parte do sistema. Os Nós apenas estão interessados em executar a tarefa para que foram programados, abstraindo-se do resto do sistema, por exemplo um Nó é responsável por controlar os motores das rodas do robô, mas o foco do projeto pode ser a deslocação dele sem colidir contra obstáculos

enquanto está a seguir um individuo [42,43].

Como cada Nó só se foca em executar uma pequena tarefa específica, implica que para o sistema correr fluidamente, os Nós têm que ter um meio de comunicação para que possam transmitir informação entre eles. Para isso acontecer, estes aderem a protocolos de comunicação como tópicos, serviços RPC e o Servidor de Parâmetros. Os Tópicos são a interface de comunicação mais usada, este cria canais onde os Nós podem conectar-se para enviar ou receber informação [44].

A informação que os Nós publicam ou recebem de um tópico chama-se de mensagem. Uma mensagem é uma estrutura específica de dados simples guardados em ficheiros de texto [45], é estruturada por um par, que é o nome da variável e o valor dela. A mensagem pode conter tipos desde booleano, inteiros, *string*, vetores ou até mesmo uma estrutura. Em casos especiais, uma mensagem pode incluir um tipo de mensagem especial denominado de cabeçalho, que inclui alguns campos metadados comuns como tempo, data e hora, identificador da *frame* e número de sequência [46].

Quando um sistema tem muitas configurações fica mais fácil guardá-las em ficheiros ou enviar esses dados logo no início da execução. O responsável por isso é o ROS Parameter Server, que é um servidor que funciona como um dicionário compartilhado e variável que é acessível através de API's de rede. Isto ajuda pois, assim os Nós utilizam este servidor para armazenar e recuperar parâmetros em tempo de execução [47,48].

2.2.3 Simulação

Após o desenvolvimento de um programa passa-se para a fase de *debug* e testes. Na robótica esta fase torna-se mais complicado de se fazer, pois isso implica preparar o robô físico e um ambiente de teste que poderá ser grande, caso o robô tenha proporções grandes. Por isso, testar os novos programas implica fazer testes com o próprio robô, mas esses testes vêm com um certo nível de risco, visto que algoritmos incorretos podem levar a danos nos componentes do robô, nos objetos à sua volta ou no pior dos casos, nas pessoas devido a movimentos bruscos ou descontrolados. Uma solução para os problemas mencionados é o uso de simuladores. Estes são capazes de modelar fisicamente o robô e o seu com- portamento da maneira mais próxima possível da realidade [49]. O uso dos simuladores é muito popular por entre os desenvolvedores, pois oferecem muitos benefícios. Alguns desses benefícios estão apresentados abaixo.

- Prototipagem rápida do robô: é possível dispensar o robô físico de fazer *debug* do programa, pois os simuladores já têm a capacidade de recriar um meio ambiente, desta forma, o processo de *debug* da aplicação torna-se muito mais prático, pois o simulador exige menos configurações e preparações que com o robô físico;

- A maioria dos simuladores já contêm ferramentas embutidas para a criação do modelo robótico, tais como os simuladores *Virtual Robot Experimentation Platform, Webots, R-Station, Marilou, 4DV-Sim*;
- Os simuladores podem aumentar o seu leque de funcionalidades através de ferramentas externas, usando *plugins*, dando mais possibilidades ao utilizador para a simulação;
- Algoritmos de física para movimentos e propriedade de ambiente realistas: a maioria dos simuladores usa Open Dynamics Engine, ODE [50], como por exemplo Gazebo, LpzRobots, Marilou e Webots ou então PhysX [51], como por exemplo Microsoft Robotics Studio ou 4DV-Sim;
- Renderização 3D realistas: ferramentas padrões de modelagem 3D ou ferramentas de terceiros podem ser usadas para construir os ambientes [52].

Gazebo

O simulador Gazebo foi criado em 2002 na Universidade do Sul da Califórnia e desde então não parou de evoluir, inserindo novas funcionalidades todos os anos. O conceito do Gazebo foi originado pela necessidade de haver um simulador de alta fiabilidade capaz de simular robôs em ambientes externos sob condições severas. Em 2009, ROS e o robô PR2 foram integrados no Gazebo, tornando-se uma das principais ferramentas de simulação usadas pela comunidade desenvolvedora de ROS. Em 2012, a *Open Source Robotics Foundation*, OSRF, tomou posse do projeto Gazebo. Alguns anos depois, o simulador alcançou um grande marco, sendo o simulador a ser usado no Virtual Robotics Challenge, que é um dos componentes do DARPA Robotics Challenge, em julho de 2013 [53].

Originalmente, o Gazebo foi projetado para avaliar algoritmos para robôs, mas, como a necessidade do momento era simular cenários reais com capacidade de conter múltiplos robôs, este foi desenvolvido e melhorado para corresponder a essas necessidades [54]. Atualmente, no Gazebo, pode-se criar um cenário 3D com robôs, obstáculos e muitos outros objetos. O simulador também usa um mecanismo físico para iluminação, gravidade, inércia e outras variáveis, pode-se avaliar e testar um robô em cenários difíceis ou perigosos sem causar danos no robô físico.

Umas das grandes vantagens do Gazebo é que este incorpora um *plugin* na infraestrutura ROS. Um *plugin* é um pedaço de código que é compilado como uma biblioteca compartilhada e inserido na aplicação *host*, neste caso, a infraestrutura ROS. Este tem acesso direto a todas as funcionalidades do Gazebo, sendo isso ótimo, porque o ROS permite aos desenvolvedores [55]:

- Controlar quase todo o leque de funcionalidades disponíveis pelo Gazebo;
- Pode ser inserido e removido de um sistema em execução;

- As rotinas como são independentes tornam-se fáceis de compartilhar.

URDF

URDF é um ficheiro em formato XML que se denomina por *Unified Robot Description Format*. Como o nome indica, este ficheiro tem o propósito de descrever um robô, ou seja, identificar as caraterísticas de cada componente incluída no robô, isto quer dizer, as dimensões, cores, correlação com outros componentes, cinemática e dinâmica, entre outras caraterísticas. URDF segue uma estrutura em forma de árvore, em que um componente será filho de outro componente e pai de vários outros componentes. Modelar um robô significa, portanto, descrever todos as partes que o compõe num ficheiro URDF [56-58]. Um ficheiro URDF começa sempre com o elemento "robot". Este elemento encapsula todo o modelo do robô que pode ser representado usando URDF. Neste componente é possível dar um nome ao robô, depois disso pode-se inserir vários tipos de elementos como por exemplo:

- O "*link*" descreve um componente rígido do robô. Este elemento descreve propriedades visuais como o tamanho, forma, cor e pode até mesmo, importar uma malha 3D para representar o componente. É possível descrever também propriedades dinâmicas, como matriz inercial e colisão [59].
- O "*joint*" descreve uma articulação do robô. Este elemento descreve a cinemática e dinâmica da articulação e também pode definir os limites do movimento e a velocidade. Os tipos de articulações que existe são "revolute", que gira ao longo do eixo; "*continuous*", que gira em torno de um eixo sem limites; "prismatic", que desliza ao longo do eixo; "*fixed*", que tem todos os graus de liberdade bloqueados; "floating", que permite movimento para todos os 6 graus de liberdade; "planar", que permite o movimento num plano perpendicular ao eixo [60].
- "Transmission" configura os atuadores reais ao *joints* virtuais;
- "Gazebo" descreve as propriedades de simulação, como fricção, gravidade, etc. O interesse deste elemento é o de incluir os parâmetros de simulação dentro do ficheiro URDF;
- "Sensor" descreve um sensor real como um componente do modelo virtual;
- "Model" descreve as propriedades de cinemática e dinâmica do robô.

Há dois elementos que têm maior importância, dado serem os mais usados, esses elementos são o "*joint*" e "*link*". Um modelo tem sempre um *link* base e a modelagem do robô começa por aí. Para construir mais componentes usa-se um *link*, estes podem estar apega- dos uns aos outros com formatos diferentes,

mas implementado nesta forma, implicaria o robô não ter mobilidade dado que os *links* são fixos. Para dar liberdade ao robô, usa-se *joints* entre dois *links*, desta forma o robô adquire mais um grau de liberdade [56-58].

Xacro

Xacro é uma abreviação de "XML Macros" e pode-se dizer que Xacro é o URDF 2.0, ou seja, uma versão melhorada. Xacro foi criado porque a flexibilidade do URDF é reduzida quando trabalhamos com modelos de robôs complexos. Descrever um modelo robótico através de URDF implica descrever o robô num único ficheiro. Caso seja preciso reutilizar um bloco em URDF é obrigatório copiar e colar o bloco pelas várias partes necessárias, estes fatores fazem com que URDF reduza a simplicidade e modularidade do código. Além disso, URDF não suporta variáveis, constantes, expressões matemáticas e declarações condicionais.

Criou-se Xacro para aperfeiçoar as caraterísticas de URDF, melhorando as áreas de simplicidade, reusabilidade, modularidade e programação. As principais vantagens de se usar Xacro é que este suporta:

- Macros, por exemplo:

```
<xacro:macro name="pr2_arm" params="suffix parent reflect">
<pr2_upperarm suffix="${suffix}" reflect="${reflect}" parent="${parent}" />
<pr2_forearm suffix="${suffix}" reflect="${reflect}" parent="elbow_flex_${suffix}" />
</xacro:macro>
. . .
<xacro:pr2_arm suffix="left" reflect="1" parent="torso" />
```

Neste exemplo, podemos ver uma macro a representar um bloco de código XML. Desta forma, este bloco não precisa de ser copiado, mas sim apenas invocado pela macro denominada "pr2_arm" [61].

- Variáveis, constantes, expressões matemáticas, declarações condicionais, e assim por diante, por exemplo:

```
<xacro: nome da propriedade = "pan_link_length" value = "0.4" />
<xacro: nome da propriedade = "pan_link_radius" value = "0.04" />
. . .
<cylinder length = "$ {pan_link_length}" radius = "$ {pan_link_radius+0.02}" />
```

Neste exemplo, pode-se encontrar o uso de contantes e expressões matemáticas. O que irá acontecer é

que o *parser* irá substituir o valor "0,4"onde aparecer o nome “pan_*link*_length” e o valor “0,04” onde aparecer o nome “pan_*link*_radius”. No atributo “radius” pode-se ver uma expressão matemática, sendo que o valor final do raio será 0.06 (0.04+0.02).

2.3 Sumário

No primeiro tópico que foi abordado neste capítulo foram apresentadas as conclusões da análise feitas acerca do estado da arte sobre Língua Gestual a ser aplicada em robôs. Desta análise identificaram-se três exemplos com maior relevo face aos requisitos e área de aplicação dos projetos, que é apoiar a comunicação entre pessoas (utilizadores da língua oral com utilizadores de LG) e robôs com pessoas, traduzindo de uma língua para outra em tempo real. Com esta análise pode-se evidenciar que, claramente, existe algum estudo na área acerca de robôs a reproduzir gestos em LG, mas é importante salientar que isto acontece quando se considera LG globalmente. Quando se específica para projetos orientados a LGP nota-se que há pouca exploração. Mesmo assim, o que se pretende deste projeto é inspirar nos projetos do estrangeiro e conseguir implementar um sistema parecido, mas dedicado para LGP.

No tópico seguinte foram apresentados conceitos fundamentais para a compreensão do projeto. Começou-se por explicar LGP, nesta secção explicou-se a origem da língua gestual e a sua motivação, que era ajudar as pessoas com deficiência na audição e/ou na fala a comunicar umas com as outras. De seguida descreveu-se como é feita a tradução com uns exemplos, no caso de palavras individuais, letras e números existe uma tradução específica, no caso de frases mostrou-se que este se distingue do léxico da Língua Portuguesa, denotando que este tem o seu próprio léxico e as palavras podem ter uma tradução diferente dependendo do contexto.

Depois foi explicada a infraestrutura ROS, esta será a base do projeto, pois a aplicação dependerá desta infraestrutura. Explicou-se a história e o objetivo da criação do ROS, de seguida identificou-se as vantagens e desvantagens do ROS face a outras infraestruturas de robótica. Por fim, foi feito um resumo de como ROS opera, descrevendo a arquitetura do sistema quando este está em funcionamento como também, a arquitetura dos ficheiros.

O último tópico destina-se a explicar a importância dos simuladores quando se trabalha na robótica. Deu-se um breve resumo do simulador que se irá usar neste projeto, falando da sua história e as suas caraterísticas que o levam a ser capaz de ser usado para simular o cenário e o robô deste projeto. Por fim, identificou-se dois tipos de ficheiros para apoiar

o processo de modelagem, explicando um pouco o que é URDF e Xacro, que são duas ferramentas

orientadas à descrição e desenho de um robô.

3 TRABALHO REALIZADO

Neste capítulo irão ser descritas as metodologias que foram aplicadas, descrevendo de forma minuciosa como se implementou o projeto. Durante o desenvolvimento tomaram-se grandes decisões, tais como:

- A escolha dos gestos que iriam ser armazenados na base de dados;
- A escolha da base de dados a usar, dado que esta tem que ser rápida para que a aplicação seja bastante responsiva;
- A modelação de um robô genérico que se assemelha a um ser humano, assim como os movimentos dos seus membros superiores, braços e mãos, isto é, o robô deve ser humanoide e deve ter os mesmos graus de liberdade e limitações que um ser humano;
- A implementação de uma aplicação que possibilite a tradução de LP para LGP em tempo real;
- O desenvolvimento de uma interface que possibilite ao utilizador escolher as palavras a traduzir e ser possível a visualização do robô a demonstrar a tradução da(s) palavra(s) escolhida(s), neste caso, através do simulador Gazebo.

Será primeiro apresentada a base do projeto, que é a especificação do modelo robótico. Depois vão ser explicadas quais foram as palavras escolhidas e a sua razão. Neste mesmo tópico será explicada a base de dados escolhida e como se procedeu para guardar as palavras. No tópico a seguir será descrito o desenvolvimento do core da aplicação, de como ela opera, ou seja, o processo de tradução de uma palavra ou frase e como são mandados os comandos para o simulador, para fazer o modelo mover-se. Por último, será descrita a implementação da interface do projeto.

3.1 Especificação do modelo robótico

Para desenvolver o modelo robótico de forma a ter uma grande semelhança com um ser humano, foram usadas as medidas de uma pessoa real. A tabela 1, apresentada abaixo, apresenta as medições tiradas das proporções do seu corpo.

Tabela 1: Medições tiradas de um adulto jovem

	Comprimento (m)
Altura	1,76
Largura dos ombros	0,46
Pernas	1,00
Tronco	0,55
cabeça e pescoço	0,33
Braço	0,64
Comprimento total dos braços estendidos (mãos, braços e tronco)	1,75
Mão	0,19
Polegar	0,065
Dedo indicador	0,080
Dedo do médio	0,085
Dedo anelar	0,080
Dedo mínimo	0,065

Um dos desafios deste projeto é conseguir criar um modelo robótico que se assemelhe a um ser humano, em que os braços sejam idênticos em aspeto e propriedades, para tal, é preciso compreender a complexidade que estes têm, devido às articulações não serem idênticas umas às outras, por terem propriedades diferentes, isto é, diferentes eixos de rotação e intervalos de ângulo diferentes.

Nos braços existem várias articulações, tendo cada uma as suas próprias caraterísticas, nesta zona do corpo encontram-se 3 tipos: uniplanar, biplanar e multiplanar. Articulações uniplanares rodam em torno de um só eixo, permitindo assim rotação em apenas um plano, exemplos disto são os cotovelos e as pontas dos dedos. Nas articulações biplanares, como o nome indica, rodam em torno de 2 eixos, permitindo assim a rotação em dois planos, exemplos disto são os pulsos e entre a palma da mão e os dedos; geralmente, este tipo de articulações permitem a flexão, extensão e rotação lateral. Por último tem-se as articulações multiplanares que permitem a rotação em três eixos, possibilitando assim o movimento completo, um exemplo disto são os ombros. [62]

A tabela 2 apresenta as conclusões da análise da propriedade de cada articulação exis- tente nos braços e mãos. É preciso ter em conta que os valores apresentados são médias pois, cada indivíduo tem as suas

capacidades, por exemplo não se pode igualar a elasticidade de um body builder com a de um praticante de yoga, pois o praticante de yoga trabalha os músculos para ganhar mais elasticidade, enquanto um body builder, por trabalhar os músculos para cresceram irá influenciar negativamente a sua elasticidade.

Como nos dedos existem 3 articulações muito semelhantes, e para ser possível distingui-las facilmente, usaram-se termos técnicos de biológica para se poder identificar uma articulação específica. A figura abaixo, figura 6, representa a mão com a posição de cada uma das articulações e o seu respetivo nome.

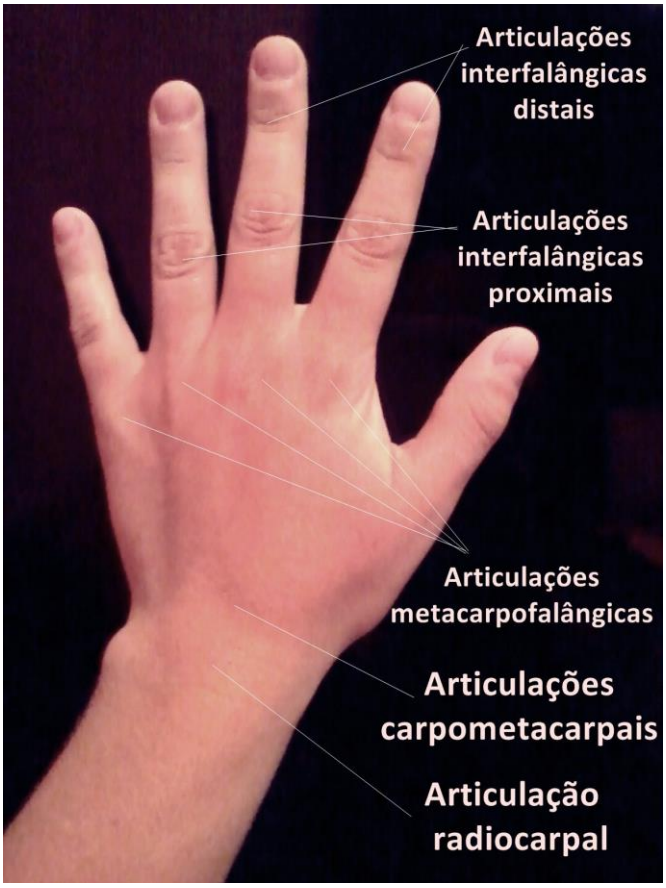


Figura 6: Articulações da mão e punho direitos [63]

Tabela 2: Especificação das articulações dos braços [64-66]

Local	Rotação	Limite Min. (o)	Limite Max. (o)
Ombro	Abdução	0	180
	Flexão	0	180
	Extensão	0	50
	Flexão horizontal	0	130
	Extensão horizontal	0	50

Cotovelo	Flexão	0	160
	Pronação	0	90
	Supinação	0	90
Pulso	Abdução	0	30
	Adução	30	0
	Flexão	0	90
	Extensão	0	70
Dedos	Flexão da interfalângica distal	0	90
	Flexão da interfalângica proximal	0	90
	Flexão da metacarpofalângica	0	90
	Abdução da metacarpofalângica	0	20
	Adução da metacarpofalângica	20	0

A construção do modelo foi feita através do URDF e XACRO. Nenhum programa de *design* de modelação foi usado, por isso o modelo foi criado apenas com objetos geométricos básicos (esfera, cubo, cilindro e paralelepípedo) oferecidos pelo URDF. O modelo foi criado manualmente por ficheiros simples em formato XML.

Dado o foco do robô serem os seus braços, para simplificar o modelo, construiu-se a base por *links* e articulações fixas, ou seja, os pés, pernas, tronco, pescoço e cabeça não têm nenhuma mobilidade, tal como se pode ver na Figura 7, em que as partes do corpo referidas anteriormente não tem nenhuma bola de cor azul para evidenciar locais de rotação.

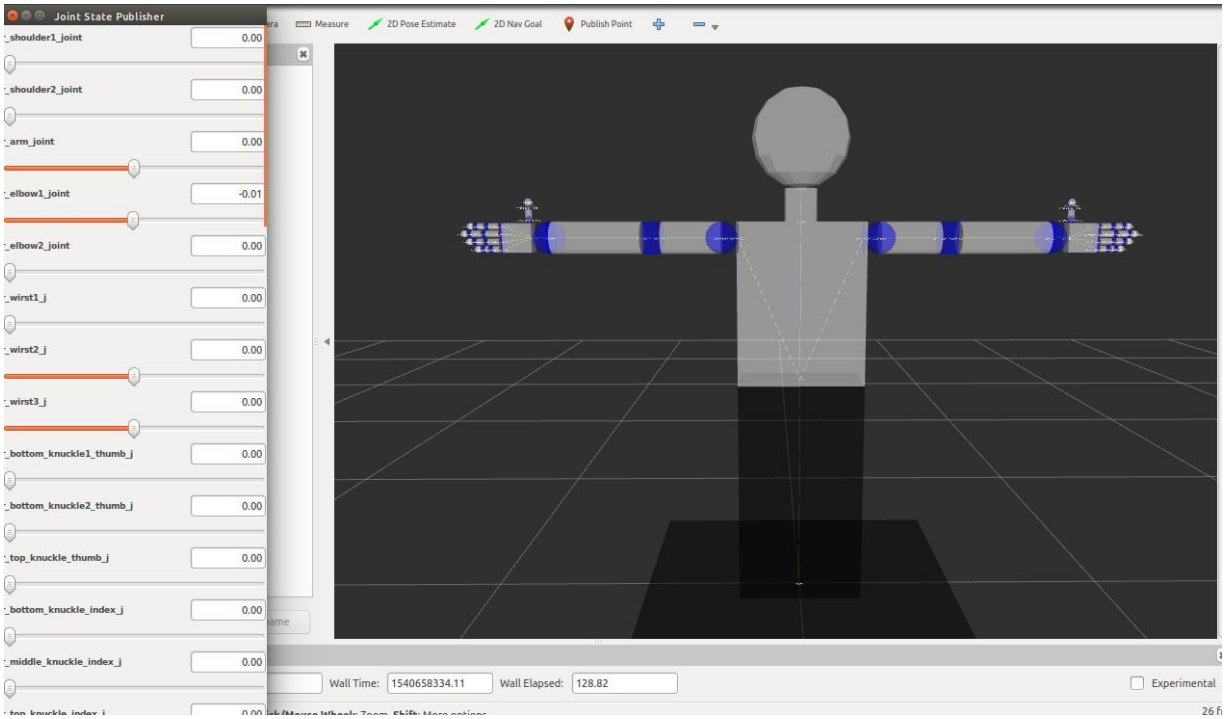


Figura 7: Centro do corpo do robô (corpo transparente)

A partir do início dos braços já se nota uma existência de complexidade nas articulações e na estrutura dos *links*. A articulação, em inglês *joint*, oferece vários tipos de rotação, as quais encontram-se sumariadas abaixo:

- Revolute - a articulação roda por 1 ou até 3 eixos e tem limites superior e inferior.
- *Continuous* - a articulação roda continuamente em torno do eixo definido e não possui limites superior e inferior.
- Prismatic – a articulação desliza ao longo do eixo específico e tem um alcance limitado que é definido pelos limites superior e inferior.
- *Fixed* – Como o seu próprio indica, nesta articulação não é possível a deslocação, pois os seus graus de liberdade estão bloqueados.
- Floating – a articulação move-se por todos os seis graus de liberdade simultaneamente.
- Planar – a articulação irá rodar num movimento plano ao eixo.

Dado existirem vários tipos de articulações, foi feito um estudo para compreender qual é a articulação que mais se adequa a representar os braços humanos o mais fielmente possível. Então, criou-se um braço protótipo para ser possível validar os resultados. Testaram-se vários cenários com o propósito de deslocar os braços em várias direções e verificar qual o tipo de articulação que conseguia habilitar os mesmos graus de

liberdade que os braços de um ser humano. Neste cenário, procurava-se ver os 4 graus de liberdade de um braço.

Os tipos *fixed*, *continuous* e planar foram desde início descartados dado que o tipo *fixed*, não permite rotação; o *continuous*, não tem limites de rotação enquanto um ser humano tem; e o tipo planar habilita apenas rotação em plano, ou seja, em 2 eixos ao mesmo tempo. Fizeram-se testes com os restantes três tipos, float, prismatic e revolute, e verificou-se que nenhum conseguia representar corretamente os movimentos esperados para os testes. Concluiu-se que o prismatic não podia ser usado pois tinha uns problemas evidenciados pelo ROS, que disseram que iriam corrigir no futuro; o float, roda em todos os eixos ao mesmo tempo, o que quer dizer que para os testes simples, que consistiriam apenas em mover num grau de liberdade, não seriam possíveis de controlar, pois este utiliza uma fórmula matemática para determinar os ângulos quando dado apenas um valor como *input*. Devido à sua complexidade, este tipo foi descartado.

Por último, o tipo revolute não passou todos os testes, mas, foi o que obteve melhor desempenho. Foram feitas mais experiências, onde se tentava explorar outras metodologias para ver se era possível melhorar a performance deste tipo de articulação. Chegou-se à conclusão que apesar de o tipo revolute dar a possibilidade de rodar nos 3 eixos, o melhor era especificar uma articulação para cada grau de liberdade e como tal, o braço protótipo, com 4 graus de liberdade, tinha 4 articulações. Desta forma, o protótipo conseguia executar todos os testes com sucesso, mostrando as capacidades, graus de liberdade e limites de um verdadeiro braço humano.

O desenho da mão e dos dedos foi feito seguindo a metodologia das articulações definida para os braços, onde foram criados entre 1 a 3 *joint*s, dependendo da capacidade de rotação da articulação humana. Devido às limitações dos modelos de desenho provenientes pelo URDF, a palma da mão e os dedos foram feitos parecidos à realidade apenas com o uso de paralelepípedos e vários cilindros respetivamente. Como se pode ver em mais detalhe nas figuras abaixo.

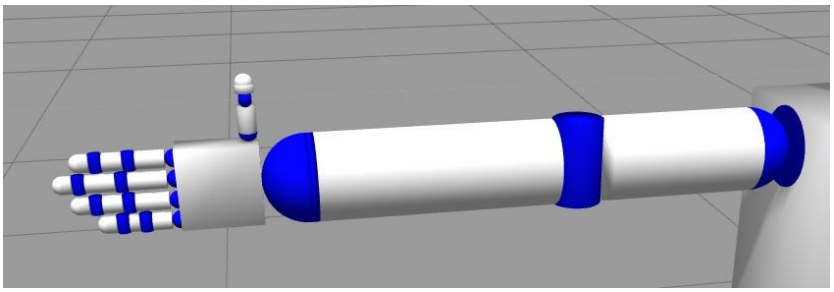


Figura 8: Braço direito do robô

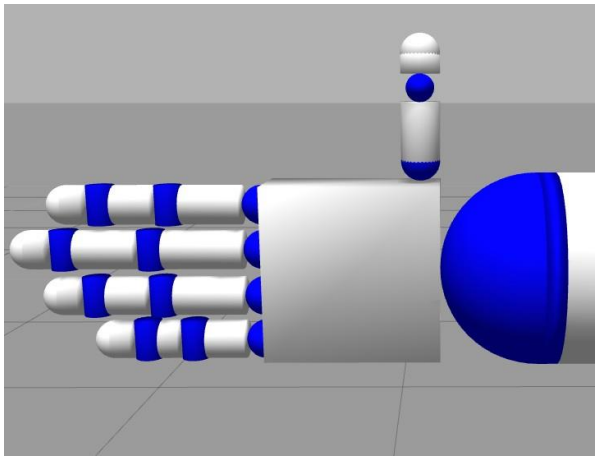


Figura 9: Mão direita do robô

O produto final da implementação do robô resulta na imagem abaixo, figura 10.

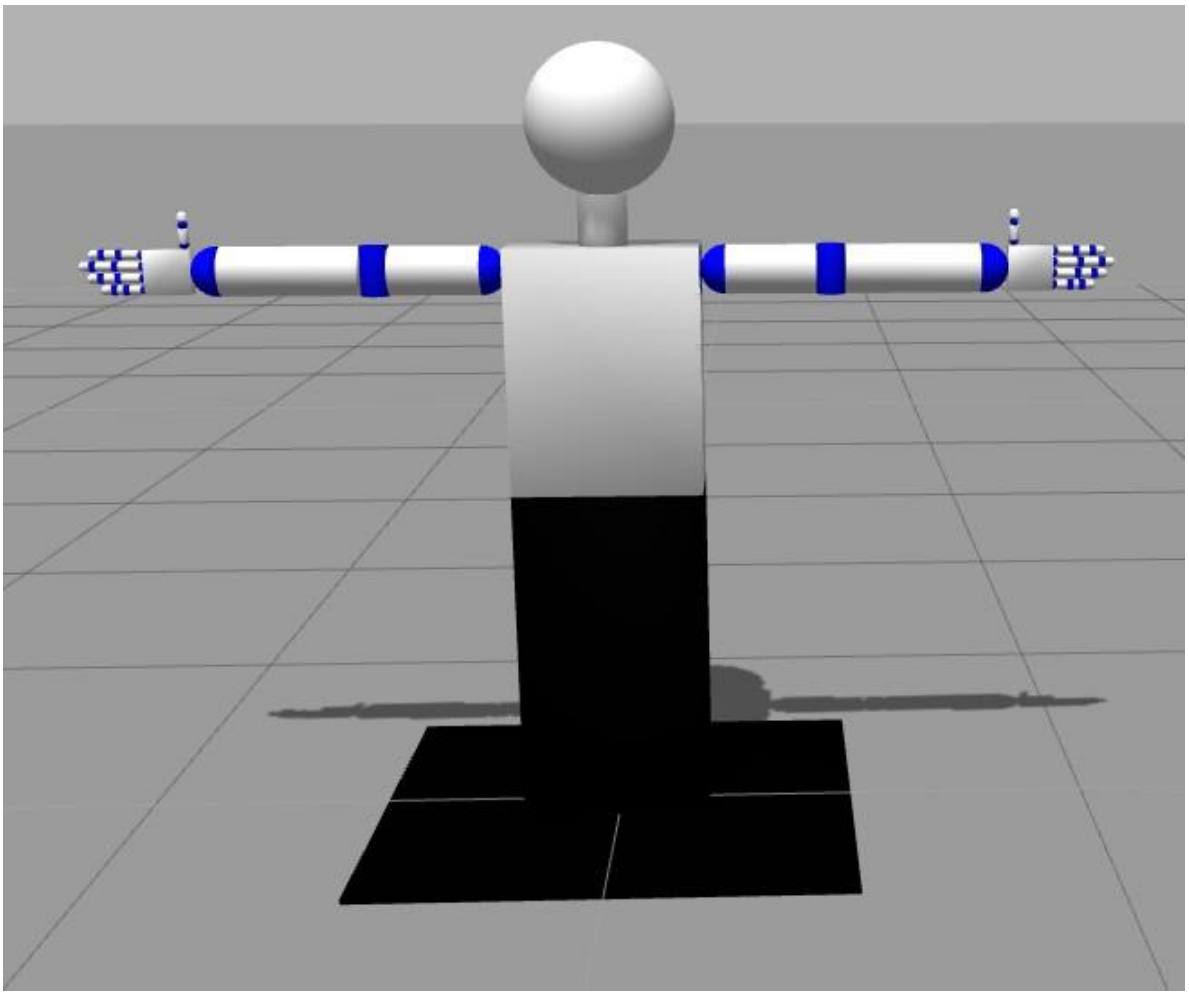


Figura 10: Corpo do robô

No anexo A – Estrutura do modelo robótico tem-se o esqueleto do robô, que descreve em modo

árvore todos os *link* e *joint*s que o compõem. Com este tipo de estrutura pode-se facilmente entender como é que os elementos se conectam, percebendo os elementos pais e filhos de cada um. Através do diagrama é possível compreender que a filosofia da criação de um robô suportado por URDF/XACRO é feita através de uma estrutura em árvore de elementos onde, na raiz da árvore, está geralmente o elemento chamado de "base_link", do qual todos os outros elementos dependem.

Outra implementação que se fez ao modelo é que inicialmente o robô iria desprezar o peso de todos os membros, portanto o robô tendo no total 1 kg, pois trata-se de um robô genérico a ser simulado num ambiente virtual mas, quando o modelo foi posto a correr no simulador executando gestos, foi verificado que o robô tinha tendência a cair, pois o simulador tem pré-definido para o ambiente uma variável para representar a gravidade. Como tal, definiu-se peso na base, cerca de 50 kg, para que, desta forma, o robô não cai.

3.2 Implementação da base de dados

Para a escolha da base de dados tiveram-se em conta vários fatores, que correspondem aos requisitos que a base de dados tem que conseguir cumprir para o projeto. Esses requisitos são:

- Ser rápida, pois espera-se que os resultados do acesso à Base de Dados tenham uma resposta de rápida;
- Não conter uma estrutura rígida, pois há gestos que se fazem com apenas uma posição e outros com duas ou até mesmo com mais de duas posições;
- Ser simples, pois o ideal seria um pedido à base dados em que retorna todos os dados para representar o gesto em LGP.

Com este projeto pretende-se uma prova de conceito e isso implica que a base de dados deve abranger os gestos essenciais para a sua validação, portanto os gestos precisam de ser únicos e com graus de dificuldade diferentes. Para o nível fácil, espera-se apenas um movimento com o(s) braço(s) e gestos estáticos; para o nível intermédio, consideram-se gestos dinâmicos (1 ou 2 movimentos) e que implicariam o uso do(s) braço(s) e mão(s); por último, o nível difícil corresponde a gestos dinâmicos que requeiram o ambas as mãos e deve ser composto por 4 ou mais movimentos.

Outro requisito seria que, ao juntar as palavras, estas pudessem formar uma frase de modo a ser possível provar o conceito de tradução de frase de LGP para LP. Face a estes requisitos, definiram-se 5 gestos e mais um extra, sendo este a posição de repouso dado que, após cada gesto, pretende-se que o robô volte à

sua posição de descanso, que serão os braços estendidos ao longo do tronco.

Na tabela apresentada abaixo, tabela 3, estão identificas todas as palavras que foram escolhidas para serem armazenadas na base de dados.

Tabela 3: Biblioteca de gestos

Letra “b”	Eu
Bom dia	Poder
Ajudar	Pose em repouso

Com as palavras identificadas na Tabela 3, é possível criar uma frase:

“Bom dia, em que posso ajudar?”

As palavras identificadas a Tabela 3 estão descritas no anexo 2 – Especificação dos gestos. Nesse anexo pode-se observar como os gestos são feitos e concluir, com base nisso, que o gesto da letra “b” é estático e usa apenas uma mão, o que significa que este gesto é, dentro dos escolhidos, o mais fácil de executar. Para o nível intermédio, tinham-se os gestos “bom dia”, “eu”, “ajudar” e “poder”. Para o maior nível de dificuldade tinha-se a tradução da frase, “Bom dia, em que posso ajudar?”. Os gestos como a pose de repouso e a letra “b” usam apenas um bloco de posições, pois são gestos fixos. As palavras “eu”, “poder” e “ajudar” são compostas por duas posições ao realizar o gesto. O gesto “bom dia” usa três posições, e tem maior complexidade, pois o gesto requer que se faça uma pausa entre o gesto “bom” e o “dia”.

Após a definição das palavras que a base de dados irá conter, foram analisadas as melhores tecnologias de base de dados disponíveis para se implementar no projeto. Para este caso, verifica-se que uma base de dados não relacional se aplica melhor, pois é mais simples e rápida a responder, dado que uma pesquisa por chave-valor é mais rápida que uma pesquisa por uma *query*, numa base de dados relacional. O fato de não conter uma estrutura rígida faz com que seja possível a criação de dados onde uma palavra pode conter uma pose ou mais. Como o modelo robótico contém 46 articulações, a base de dados irá ter que guardar o ângulo para cada articulação, isso implica que para uma dada palavra a base de dados contém no mínimo a descrição de 46 ângulos até 46xN ângulos, sendo N o número de poses que o gesto exige fazer. Dado que o interesse é guardar os ângulos específicos de cada palavra, isto faz que com que cada uma delas seja independente das outras, devido a este pormenor pode-se verificar claramente que uma base de dados que implementa o modelo relacional deixa de ter efeito para o que se pretende para este projeto, filtrando as opções para apenas as bases de dados não relacionais.

De entre as opções existentes foi escolhido o MongoDB, por esta base de dados não relacional ser a mais popular e já vir integrada com o ROS *framework*. O package é denominado por mongodb_store [67].

Neste projeto espera-se que a base de dados esteja populada com palavras e gestos, mas para tal, foi preciso criar um pequeno programa que criasse e inserisse gestos. O primeiro passo foi a aquisição dos ângulos para cada articulação na execução de um determinado gesto. O método que se usou foi o programa Rviz, que vem integrado com o ROS. Com este programa importava-se o modelo robótico e rodavam-se os ângulos de cada articulação até este refletir o gesto como um ser humano faria. O programa Rviz tem um *plugin* que consegue detetar as articulações habilitadas para rotação e os seus respetivos limites, desta forma só é preciso rodar nos intervalos disponíveis, pois estes já representam os limites que a articulação teria no braço de um ser humano. As Figuras 11 e 12, apresentadas abaixo, demonstram como os passos eram feitos, onde se o programa Rviz aberto com um protótipo inicial dos braços. No canto inferior esquerdo encontra-se uma janela chamada "*Joint* State Publisher" é nesta janela que é rodar manualmente as articulações. Nestes dois exemplos consegue-se ver o ombro esquerdo do robô a rodar desde o ângulo máximo até ao mínimo.

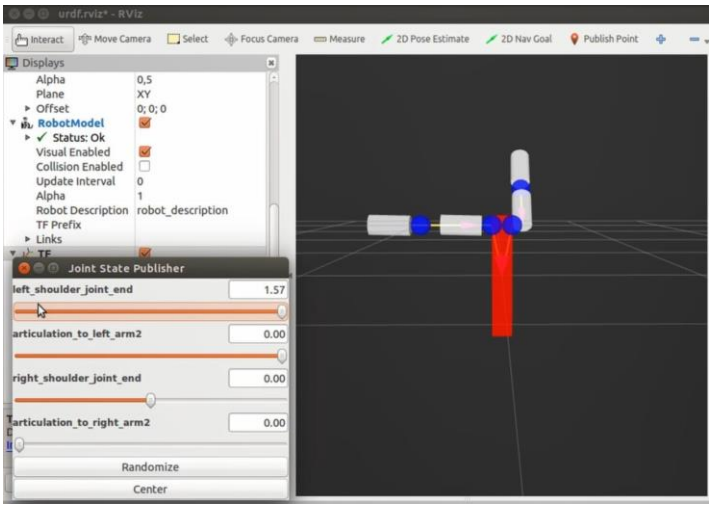


Figura 11: Protótipo dos braços em Rviz, ombro esquerdo no ângulo máximo

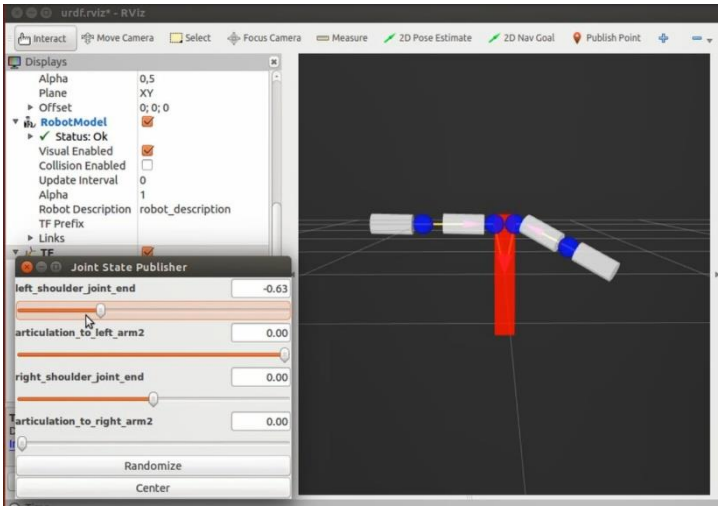


Figura 12: Protótipo dos braços em Rviz, ombro esquerdo num ângulo mínimo

Após a aquisição dos ângulos das articulações para todos os gestos, segue-se para o próximo passo que é a criação da base de dados e inserção dos gestos. O pacote mongodb_store está preparado para oferecer uma API que permite executar as operações essenciais à base de dados, sendo essas a criação, inserção e pesquisa. Por isso, o workflow desta pequena aplicação é apenas criar a base de dados localmente e depois inserir as palavras. A inserção dos dados na base de dados é através de um documento simples em JavaScript Object Notation (JSON), composto pela chave, que corresponde à palavra, e os dados, que são uma lista de valores flutuantes em que cada um representa um ângulo específico de uma articulação.

Por fim, obteve-se uma base de dados preparada para o projeto principal, usando estas configurações o projeto principal será capaz de traduzir cerca de cinco palavras, com graus de dificuldade diferentes, em que exigem o deslocamento de braços, mãos e dedos.

3.3 Implementação da aplicação

O foco deste projeto é criar uma aplicação capaz de receber textos em LP e traduzi-los para LGP, como esta língua requer gestos, foi necessário usar um simulador, onde este porventura terá um robô virtual modelizado em 3D para realizar o texto em gestos. Para que o projeto seja capaz de interpretar palavras em LP, é fundamental ter uma base de dados, para que assim, a aplicação tenha um local onde armazena as informações necessárias para ser possível associar uma palavra ao seu respetivo gesto, como também dar a informação de como se faz o gesto, de modo a ser possível replicá-lo no robô.

Com esta descrição chega-se à conclusão que há 3 componentes adicionais para a execução da

aplicação. A interface, esta possibilita que os utilizadores possam escrever textos em LP, estes textos conterão palavras que irão ser traduzidas pelo interpretador e os respetivos gestos executados pelo robô; a base de dados, onde irá conter todas informações essenciais de modo a haver uma associação de uma palavra para um gesto e por fim, o simulador que irá conter um modelo robótico virtual 3D para exemplificar os gestos que correspondem ao conjunto de palavras enviadas pelo utilizador. Sendo assim, a aplicação irá ser a ponte de ligação entre os vários componentes e o motor que irá proceder à interpretação das palavras. Na figura 13 está ilustrado um diagrama geral que descreve o conceito da aplicação com os derivados componentes.

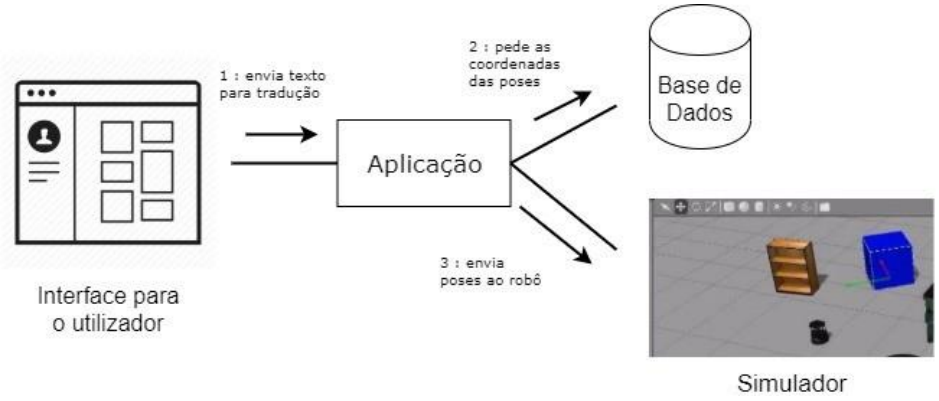


Figura 13: Diagrama geral do funcionamento da aplicação

A interface tem como objetivo mostrar as palavras disponíveis na base de dados e permitir que o utilizador possa escolher a(s) palavra(s) que deseja que sejam interpretadas. Quando o utilizador faz o pedido para interpretar uma frase, será enviada uma mensagem para o *core* da aplicação, interpretador. O conteúdo dessa mensagem é uma *string*, que continha o conjunto de palavras que foram selecionadas pelo utilizador (1ª etapa na figura 13). Quando o interpretador, recebe a frase para ser traduzida, este lê-a e deve fazer pedidos à base de dados para interpretar as palavras. A comunicação entre a aplicação e a base de dados deve seguir o padrão de comunicação de pedido-resposta, ou seja, quando é enviada uma palavra que se quer traduzir, a base de dados irá retornar, como resposta, o gesto que está associado a essa palavra (2ª etapa na figura 13). Quando o interpretador conclui a tradução das palavras, conterá uma lista de gestos, que serão enviados para o simulador, para que este os simule com o robô virtual (3ª etapa na figura 13).

Após a análise do propósito e requisitos do projeto, e de desenhada uma arquitetura que fosse mais eficaz para os requisitos levantados, tinha-se como resultado o seguinte diagrama do modelo da aplicação, figura 14.

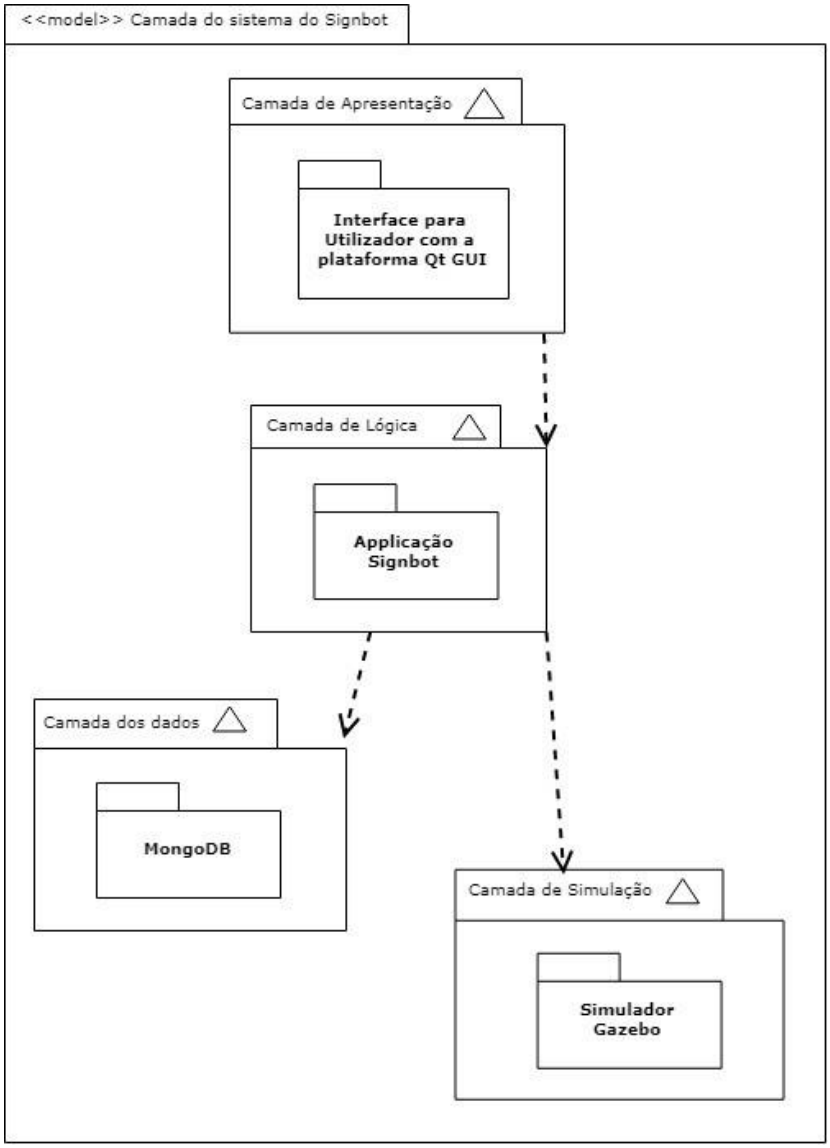


Figura 14: Diagrama modelo

No diagrama apresentado acima, estão identificadas quatro camadas, cada uma com a sua tarefa.

A camada de apresentação é responsável pela ilustração de uma página visual que permite a interação entre o utilizador e a máquina, este deve permitir ao utilizador usar a página e escolher opções e a página deve ilustrar as opções possíveis, neste caso, as palavras que a base de dados contém. Nesta camada contém o componente IG da aplicação. A descrição da implementação deste componente encontra-se com maior detalhe no capítulo 3.4 Implementação da IG.

A camada de lógica é responsável pela componente operacional do projeto, ou seja, da descrição de todos os procedimentos necessários para correr o fluxo da aplicação, neste caso, o interpretador, dado que o objetivo do projeto é implementar um robô capaz de traduzir LP para LGP. Nesta camada tinha-se a aplicação, em que foi nomeado com o nome de “Signbot”. Mais à frente neste relatório, serão detalhados os problemas e

metodologias que foram aplicadas para o desenvolvido deste componente.

A camada de dados é responsável pelo armazenamento dos dados para possibilitar o robô de fazer os gestos, cada gesto está associado a uma palavra em LP, para que seja fácil distingui-los. Nesta camada está contido o componente MongoDB, que foi a tecnologia que se usou para a base de dados do projeto. Para a escolha deste componente foi preciso analisar as tecnologias existentes e estudar qual a melhor opção para o problema do projeto. Todos estes critérios de seleção estão descritos em maior detalhe num capítulo à parte, capítulo 3.2 Implementação da base de dados.

A camada de simulação é responsável por mostrar um simulador e um robô virtual humanoide a executar os gestos quando é pedido para traduzir um conjunto de palavras. Na camada de simulação foi explorada uma solução que teve resultados negativos. Nesta solução foi explorado o uso de dois simuladores, pois pretendia-se que um dos simuladores tratasse de criar o ambiente e robô, ou seja, prepará-lo para ser apresentado ao utilizador e o outro simulador a planear os deslocamentos do gesto. Para o primeiro simulador, escolheu-se o Gazebo para representar o modelo num ambiente virtual que se assemelha ao mundo real, dado conter variáveis como gravidade e outras propriedades para ser possível criar um ambiente real, para o segundo simulador, escolheu-se o Moveit, que iria estar a correr em background e se dedicaria ao planeamento dos membros quando fosse pedido para fazer um gesto. Para esta solução pretendia-se guardar apenas as coordenadas iniciais e finais da pose de cada gesto e deixar o simulador a planear os deslocamentos. Mas esta metodologia rapidamente provou não ser exequível para o propósito do projeto, dado que o simulador se preocupa em planear a trajetória mais rápida e como tal, ele pode não respeitar o deslocamento definido para essa palavra, mesmo que se defina os gestos com mais poses, isto é, em vez de se definir apenas o início e fim do gesto, definir mais deslocamentos, por exemplo, inicio, um-quarto, metade, três-quartos e fim, não iria ser o suficiente, pois o deslocamento do gesto definido por LGP pode não ser o mais curto e isso vai contra a filosofia dos planeamentos de trajetórias no Moveit. Isto leva a outro problema, dado que o simulador Moveit, foi implementado com uma filosofia contrária à do projeto, pois os criadores queriam que o planeamento fosse dinâmico, de modo a variar com as situações; que fosse curto e eficiente, pois o tempo de deslocamento é um dos fatores importantes para a avaliação do planeamento; e queriam aplicar o conceito end of effectors do robô, que é uma extremidade do membro, por exemplo, fazer a extremidade do membro ir do ponto A para o ponto B ou planear o robô mover-se sem colisões, por causa destes fatores desistiu-se desta opção e procurou-se outra solução. A solução encontrada foi de fazer os planeamentos manuais, isto é, de definir os ângulos de cada articulação para a execução dos gestos. Nesta camada, só se tem então um componente, que é o simulador Gazebo. O objetivo do componente é meramente recriar um cenário como se fosse o ambiente real e ter o robô a executar os gestos para o

utilizador poder ver.

O projeto foi desenvolvido por cima da plataforma do ROS, este *middleware* tem uma base de dados gigante com múltiplas funcionalidades *open source* para que um desenvolve- dor possa usufruir, usando-os nos seus projetos, facilitando a vida do desenvolvedor, pois não precisa de implementar essa parte. Como tal, o projeto também usufrui de algumas funcionalidades, para simplificar a complexidade do projeto, como tal, o projeto precisa desses pacotes, instalados. Neste projeto foi preciso instalar pacotes para criar modelos 3D, pacotes dedicados a fornecer uma API para aceder à base de dados, pacotes para usar simuladores, como também incluir o modelo personalizado no simulador, e outros mais. Na figura abaixo estão identificados os vários componentes que foram instalados.

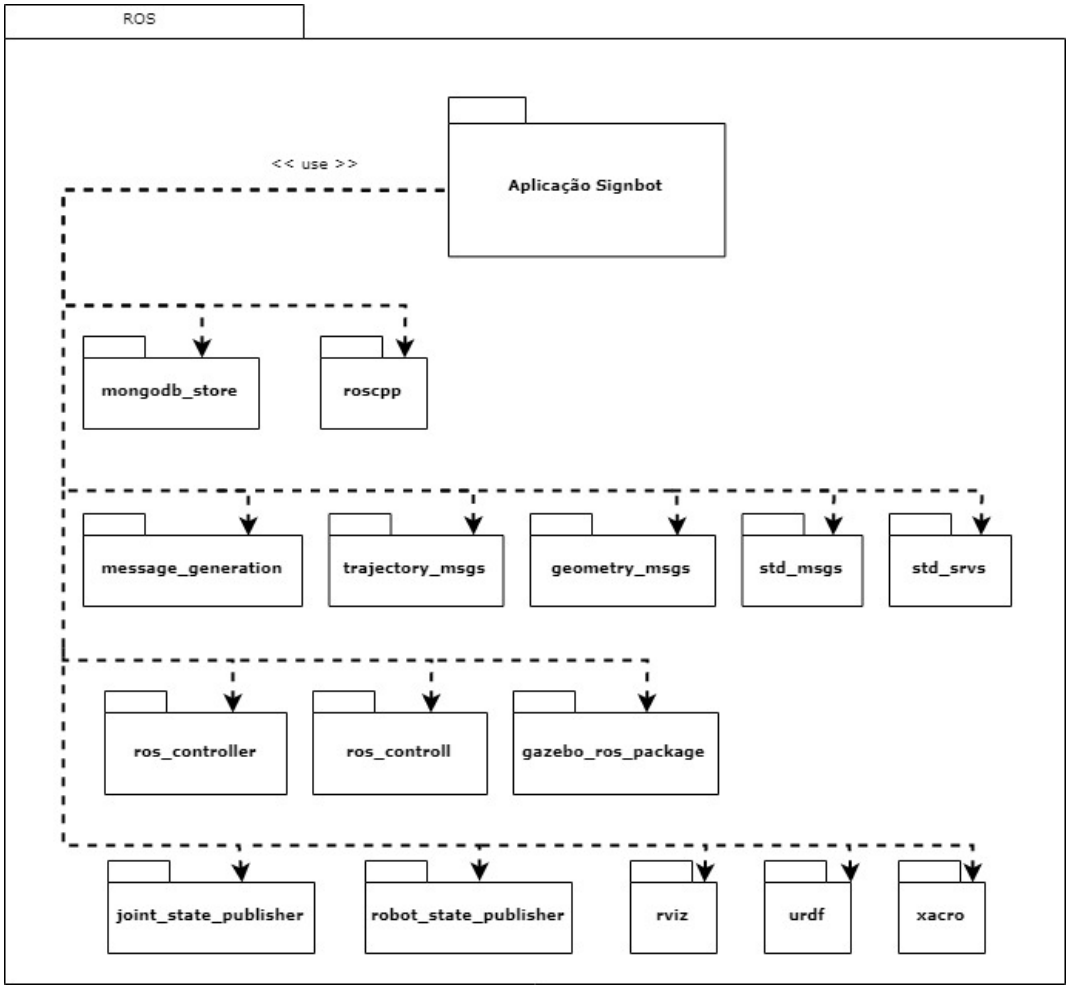


Figura 15: Pacotes de dependências do projeto

No diagrama acima estão apresentados todos os pacotes necessários para executar o projeto, todos eles oferecem uma funcionalidade crítica para a operação do sistema. Foi preciso instalar os pacotes "joint_state_publisher", “robot_state_publisher”, “rviz”, “urdf” e “xacro” para oferecer funcionalidades

essenciais para a descrição do modelo robótico. O pacote "mongodb_store" oferece uma API para se poder interagir com a base de dados. O pacote "roscpp" oferece uma API para ser possível implementar a aplicação na linguagem de programação C++. Os pacotes "message_generation", "trajectory_msgs", "geometry_msgs" e "std_msgs" oferecem funcionalidades essenciais para a comunicação dos componentes, ou seja, a criação dos tópicos e envio de mensagens. O pacote "Gazebo_ros_packages" oferece um *plugin* para permitir o uso e controlo do simulador Gazebo. Os pacotes "ros_controller" e "ros_control" servem para possibilitar o controlo e movimentação do robô.

Esta secção foi focada na descrição pormenorizada da camada lógica, ou seja, a aplicação que tem como finalidade ser o interpretador, onde este consegue receber como entrada um texto, que representa um conjunto de palavras a traduzir, e consegue interpretá-las e transformar a frase numa lista de gestos a executar. Para simplificar melhor esta lógica, segue-se um diagrama de sequência para mostrar o fluxo do sistema (figura 16).

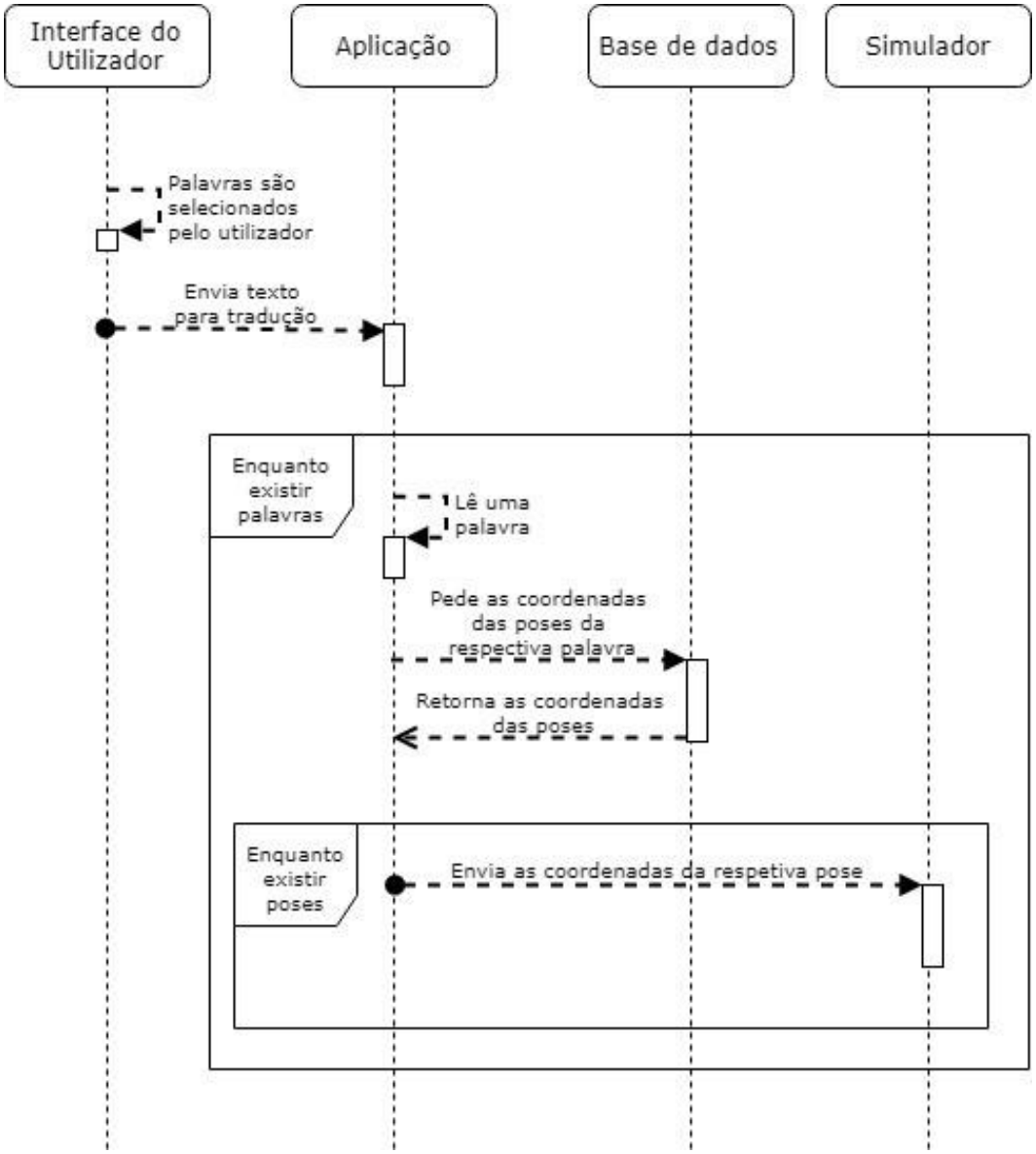


Figura 16: Diagrama de sequência

Quando a aplicação arranca esta inicializa todos os componentes necessários para que o sistema esteja pronto para executar. Todo o sistema fica ativo, mas em modo de espera, isto é, à espera de receber um sinal na entrada, que neste caso será uma mensagem. Isto ocorre porque cada componente tem a sua responsabilidade, a interface só constrói a frase para ser traduzida, de acordo com a seleção do utilizador; a aplicação fica à espera da frase para a traduzir produzindo um conjunto de gestos para enviar para o simulador; a base de dados fica à espera de palavras para traduzir, retornando os gestos que o associa; e o simulador, que tem o robô virtual num ambiente simulado, fica à espera de posições (coordenadas) para poder executar no robô, como tal, para eles comunicarem entre si, enviam mensagens. O fluxo de cada componente é esperar por uma receção de uma mensagem, fazer determinadas operações com os dados

vindos da mensagem ou de acordo com os parâmetros da mensagem e enviar os resultados das respetivas operações noutra mensagem, caso este não seja o componente final. As trocas de mensagens são feitas através de tópicos, onde cada tópico serve um propósito, por exemplo, existe um tópico específico para envio de mensagens com um texto em LP e outro para envio de poses dos gestos em LGP. O funcionamento do sistema é a troca de mensagens entre os vários nós ativos onde recebem mensagens e trabalham os dados que vêm delas e depois enviam outra mensagem com o resultado das operações.

O fluxo do sistema começa na interface com o utilizador, onde este escolhe as palavras disponíveis. O utilizador pode invocar num único pedido a tradução de uma só palavra ou de várias. Esse pedido é enviado por uma mensagem, num tópico específico. Nesse tópico há um componente à escuta, que é o interpretador. Este recebe a mensagem e começa a processá-la, entrando num ciclo, por cada palavra existente na frase. Para cada palavra este faz um pedido de tradução à base de dados, que lhe retorna um conjunto de poses que representam o respetivo gesto em LGP. Depois de receber o respetivo, conjunto de poses, entra noutro ciclo por cada pose, enviando em cada mensagem uma pose, que contem vários valores flutuantes, que representam os ângulos de cada articulação do robô.

Terminado o desenvolvimento do projeto é possível verificar o sistema a correr onde todos os componentes estão integrados uns com os outros. Para a validação do sistema é possível utilizar comandos nativos do ROS para verificar que os nós estão de facto ativos. Abaixo, encontram-se duas figuras que mostram cumprir o mesmo propósito só que por meios diferentes. Uma opção é visualizando através do rqt_ graph que apresenta num esquema, a outra opção é pela linha de comandos, usando o “rosnode list”. Nas figuras que se seguem, pode-se observar as respetivas opções.

Figura 17: Visualização dos nós ativos através do *rqt_ graph*

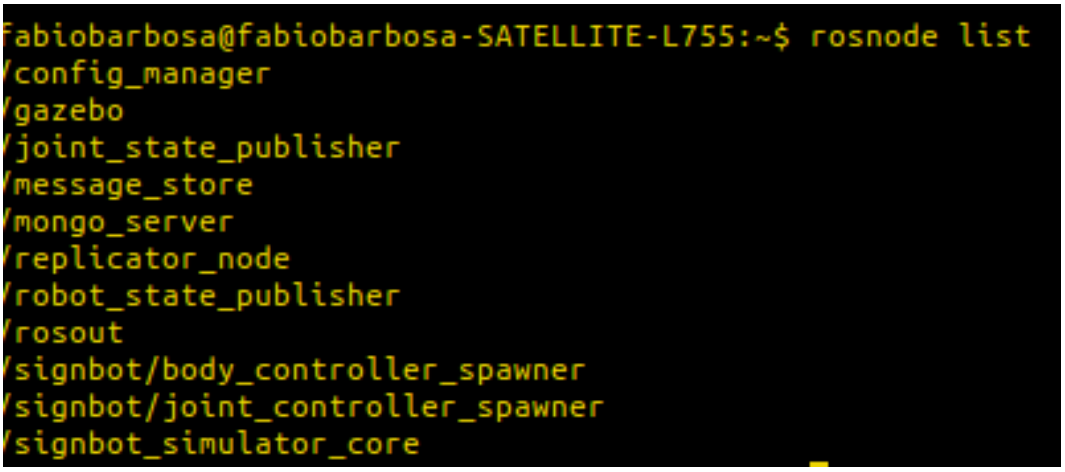


Figura 18: Visualização dos nós ativos através da linha de comandos

Como mencionado quando se estava a descrever o diagrama de sequência, a forma de os nós comunicarem uns com os outros é através dos tópicos, que têm como função ser um canal de transmissão de mensagens especificas. Uma possível forma de identificar os tópicos ativos durante a execução é usando um comando nativo do ROS, "*rostopic*". A figura abaixo mostra todos os canais ativos durante a execução da aplicação.

Figura 19: Tópicos ativos do sistema

Como é possível ver na figura 19, existe muitos tópicos ativos, mas há três tópicos que são importantes de salientar, isto porque os outros, são tópicos base, que são criados por *default* quando o *rosmaster* se inicializa. Os três tópicos que vão ser explicados com maior detalhe são: os tópicos com prefixo "/*message_store*", a sua função deste tópico é enviar pedidos à base de dados para recolher dados, ou seja, as poses do gesto; o "/*phraseLP*", a sua função é enviar mensagens que contenham a frase em português para ser traduzida e por

fim, o tópico "*signbot/body_controller/comand*", a sua função é enviar as poses do gesto, os ângulos das articulações, para que assim seja possível fazer deslocar o robô virtual.

Na figura abaixo, encontra-se o nó essencial do sistema que é o *signbot_core*, que administra a aplicação, interpretando palavras e enviando os comandos para que o robô no simulador as execute. Como descrito, o ponto de entrada deste nó é receber uma mensagem com a frase em LP, portanto, o nó está subscrito a um tópico chamado de "*/phraseLP*", neste tópico publicam-se mensagens, e que cada uma delas contem uma frase, que possui um conjunto de palavras a traduzir. Após a tradução dessas frases, o nó manda uma mensagem, para o tópico "*/signbot/body_controller/command*", nas mensagens vão um conjunto de ângulos que definem uma pose. As representações dos gestos podem exigir uma ou mais poses. Usando um comando nativo do ROS, "*rosnode info*" verificam-se as informações de um determinado nó. A figura 20 ilustra o caso do *signbot_simulator_core*.

A mesma informação pode ser adquirida fazendo a verificação a partir dos tópicos. O ROS contém

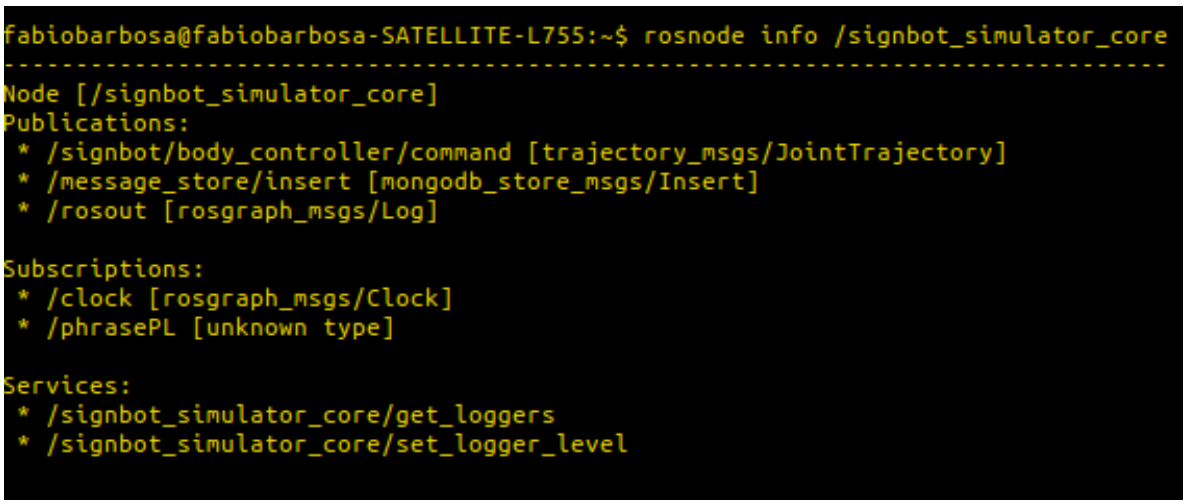


Figura 20: Informação do nó /signbot_simulator_core

também um comando para verificar isso, chamado "*rostopic*". Na figura abaixo encontra-se a informação relacionada com o tópico "*/phrasePL*" e "*/signbot/body_controller/command*".

Figura 21: Informação sobre os tópicos "*phraseLPL*" e "*/signbot/body_controller/command*"

Para validar a integração entre a aplicação e o simulador Gazebo, foram feitos testes de integração, em que se pretendia publicar uma mensagem para o tópico que o Gazebo estava à escuta, "*/signbot/body_controller/command*", neste teste espera-se que após a publicação da mensagem, o robô desloca-se de acordo com as definições na mensagem. Na figura 21, é especificado o tipo das mensagens, que é *trajectory_msgs/Joint- Trajectory*. Consultando a *wiki* do ROS [68,69], verifica-se que para fazer uma publicação válida é necessário definir uma lista de articulações do robô e os respetivos ângulos de deslocação. Estes ângulos devem definidos em radianos e devem respeitar os limites das articulações. Sendo assim, o comando enviado para o tópico "*/signbot/body_controller/command*" foi o seguinte:

rostopic pub /signbot/body_controller/command trajectory_msgs/JointTrajectory "{joint_na- mes: ["r_shoulder1_joint", "r_shoulder2_joint", "r_arm_joint", "r_elbow1_joint", "r_el- bow2_joint", "r_wirst1_j", "r_wirst2_j", "r_wirst3_j", "r_bottom_knuckle1_thumb_j", "r_- bottom_knuckle2_thumb_j", "r_top_knuckle_thumb_j", "r_bottom_knuckle_index_j", "r_- middle_knuckle_index_j", "r_top_knuckle_index_j", "r_bottom_knuckle_middle_j", "r_mid dle_knuckle_middle_j", "r_top_knuckle_middle_j", "r_bottom_knuckle_third_j", "r_middle_- knuckle_third_j", "r_top_knuckle_third_j", "r_bottom_knuckle_little_j", "r_middle_knuc- kle_little_j", "r_top_knuckle_little_j", "l_shoulder1_joint", "l_shoulder2_joint", "l_arm_- joint", "l_elbow1_joint", "l_elbow2_joint", "l_wirst1_j", "l_wirst2_j", "l_wirst3_j", "l_bot- tom_knuckle1_thumb_j", "l_bottom_knuckle2_thumb_j",

"l_top_knuckle_thumb_j", "l_bot- tom_knuckle_index_j", "l_middle_knuckle_index_j", "l_top_knuckle_index_j", "l_bottom_- knuckle_middle_j", "l_middle_knuckle_middle_j", "l_top_knuckle_middle_j", "l_bottom_- knuckle_third_j", "l_middle_knuckle_third_j", "l_top_knuckle_third_j", "l_bottom_knuc- kle_little_j", "l_middle_knuckle_little_j", "l_top_knuckle_little_j"], points: [{positions: [1.57, 1.50, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.57, -1.50, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], time_from_start: [1.0, 0.0] }]} "

As imagens abaixo ilustram o que aconteceu no simulador, respetivamente ao robô virtual quando esse comando é publicado para o tópico.

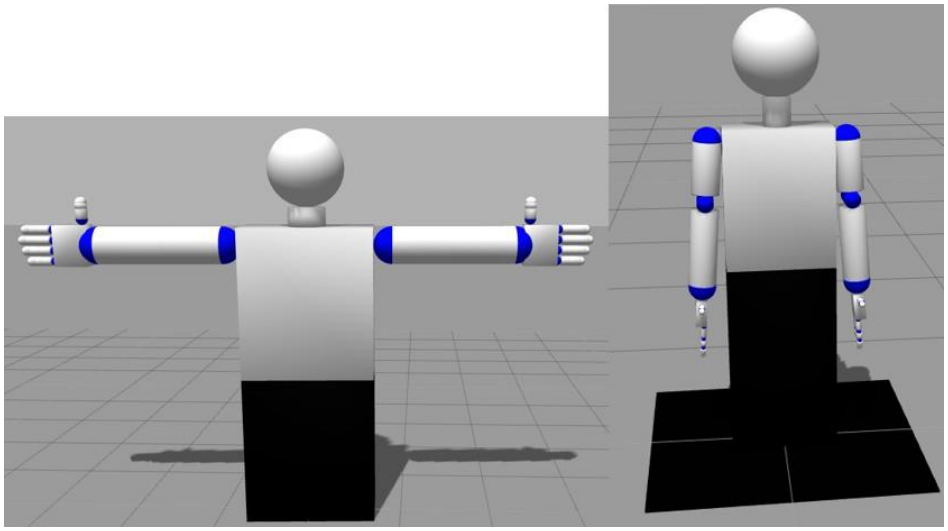


Figura 22: Deslocamento dos braços do robô antes de o comando ser executado (figura da esquerda) e após comando ser executado (figura da direita)

Relativamente ao tópico com o prefixo "/*message_store*", pode-se retirar informações da mesma forma como se fez com os outros tópicos. Usando o comando "*rostopic*". Na figura 23, é identificada a informação referente ao tópico.

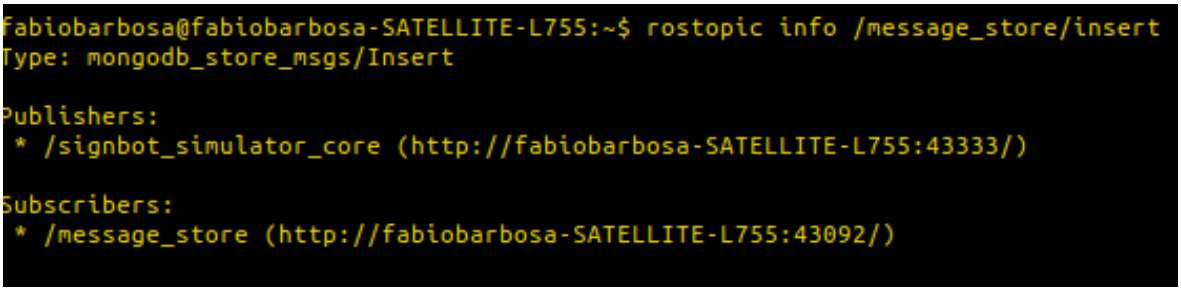


Figura 23: Informação sobre o tópico "*/message_store/insert*"

Como esperado, o tópico tem um publicador, que é o Nó "*/signbot_simulator_core*". O Nó usa este tópico para fazer pedidos à base de dados, e este retorna as poses do gesto da palavra pedida. O Nó subscrito é o Nó denominado de "*/message_store*". Este Nó é que realmente interage com a base de dados. Caso não encontre a palavra procurada na base de dados retorna uma mensagem vazia para alertar que a pesquisa foi inválida.

3.4 Implementação da IG

Como foi descrito no capítulo anterior, para se mandar um pedido para interpretar um conjunto de palavras é preciso enviar uma mensagem pelo tópico "*/phrasePL*, através da linha de comandos. A linha de comandos não é muito intuitiva porque força o utilizador a conhecer o projeto, pois precisa de ter conhecimento das palavras que existem na base de dados, como também de arrancar o pro- grama, conhecer ROS o suficiente para saber como se envia mensagens para os tópicos, através da linha de comandos, e também ter conhecimento quais são os tópicos responsáveis por receber as palavras para serem traduzidas e enviadas para o simulador. Dado isto, foi implementada uma *Graphical User Interface* (IG), para simplificar ao utilizador o uso da aplicação.

Vários modelos de *design* eram possíveis de implementar, mas dinimuiu-se para que a IG respeita-se todos os requisitos do projeto. É esperado que o projeto tenha uma interface *user friendly*, simples e intuitiva, como também a sua usabilidade se abstraia do modo de execução da aplicação, ou seja, ser possível aos utilizadores premir um botão e com ele fazer correr a aplicação sem ter a noção de como este funciona. Com isso em mente, chegou-se a uma conclusão que resultou na seguinte figura, Figure 11, que representa um diagrama de casos de uso da interface.

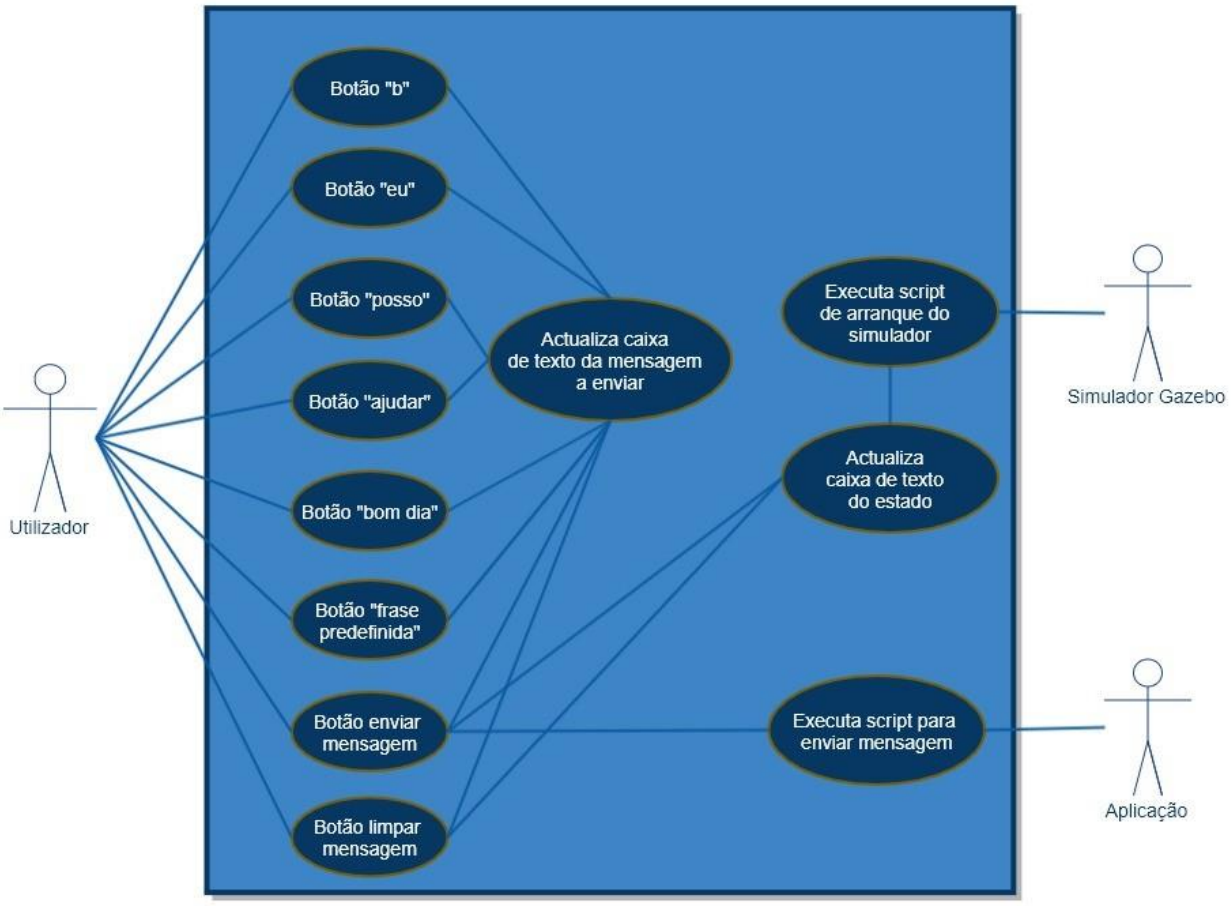


Figura 24: Diagrama de use case da aplicação IG

Com o diagrama presente acima, é possível entender que o utilizador só vai ter visibilidade apenas a botões que representam palavras ou frases, e às caixas de textos que mostram a mensagem a enviar, como também o estado da aplicação. O utilizador só está habilitado na IG a premir os botões disponíveis e estes iniciam processos em background. Isto é, todos os botões quando são premidos atualizam a caixa de texto da mensagem a enviar. Em outros cenários, botões específicos podem atualizar a caixa de texto do estado, como os casos do botão de enviar a mensagem e limpar a mensagem. Há dois processos que trabalham com os atores de saída, a aplicação em si e o simulador Gazebo. Quando se arranca a janela da IG, este arranca um processo em background para executar um script responsável para iniciar o simulador Gazebo e instanciar o modelo do robô. Outro processo só é ativado quando o botão de enviar mensagem for premido. Este processo irá executar outro script responsável para enviar a mensagem a um tópico, instanciado pela aplicação, que estará à espera de receber mensagens com o texto em LP, para serem traduzidas para LGP e enviadas para o robô virtual.

A apresentação da aplicação é composta por uma janela do simulador Gazebo, que é imediatamente instanciada, quando a IG arranca e uma janela desenvolvida por QtCreator em C++ para mostrar as possíveis opções de palavras que existem na base de dados e o estado da aplicação, como se pode ver nas figuras 25 e 26.

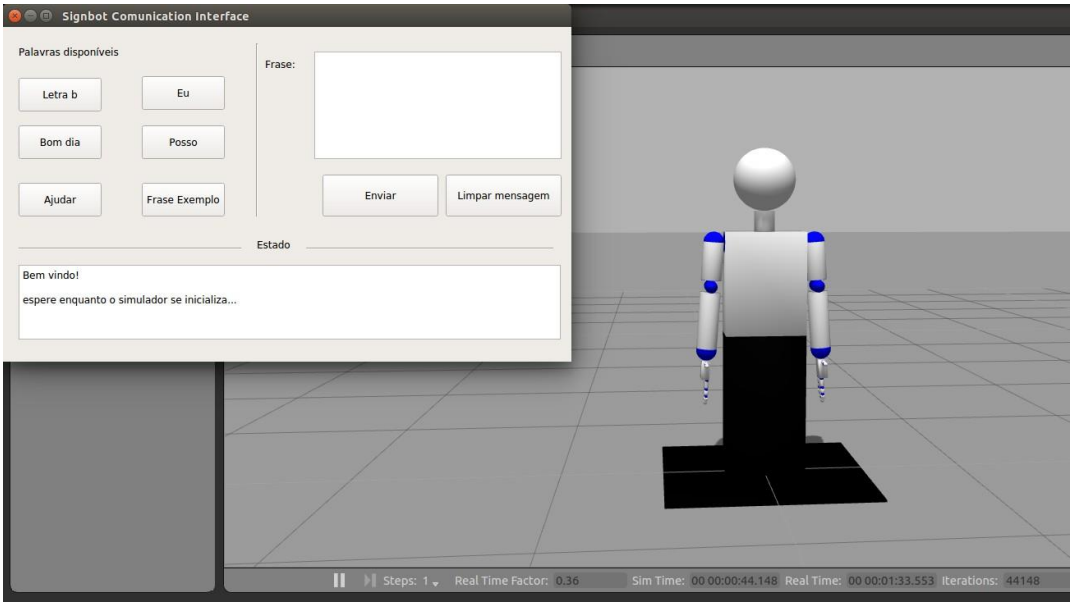


Figura 25: Apresentação geral da aplicação

Figura 26: Janela IG da aplicação

Na figura acima, figura 26, verifica-se que a janela IG está separada por três secções, uma na

esquerda que contém seis botões, com cinco deles a representarem uma a duas palavras simples como a letra “B”, “Eu”, “Bom dia”, “Poder”, “Ajudar” e o sexto botão é uma frase pré-definida, “Bom dia! Em que posso ajudar?”. A secção da direita tem uma caixa de texto, que mostra as palavras/frases selecionadas a partir dos botões da secção da esquerda e abaixo da caixa de texto tem dois botões, o “Enviar”, que é para o utilizador enviar o pedido de tradução para um tópico específico e o botão “Limpar mensagem” que limpa todas as palavras/frases que foram inseridas na caixa de texto. A terceira e última secção encontra-se posicionada no fundo da janela e é responsável por mostrar o estado da aplicação. Esta secção é só composta por uma caixa de texto para enviar algumas mensagens sobre o estado da aplicação, como a tradução da mensagem, se a aplicação está pronta para começar a interpretar palavras ou então dar o feedback que a mensagem foi lida e vai enviar os gestos para a simulação.

Selecionando os botões das palavras/frases na secção da esquerda, estas começam a ser registadas na caixa de texto da secção da direita, como se pode ver nas próximas imagens. É de notar que a janela IG permite a possibilidade de o utilizador traduzir uma ou várias palavras num só envio para o tópico que espera palavras para traduzir.

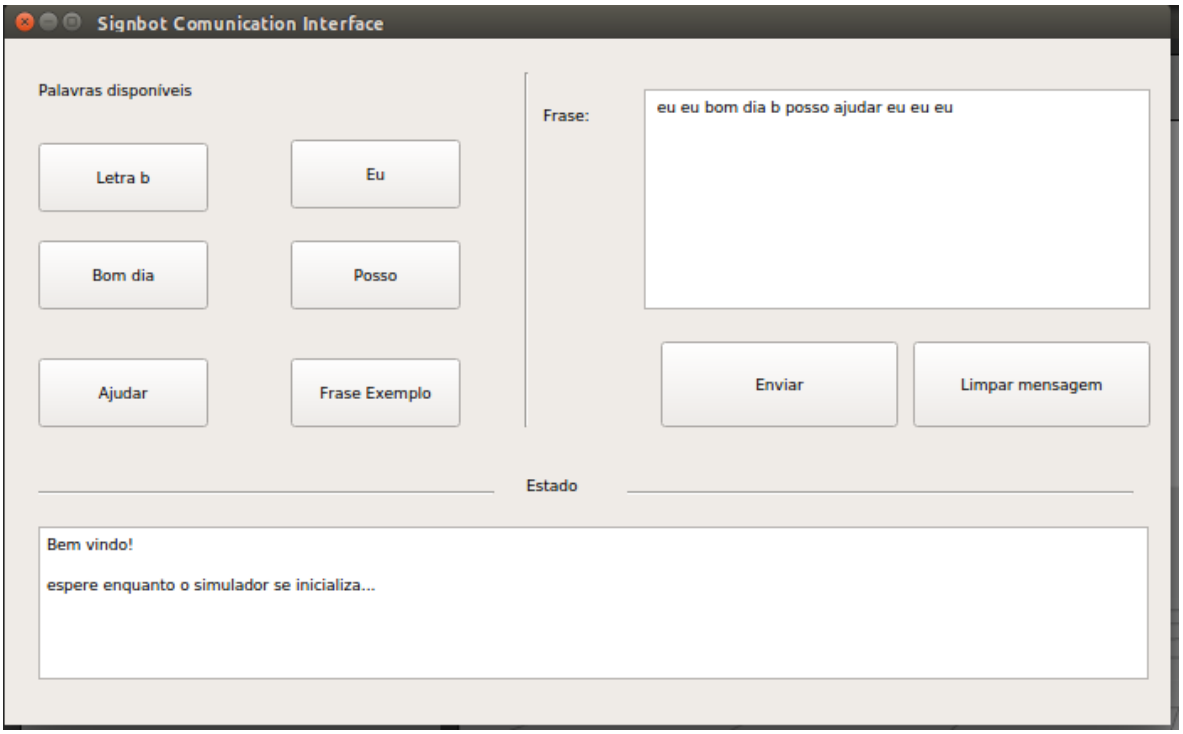


Figura 27: Janela IG - com várias palavras

Após a escolha das palavras a traduzir, o próximo passo é a escolha de apagar a mensagem que se construiu ou se pretende enviar a frase para a aplicação interpretar. A figura abaixo, figura 28, apresenta um

cenário que se pretende apagar a mensagem. De notar que para além da caixa de texto da secção da direita ser apagada, a caixa de texto na secção do estado também reporta informação que a frase foi apagada.

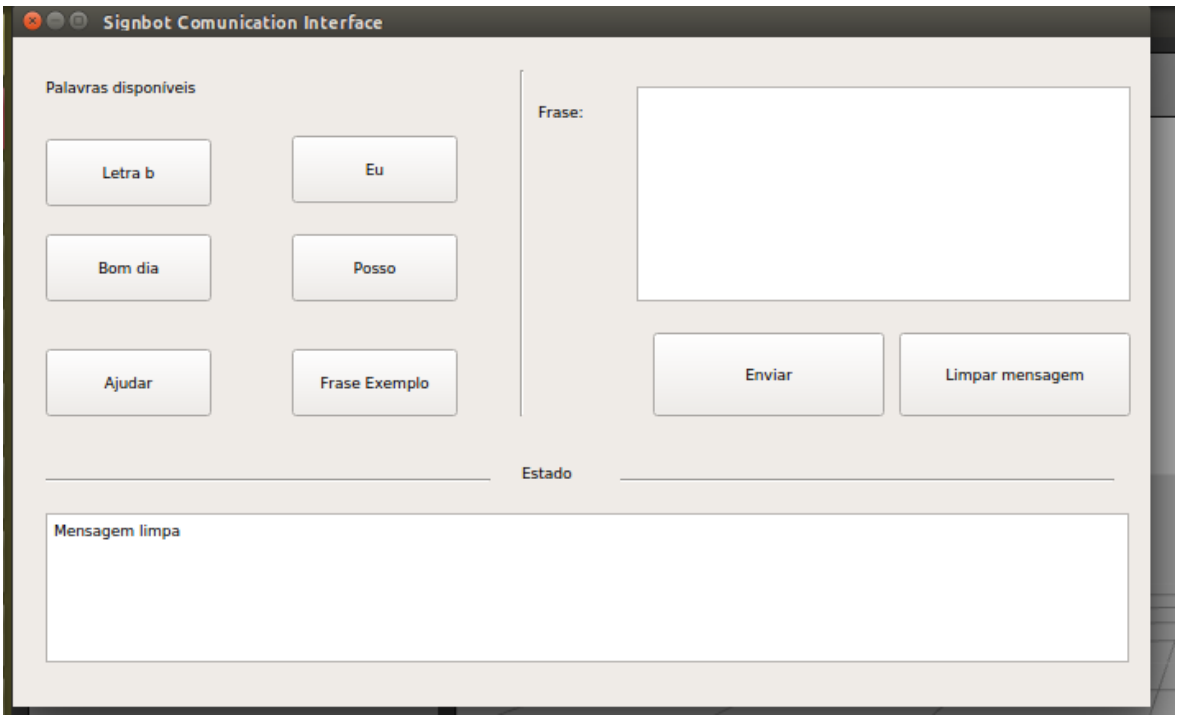


Figura 28: Janela IG - mensagem limpa

Se for premido o botão "Enviar" com uma frase já construída, a caixa de texto da secção da direita apaga e a caixa de texto da secção do estado atualiza, mostrando a frase já traduzida em LGP, reportando se a mensagem foi enviada com sucesso para o tópico e se o robô irá começar a fazer os gestos, como aparece na imagem abaixo, figura 29.

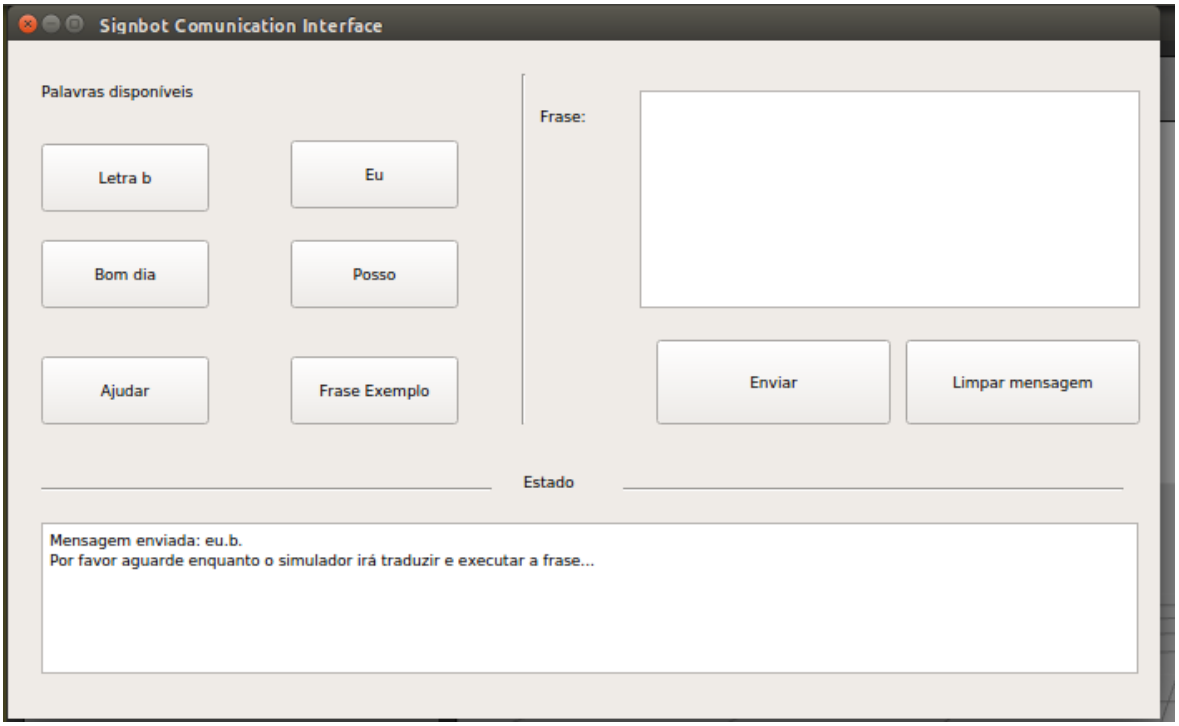


Figura 29: Janela IG - mensagem enviada

Muitas das vezes o texto criado irá ser igual quando se escreve com as palavras simples. Isto porque a aplicação ainda não capacidade para conseguir interpretar uma frase em LP LGP seguindo as regras do LGP. Neste caso verifica-se com a imagem abaixo, figura 30, que apesar de se ter construído manualmente a frase "Bom dia posso ajudar" em LP, a caixa de texto do estado mostra que a frase em LGP é interpretada na mesma forma. Caso fosse premido o botão com a frase pré-definida e em seguida enviada a frase, pode-se reparar, a partir na figura 31, que na caixa de texto na seção do estado, a frase em LGP muda a ordem das palavras. Este cenário serve para mostrar o que deveria acontecer se a aplicação tivesse conhecimento de construção das frases em LGP.

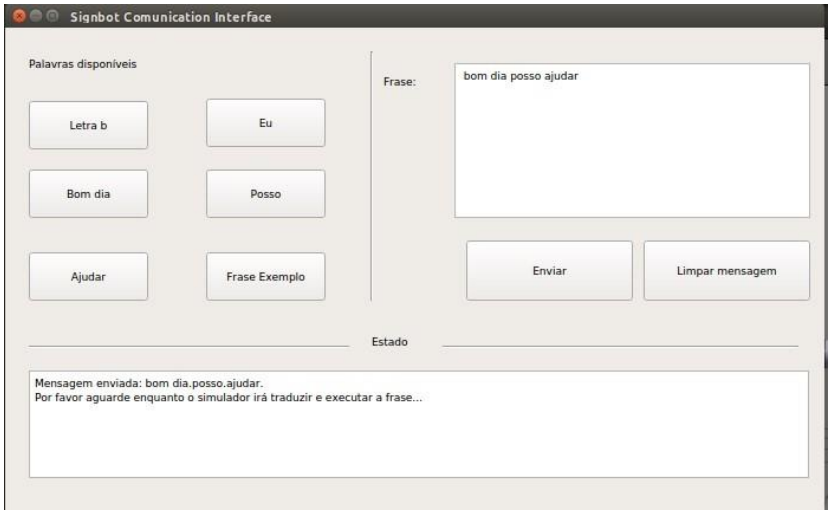


Figura 30: Frase construída manualmente

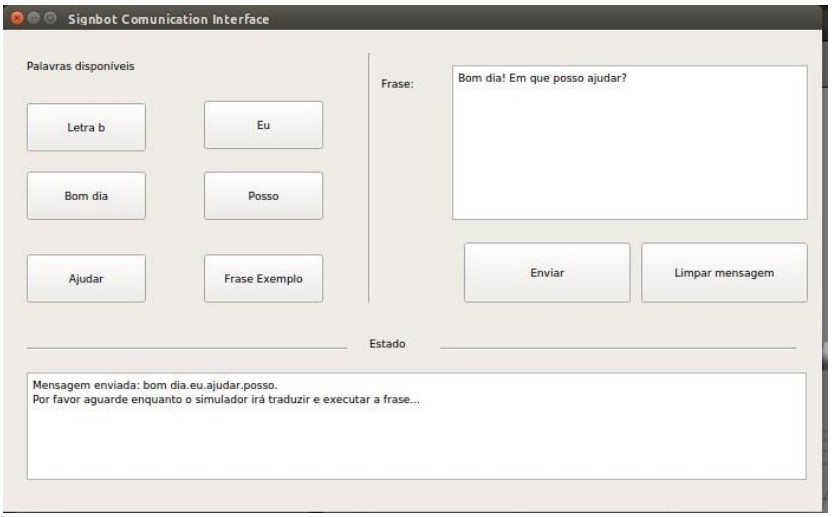


Figura 31: Frase construída pelo botão predefinido

3.5 Sumário

O projeto mostrou ser bastante interessante pois trabalha com temas sérios e complexos, mas acima de tudo, o projeto aborda um problema real e arranja uma opção viável para o resolver. Este projeto prova ser multidisciplinar, pois trabalha na área de robótica, onde este aborda *framework*s complexas como o ROS e simulações, e destina-se à área de saúde, pois procura ajudar pessoas com necessidades.

Durante o processo de desenvolvimento verificou-se que houve mudanças de metodologias de implementação por terem sido encontradas limitações no primeiro plano delineado. No capítulo 3.1 Especificação do modelo robótico, foi descrito o processo de modelar o robô, ao desenhá-lo foi necessário compreender melhor o funcionamento do corpo, analisando as capacidades das articulações. Durante a

modelagem do robô foi encontrado um desafio que levou a ponderar qual a melhor maneira de modelar o robô para ser possível recriar as articulações do ser humano. Dado que as articulações nos braços têm entre dois ou três Graus de Liberdade (GDL). Após um estudo, chegou-se à conclusão de que a melhor opção seria a construção de articulações com apenas um GDL, e no caso de uma zona ter três GDL, construir 3 articulações juntas para representar o mais fielmente possível o ser humano. No capítulo 3.2 Implementação da base de dados, foi descrita a escolha das palavras, ao todo foram selecionadas 6 palavras, onde estas foram selecionadas com intenção de ser possível construir uma frase. Outro ponto que se descreveu foi implementação da base de dados, onde foi discutida qual a melhor tecnologia a usar para armazenar os gestos, se deveria optar-se por uma base de dados relacional ou não relacional, e como também esses gestos deveriam ser definidos, a partir dos testes feitos ao modelo protótipo, verificou-se que a melhor opção seria armazenar os ângulos das articulações em radianos. No capítulo 3.3 Implementação da aplicação foram explicados os requisitos do projeto e a sua arquitetura. Depois, descreveu-se o fluxo do sistema e como os diferentes componentes interagem entre eles. Neste capítulo foi identificado um problema, sendo que a primeira solução provou não ser exequível, e como tal, foi preciso mudar a estratégia para se conseguir atingir os objetivos e requisitos definidos. No capítulo 3.4 Implementação da IG foi descrita a lógica da interface, esta tem como objetivo permitir ao utilizador escolher as palavras que quer traduzir, e para tal foram definidos os vários casos de usos e a descrição das operações que a interface contém.

4 ANÁLISE DE RESULTADOS

Neste capítulo irão ser descritos os testes que foram executados de modo a validar o projeto, dado que, desta forma percebe-se realmente as vantagens e limitações do projeto, como também determinar se a aplicação conseguiu ser desenvolvida de forma a cumprir os vários objetivos pretendidos. Como tal, foram descritos três testes, explicando o seu propósito e como o teste é procedido. No final de cada uma das descrições foi feito um resumo dos resultados, discutindo alguns pontos relevantes.

4.1 Tempo de execução da aplicação

Um dos testes é saber se a aplicação é rápida e bastante responsiva aos comandos do utilizador. Para isso criaram-se os seguintes cenários de teste que validam a rapidez e o tempo de resposta de um dado comando. Os cenários de teste são:

- Cenário 1: O arranque da aplicação é rápido?
- Cenário 2: A interface responde bem aos comandos do utilizador?
- Cenário 3: Dado um pedido de tradução, qual a rapidez com que o robô responde?

4.1.1 Resultados

Os cenários foram executados e os tempos de processamento da aplicação foram cronometrados. Repetiu-se o teste na mesma máquina 5 vezes para se poder obter uma média entre o pior e melhor cenário. A tabela abaixo, tabela 4, apresenta a média obtida para cada um dos cenários.

Tabela 4: Testes de resposta e fluidez do sistema

Cenários de teste	Média (segundos)
1: Tempo gasto para arrancar a aplicação	7
2: Tempo gasto para a IG responder aos comandos do utilizador	<1
3: Tempo gasto desde o pedido de tradução até o robô começar a traduzir	1-2

No cenário 1 verifica-se que aplicação precisa de um certo tempo para executar e preparar o sistema. Isto deve-se a uma combinação de dois fatores. Quando o sistema arranca este abre a IG, cria o ROS Master, os

Nós e tópicos, e, por último, abre o simulador Gazebo. Como no arranque da aplicação são processadas muitas instanciações de objetos e até mesmo de outras aplicações, implica que o sistema vai demorar a iniciar e, dado que o simulador é uma aplicação pesada, este tende a gastar algum tempo para executar todos os seus processos, mesmo quando esta é aberta individualmente nota-se um tempo de espera significativo. O outro fator que aumenta o tempo de espera são as caraterísticas do computador, que podem ser consultadas no anexo 3 – Caraterísticas do computador.

Para o cenário 2, a IG mostra-se bastante responsiva, mostrando tempos muitos baixos. Este resultado é excelente, pois atinge o que se pretende.

Para o cenário 3, verificou-se que há um atraso entre o pedido do comando e a execução do robô. Isto deve-se, novamente, às características do computador, apresentadas no anexo 3, pois verifica-se que não só o sistema fica menos responsivo como também o sistema operativo, por isso, este valor reflete um mau resultado pois está a ser bastante influenciado pelas capacidades do computador usado.

4.2 Tempo de execução dos gestos

Outro cenário importante de testar é averiguar se os tempos de execução dos gestos são iguais ou parecidos, quando o robô e um praticante de LGP os fazem. Para o propósito foram testados todos os gestos que a base de dados contém de momento, um total de cinco gestos e mais uma frase pré-definida.

4.2.1 Resultados

Foi pedido a um intérprete para fazer cada um dos gestos que a aplicação tem na base de dados e, por fim, a frase pré-definida. Depois executou-se a aplicação para fazer os mesmos gestos e a frase pré-definida. Os resultados obtidos encontram-se na tabela abaixo, tabela 5.

Tabela 5: Tempo de execução de cada gesto

Gestos	Executados pelo robô (segundos)	Executados por uma Pessoa (segundos)
Letra 'b'	5	1
Bom dia	9	2.30
Ajudar	8	1.30
Eu	4.30	1.30

Poder	8.30	2
Frase: “Bom dia! Em que posso ajudar?”	27	6

Analisando a tabela 5, pode-se concluir que o robô é muito mais lento, mas isto deve-se porque foi pedido a um praticante com muita experiência, ou seja, este sujeito já tem muita prática e fluidez no movimento. Para iniciantes da língua a velocidade é reduzida, pois encontram-se na fase de aprendizagem e também precisam de ver os gestos mais lentamente para conseguirem entender, por isso, pode-se aproveitar o robô ser mais lento para que qualquer sujeito, iniciante ou experiente, entenda os gestos.

Outro ponto a evidenciar é que foi estabelecida uma velocidade que se considerou boa para a compreensão dos gestos executados, caso essa velocidade seja considerada demasiado lenta, é sempre possível alterar os tempos de execução, dado que no simulador não existem limitações das velocidades de rotação e deslocamento dos membros. Por isso, de momento, o robô tem esta velocidade, para que qualquer indivíduo perceba os gestos, mas é possível no futuro acelerar os movimentos do robô no simulador, sem ter alguma penalização.

4.3 Teste com praticantes LGP

Este teste foca-se em validar o projeto como um todo, para isso foi pedido a um pequeno grupo de pessoas que sabem LGP para verem o robô 3D do simulador a fazer gestos quer de uma palavra ou frase. Com esse intuito, pediu-se principalmente a intérpretes de LGP para fazerem de avaliadores. O objetivo do teste é analisar e avaliar o desempenho do robô virtual a reproduzir os gestos. O teste que se propôs a fazer com os intérpretes seria um teste prático, isto é, sem nenhum uso de guias detalhados com os passos a executar. Pretendeu-se mostrar a aplicação a correr e dar a liberdade ao intérprete de mexer na aplicação, dada a sua interface ser simples e com um pequeno leque de botões para explorar. Desta forma, os utilizadores tinham todo o controlo do teste, escolhendo os gestos que desejavam ver e fazer as combinações à sua escolha.

Com a interface da aplicação, o utilizador podia executar um gesto estático, que era uma letra do alfabeto, letra ”B”; gestos dinâmicos que representam palavras como “bom dia”, “ajudar”, “eu” e “poder”; e o último botão continha uma frase pré-definida pela aplicação, “Bom dia. Em que posso ajudar?”. Os testes envolviam o utilizador a premir um botão para visualizar esse específico gesto que queria ver ou então um conjunto deles, dado que a aplicação só começa a traduzir quando o utilizador carregar no botão de “enviar”. Sendo assim, a aplicação permite a opção de misturar um conjunto de palavras que existam na

biblioteca (base de dados) conseguindo formar uma frase. Isto deve-se porque a aplicação ainda não tem a funcionalidade de tradução inteligente, em que é capaz de interpretar frases de LP para LGP.

4.3.1 Resultados

No fim dos testes foram feitas algumas perguntas acerca da representação dos gestos que o robô 3D executou, tendo como finalidade das perguntas perceber se esses gestos eram capazes de ser interpretados por qualquer indivíduo que fale LGP, ou até mesmo perceber o quão próximo foram os gestos de replicar perfeitamente uma pessoa.

Foi pedido aos testadores que descrevessem o que pensavam acerca do projeto. Pedindo que classificassem os gestos em termos de fluidez de movimentação dos membros, de rapidez e se eram bem distintos e fáceis de compreender naturalmente. Depois de recolhida toda a informação do grupo de pessoas que testaram a aplicação, as conclusões dos resultados estão identificadas na tabela abaixo, tabela 6. Os valores da tabela demonstram as avaliações numa escala de 1 a 5. Sendo 1 - discordo complemente; 2 - discordo; 3 - indiferente; 4 - concordo e 5 - concordo completamente.

Tabela 6: Resultado do teste do protótipo

	Os gestos que foram rápidos?	Os movimentos dos gestos foram fluidos?	O gesto foi fácil de identificar?	Consegui identificar todo o contexto da frase?
Letra B	5	5	5	NA
Bom dia	4	4	5	NA
Ajudar	4	4	5	NA
Eu	5	4	5	NA
Posso	4	5	5	NA
Frase “Bom dia em que posso ajudar?”	4	4	5	4
Várias palavras agrupadas aleatoriamente	3	3	4	-

Uma das questões realizadas inquiria sobre a rapidez e fluidez dos gestos, para cada cenário, entendendo que um gesto rápido é determinado pelo quão rápido foi feito, quando esse é comparado com a velocidade média de um praticante de LGP, e a fluidez do gesto, a forma como os membros se mexem. Nesta secção observa-se que a classificação 5 aplica-se principalmente a gestos que só requerem um

movimento e a classificação 4 é a média da soma de todos os testes, sendo isto algo esperado dado que, no desenvolvimento dos gestos, um dos parâmetros da definição do gesto é o tempo que deve demorar a movimentar-se. Os tempos baseiam-se na duração que os praticantes de LGP demoram a fazer e apesar de o robô ter o mesmo tempo a mover os membros, o tempo associado à transição de um movimento para outro já não se assemelha ao do ser humano, dado que a máquina, o computador do desenvolvedor, demora a processar. Daí ser expectável que as avaliações andem em torno do valor 4.

As avaliações com valor 3 resultam de se tratar de várias palavras, com um ou mais gestos, pois quantas mais transições de movimentos o robô tem que fazer, mais a máquina acresce tempo de processamento a essa parte. É, igualmente, normal que as pessoas façam uma pequena pausa depois de cada frase ou conjunto de palavras, no entanto como a aplicação não tem essa sensibilidade está, por defeito, fazer uma pequena pausa após cada palavra.

Outras questões feitas abrangiam: se o gesto era facilmente reconhecido, para isso avaliou- se se o gesto efetuava a trajetória certa; e se os membros estavam bem posicionados ao representar o gesto. Nesta questão o resultado atingiu valores quase ideais, em que todas as palavras e a frase foram facilmente identificadas e representadas. O problema surgia quando o utilizador construía um conjunto de palavras. Como algumas palavras em LGP têm vários contextos, implica que nem todas palavras em LP correspondem apenas a um gesto em LGP e como a biblioteca foi preparada com associações de um para um, isto é, uma palavra em LP só está ligada a um gesto em LGP, nesses casos específicos a aplicação não irá corresponder aos gestos certos.

A última questão aplica-se apenas para o cenário da frase, portanto na tabela acima, tabela 6, a última coluna só será preenchida quando se referir à frase. O objetivo desta questão era perceber se a frase estava bem representada, não só mostrando os gestos corretamente, mas também dando emoção à frase, porque em LGP por vezes intensificam-se algumas frases, para dar mais significado às ações. Como o robô não tem cara, as emoções são não existentes.

4.4 Sumário

Para avaliação do projeto foram realizados alguns testes, feitos através da IG da aplicação. Vários cenários foram apresentados para experimentar, depois deu-se a possibilidade de as pessoas que testaram a aplicação contruírem o seu próprio conjunto de palavras, escolhidas por elas. Os testes baseavam-se em representar um leque pequeno de palavras, dado o projeto ser uma prova de conceito. Sendo que as palavras definidas foram: letra “B”, “eu”, “poder”, “ajudar”, “bom dia” e a frase “Bom dia! Em que posso

ajudar?". Cronometrou-se os tempos de execução de cada um dos gestos para facilmente se poder comparar com os tempos executados por uma pessoa, os resultados levaram à conclusão que um ser humano com experiência é mais fluente e rápido, mas, mesmo assim, os tempos que o robô atingiu não são maus, e a sua velocidade assemelha-se à de um iniciante. Sendo assim, o robô mostra ter a velocidade certa para que qualquer praticante de LGP, independente da sua experiência, facilmente entenda o que o robô está a dizer.

No final foram feitas algumas questões a respeito da aplicação. Como por exemplo, se os gestos foram bem feitos, rápidos e fluídos. Os resultados obtidos provam ser positivos, mas também se encontraram algumas limitações do projeto, sendo uma delas a falta de rosto para mostrar expressões faciais enquanto traduz frases, dado que o rosto é um fator fundamental para quando se quer intensificar o contexto de uma frase.

5 Conclusão e perspetivas futuras

O interesse deste projeto é de o estudar e implementar um sistema fiável e altamente responsivo em tempo real que tem como objetivo resolver um problema, na área da saúde, focando-se maioritariamente nas pessoas com deficiências, mais especificamente surdas e/ou mudas. Foi planeado para este projeto implementar os requisitos com apoio das infraestruturas e funcionalidades da área de robótica, isto é, o projeto é multidisciplinar e interage com a área de robótica aplicada à saúde. Durante o desenvolvimento do projeto realizaram-se os seguintes passos:

- Estudo e compreensão de outras aplicações com o mesmo âmbito oriundo do estrangeiro;
- Exploração de alguns simuladores, de modo a conseguir definir qual o melhor simulador para implementar as funcionalidades requeridas durante a análise do projeto;
- Desenho de um modelo genérico de um robô humanoide, baseado nas medidas de ser humano adulto;
- Implementação de uma infraestrutura de modo a ser capaz de correr a aplicação sem estar dependente do robô físico, ser rápida e contendo todas as funcionalidades essenciais para provar o conceito;
- Implementação de uma base de dados não relacional para armazenar todos os gestos já adquiridos, preocupando-se com a performance da aplicação e estabilidade;
- Implementação de uma aplicação IG para facilitar a realização dos testes, como também a introdução desta nova infraestrutura para outras pessoas.

5.1 Conclusões

A comunicação via gestos sempre foi uma boa maneira para indivíduos que estão impossibilitados de comunicar oralmente, devido a esse fator, a língua gestual, neste caso LGP, foi criada para ajudar pessoas que não poderiam usar a fala. Na robótica já se verificam também os primeiros passos a ser implementados referentes à comunicação via gestos.

Para este projeto propôs desenvolver uma infraestrutura que traduzisse e manipulasse membros robóticos, como mãos e braços, para se conseguir realizar gestos associados a uma palavra em LP. Dando assim a possibilidade de avançar mais uma etapa de interação entre Homem e máquina. O objetivo era

conseguir implementar um protótipo capaz de apoiar pessoas com deficiência, criando uma aplicação capaz de traduzir palavras em LP para LGP. Outro aspeto importante a reter é de se estar a implementar um projeto com foco a se tornar uma infraestrutura abstrata, ou seja, que não depende das propriedades físicas do robô, capaz de simular um modelo genérico, mostrando os gestos e sabendo que o que aparece no simulador será uma representação fidedigna do que irá acontecer com um robô físico.

As pessoas que testaram a infraestrutura responderam positivamente às perguntas feitas e afirmaram que a infraestrutura se mostrou capaz de simular e representar gestos de acordo com LGP. Esta conseguiu cumprir os requisitos definidos, alcançando boas respostas face ao problema proposto. Além das respostas positivas vindas pelos que testaram a aplicação, os intérpretes profissionais de LGP identificaram um pequeno problema, sendo este a incapacidade de o robô mostrar expressões faciais, dado que em LGP, para além de se usar as mãos para comunicar, também se usa algumas expressões na cara para realçar certas palavras, no caso de descrição de emoções ou quando se refere a proporções de um objeto.

5.2 Limitações

Um dos principais problemas identificados pelos intérpretes de LGP que chegaram a testar o projeto foi o facto de o robô não ter uma cara. Isto prejudica os gestos em LGP porque a Língua não só usa as mãos e os braços, mas também expressões faciais. Estas têm um fator significativo pois os utilizadores de LG usam as expressões faciais para salientar algo, como por exemplo, só existe apenas um gesto para dizer “pequeno”, mas com as expressões faciais pode-se salientar se o que se refere é “pequeno” ou “muito pequeno”.

Outro problema referia-se à velocidade a correr os gestos, em que o tempo que o robô demora nas transições de movimentos acaba por se tornar grande, depois de uma breve investigação concluiu-se que isto se deve ao computador usado, pois por ser um computa- dor portátil, com alguns anos, a execução do ROS, do Gazebo, mais a aplicação deixavam o computador lento. Este problema não foi identificado como sendo de grande risco, já que trocando por um computador mais recente, com maior poder computacional tornaria a execução da aplicação e ferramentas de terceiros mais fluente e mais rápidas.

O projeto foi desenvolvido com o intuito de ser uma prova de conceito, como tal esta infraestrutura é apenas um protótipo para mostrar as capacidades que a robótica pode atingir. A sua base de dados de gestos/palavras ainda é muito reduzida, pois foram registados apenas alguns gestos com o interesse de criar um pequeno leque de exemplos para pôr em prova o conceito do projeto.

5.3 Contribuições do projeto

Este projeto destinou-se a investigar uma área com pouca exploração, como tal, interessava criar uma base para que outros pudessem usufruir dela nos seus projetos ou então para a melhorar, corrigindo certas falhas ou limitações que foram identificadas nos capítulos anteriores. Sendo assim, o que se pretendia com este projeto era criar se possível uma infraestrutura fiável e genérica para qualquer tipo de robô humanoide, para que quem precisasse de usar gestos pudesse ter esta infraestrutura, como acontece em outros projetos existentes em ROS, que é criar packages, aplicações ou mesmo infraestruturas, para o interesse do próprio desenvolvedor e para terceiros.

5.4 Trabalho para futuro

Sendo este projeto um protótipo há certos elementos que foram implementados com o interesse de apresentar um conceito, como tal, na base de dados é necessário acrescentar mais dados, isto é, inserir mais traduções de palavras LP para LGP.

Um dos aspetos mencionados pelos intérpretes de LGP foi a ausência de uma cara no robô. Para trabalho futuro poder-se-ia desenhar uma cara para o robô, onde os olhos, sobrancelhas e boca fossem deslocáveis, com essas partes da cara deslocáveis, o robô poderia gerar algumas expressões faciais básicas enquanto traduzia o texto. Poderia ser estudada e implementada uma inteligência artificial, IA, focada na geração de expressões, para que o robô fosse capaz de demonstrar expressões enquanto fazia os gestos. Isto podia basear-se no contexto da conversa ou texto, para facilitar qual a expressão a usar.

De momento a infraestrutura consegue traduzir palavras individuais, mas falta-lhe uma componente de IA para que assim a infraestrutura possa aceitar frases ou textos, traduzindo-os de acordo com as palavras e o contexto, dado haver certas palavras em que a tradução não é direta pois depende do contexto.

Bibliografia

[1] “Toshiba: News release (6 Oct, 2014): Toshiba Corporation Develops Lifelike Communication Android”, Toshiba.co.jp, 2018 [online]. Disponível em: https://www.toshiba.co.jp/about/press/2014_10/pr0601.htm. [Acedido a 24 de Outubro de 2018].

[2] T. language, “Toshiba android tries to charm with sign language”, CIO, 2018. [Online]. Disponível em: https://www.cio.com.au/article/556791/toshiba-android-tries-charm-sign language/. [Acedido a 24 de Outubro de 2018].

[3] "Toshiba Android Will Take You for a Trip Down the Uncanny Valley", IEEE Spectrum: Technology, Engineering, and Science News, 2018. [Online]. Disponível em: https://spectrum.ieee.org/automaton/robotics/humanoids/toshiba-aiko-communicationandroid. [Acedido a 24 de Outubro de 2018].

[4] "Toshiba’s new robot can speak in sign language", CNET, 2018. [Online]. Disponível em: https://www.cnet.com/news/toshibas-new-robot-can-speak-in-sign-language/. [Acedido a 24 de Outubro de 2018].

[5] "Cutesy robot Asimo gets upgraded", BBC News, 2018. [Online]. Disponível em: https://www.bbc.com/news/technology-28332198. [Acedido a 24 de Outubro de 2018].

[6] B. Team, "ASIMO Robot learns Sign Language - British Deaf News", British Deaf News, 2018. [Online]. Disponível em: https://www.britishdeafnews.co.uk/asimo-robot-learns-sign-language/. [Acedido a 24 de Outubro de 2018].

[7] "ASIMO | The Most Advanced Humanoid Robot | Honda", Honda.com, 2018. [Online]. Disponível em: https://www.honda.com/mobility/say-hello-to-asimo. [Acedido a 24 de Outubro de 2018].

[8] "Honda ASIMO learns sign-language and masters stairs", SlashGear, 2018. [Online]. Disponível em: https://www.slashgear.com/honda-asimo-learns-sign-language-and-masters-stairs-18325765/. [Acedido a 24 de Outubro de 2018].

[9] Anon, (n.d.). [image] Disponível em: https://tt.m.wikipedia.org/wiki/%D0%A4%D0%B0 %D0%B9%D0%BB:HONDA_ASIMO.jpg [Acedido em12 Jul. 2019].

[10] G. Nichols, "Arduino powered and 3D printed, this robot translates to sign language | ZDNet", ZDNet, 2018. [Online]. Disponível em: https://www.zdnet.com/article/arduinopowered-and-3d-printed-this-robot-translates-to-sign-language/. [Acedido a 24 de Outu- bro de 2018].

[11] "3D-printable robot arm is a sign language interpreter", Newatlas.com, 2018. [Online].

Disponível em: https://newatlas.com/aslan-sign-language-robot-arm/50951/. [Acedido a 24 de Outubro de 2018].

[12] "3D-printed robotic arm translates speech into sign language", designboom | architec- ture & design magazine, 2018. [Online]. Disponível em: https://www.designboom.com/technology/sign-language-robot-project-aslan-08-23-2017/. [Acedido a 24 de Outubro de 2018].

[13] "There's not enough sign language translators, so these students 3D printed a humanoid robot.", 3D Hubs Blog, 2018. [Online]. Disponível em: https://www.3dhubs.com/blog/theres-not-enough-sign-language-translators-so-these-students-3d-printed-a-humanoid-robot/. [Acedido a 24 de Outubro de 2018].

[14] "ASLAN robot arm translates words into sign language for deaf people | Daily Mail Online", Dailymail.co.uk, 2018. [Online]. Disponível em: https://www.dailymail.co.uk/sciencetech/article-5517971/ASLAN-robot-arm-translates-words-sign-language-deafpeople.html. [Acedido a 24 de Outubro de 2018].

[15] "Robot arm that uses sign language could help 70 million deaf people | Metro News", Metro.co.uk, 2018. [Online]. Disponível em: https://metro.co.uk/2018/03/19/robot-armuses-sign-language-help-millions-deaf-people-7398862/. [Acedido a 24 de Outubro de 2018].

[16] "There's not enough sign language translators, so these students 3D printed a humanoid robot.", 3D Hubs Blog, 2018. [Online]. Disponível em: https://www.3dhubs.com/blog/theres-not-enough-sign-language-translators-so-these-students-3d-printed-a-humanoid robot/. [Acedido a 24 de Outubro de 2018].

[17] Associação Portuguesa de Surdos. (2011). Comunidade. Disponível em: http://www.apsurdos.org.pt/index.php?option=com_content&view=article&id=43&Itemid=57. [Acedido a 5 de Setembro de 2018].

[18] Carvalho, P., Breve história dos surdos – no mundo e em Portugal, 1a ed.. Lisboa: Surd'Universo, 2007.

[19] Pinto, C. (2010). Dia nacional da Língua Gestual Portuguesa.. [image] Disponivel em: https://geopalavras.wordpress.com/2010/11/14/dia-nacional-da-lingua-gestual-portuguesa/ [Acedido a 5 Jul. 2019].

[20] Amaral, M. A., Coutinho, A., & Martins, M. R. D., Para uma Gramática da Língua Gestual Portuguesa. Lisboa: Editorial Caminho, SA., 1994.

[21] D. Capovilla, "Alunos Surdos: Mais-estudante", Mais-estudante.webnode.pt, 2018. [Online]. Disponível em: https://mais-estudante.webnode.pt/alunos-surdos/. [Acedido a 24 de Outubro de 2018].

[22] "18 Extreme All-terrain Robots", Into Robotics, 2018. [Online]. Disponível em: https://www.intorobotics.com/18-extreme-all-terrain-robots/. [Acedido a 24 de Outubro de 2018].

[23] "Could These Swimming Robots Help Local Communities?", iRevolutions, 2018. [Online]. Disponível em: https://irevolutions.org/2016/08/02/swimming-robots/. [Acedido a 24 de Outubro de 2018].

[24] "Drones fazem de tudo; confira lista das tarefas dos 'robôs voadores'", TechTudo, 2018. [Online]. Disponível em: https://www.techtudo.com.br/noticias/noticia/2013/12/drones fazem-de-tudo-confira-lista-das-tarefas-dos-robos-voadores.html. [Acedido a 24 de Outubro de 2018].

[25] S. Bouchard, "Industrial robots: What are the different types?", Blog.robotiq.com, 2018. [Online]. Disponível em: https://blog.robotiq.com/bid/63528/what-are-the-differenttypes-of-industrial-robots. [Acedido a 24 de Outubro de 2018].

[26] "Smart Collaborative Robots for Factory Automation | Rethink Robotics", Rethink Robotics, 2018. [Online]. Disponível em: https://www.rethinkrobotics.com/smart collaborative-difference/. [Acedido a 24 de Outubro de 2018].

[27] "Top 6 Robotic Applications in Medicine", Asme.org, 2018. [Online]. Disponível em: https://www.asme.org/engineering-topics/articles/bioengineering/top-6-robotic applications-in-medicine. [Acedido a 24 de Outubro de 2018].

[28] "Robots in Agriculture", Into Robotics, 2018. [Online]. Disponível em: https://www.intorobotics.co robots-in-agriculture/. [Acedido a 24 de Outubro de 2018].

[29] "ROS.org | History", Ros.org, 2018. [Online]. Disponível em: http://www.ros.org/history/. [Acedido a 24 de Outubro de 2018].

[30] "ROS/Introduction - ROS Wiki", Wiki.ros.org, 2018. [Online]. Disponível em: http://wiki.ros.org/ROS/Introduction. [Acedido a 24 de Outubro de 2018].

[31] "Robot", En.wikipedia.org, 2018. [Online]. Disponível em: https://en.wikipedia.org/wiki/Robot. [Acedido a 24 de Outubro de 2018].

[32] "Robots to get their own operating system | World News", Web.archive.org, 2018. [Online]. Disponível em: https://web.archive.org/web/20090918055715/http://www.ethio-pianreview.com/articles/23156. [Acedido a 24 de Outubro de 2018].

[33] L. Joseph, Mastering ROS for Robotics Programming, Ebook.

[34] "ROS.org | Is ROS For Me?", Ros.org, 2018. [Online]. Disponível em: http://www.ros.org/is-ros-for-me/. [Acedido a 24 de Outubro de 2018].

[35] "The 3-Clause BSD License | Open Source Initiative", Opensource.org, 2018. [Online].

Disponível em: http://opensource.org/licenses/BSD-3-Clause. [Acedido a 24 de Outubro de 2018].

[36] "BSD licenses", En.wikipedia.org, 2018. [Online]. Disponível em: http://en.wikipedia.org/wiki/BSD_licenses. [Acedido a 24 de Outubro de 2018].

[37] J. Carette, "What is ROS and ROS-Industrial for Robots?", Blog.robotiq.com, 2018. [Online]. Disponível em: https://blog.robotiq.com/bid/70845/What-is-ROS-and-ROSIndustrial- for-Robots. [Acedido a 24 de Outubro de 2018].

[38] J. O'Kane, A Gentle Introduction to ROS, 2a edição, Columbia, 2016.

[39] "ROS tutorial #1: Introduction, Installing ROS, and running the Turtlebot simulator.", YouTube, 2018. [Online]. Disponível em: https://www.youtube.com/watch?v=9U6GDonG FHw. [Acedido a 24 de Outubro de 2018].

[40] T. ARTICLES, N. PRODUCTS, G. ELECTRONICS, C. PROJECTS, E. MICRO, V. Lectures, I. Webinars, I. Training, P. Search, T. DB, B. Tool, R. Designs, Y. Tawil e Y. Tawil, "An Introduction to Robot Operating System (ROS)", Allaboutcircuits.com, 2018. [Online]. Disponível em: https://www.allaboutcircuits.com/technical-articles/an-introduction-to-robot-operating-system-ros/. [Acedido a 24 de Outubro de 2018].

[41] Martignoni, N. (2016). Schéma de fonctionnement de ROS. [image] Disponível em: https://commons.w master-node-topic.png [Acedido em14 Jul. 2019].

[42] "Master - ROS Wiki", Wiki.ros.org, 2018. [Online]. Disponível em: http://wiki.ros.org/Master. [Acedido a 24 de Outubro de 2018].

[43] A. Martinez e E. Fernández, Learning ROS for Robotics Programming, Ebook.

[44] "Nodes - ROS Wiki", Wiki.ros.org, 2018. [Online]. Disponível em: http://wiki.ros.org/Nodes. [Acedido a 24 de Outubro de 2018].

[45] "Topics - ROS Wiki", Wiki.ros.org, 2018. [Online]. Disponível em: http://wiki.ros.org/Topics. [Acedido a 24 de Outubro de 2018].

[46] "Messages - ROS Wiki", Wiki.ros.org, 2018. [Online]. Disponível em: http://wiki.ros.org/Messages. [Acedido a 24 de Outubro de 2018].

[47] "Services - ROS Wiki", Wiki.ros.org, 2018. [Online]. Disponível em: http://wiki.ros.org/Services. [Acedido a 24 de Outubro de 2018].

[48] "Parameter Server - ROS Wiki", Wiki.ros.org, 2018. [Online]. Disponível em: http://wiki.ros.org/Parameter%20Server. [Acedido a 24 de Outubro de 2018].

[49] M. Griesser, Robot Simulation and Motion Planning, Thesis, 2014.

[50] "Open Dynamics Engine - home", Ode.org, 2018. [Online]. Disponível em:

http://www.ode.org/. [Acedido a 24 de Outubro de 2018].

[51] "PhysX SDK", NVIDIA Developer, 2018. [Online]. Disponível em: https://developer.nvidia.com/physx-sdk. [Acedido a 24 de Outubro de 2018].

[52] "Robotics simulator", En.wikipedia.org, 2018. [Online]. Disponível em: https://en.wikipedia.org/wiki/Robotics_simulator. [Acedido a 24 de Outubro de 2018].

[53] "Gazebo", Gazebosim.org, 2018. [Online]. Disponível em: http://gazebosim.org/. [Acedido a 24 de Outubro de 2018].

[54] V. Mazzari, "Robotic simulation scenarios with Gazebo and ROS", Génération Robots - Blog, 2018. [Online]. Disponível em: https://www.generationrobots.com/blog/en/2015/02/robotic-simulation-scenarios-with-gazebo-and-ros/. [Acedido a 24 de Outubro de 2018].

[55] "Gazebo: Tutorial : *Plugin*s 101", Gazebosim.org, 2018. [Online]. Disponível em: http://gazebosim.org/tutorials/?tut=*plugin*s_hello_world. [Acedido a 24 de Outubro de 2018].

[56] "urdf/XML - ROS Wiki", Wiki.ros.org, 2018. [Online]. Disponível em: http://wiki.ros.org/urdf/XML. [Acedido a 24 de Outubro de 2018].

[57] "urdf/Tutorials/Create your own urdf file - ROS Wiki", Wiki.ros.org, 2018. [Online]. Disponível em: http://wiki.ros.org/urdf/Tutorials/Create%20your%20own%20urdf%20file. [Acedido a 24 de Outubro de 2018].

[58] "Gazebo: Tutorial : URDF in Gazebo", Gazebosim.org, 2018. [Online]. Disponível em: http://gazebosim.org/tutorials/?tut=ros_urdf. [Acedido a 24 de Outubro de 2018].

[59] "urdf/XML/link - ROSWiki", Wiki.ros.org, 2018. [Online]. Disponível em: http://wiki.ros.org/urdf/XML/link. [Acedido a 24 de Outubro de 2018].

[60] "urdf/XML/*joint* - ROSWiki",Wiki.ros.org, 2018. [Online]. Disponível em: http://wiki.ros.org/urdf/XML/*joint*. [Acedido a 24 de Outubro de 2018].

[61] "Xacro - ROS Wiki", Wiki.ros.org, 2018. [Online]. Disponível em: http://wiki.ros.org/xacro. [Acedido a 24 de Outubro de 2018].

[62] Edwards, M. (2017). Axis of Rotation. [online] Acefitness.org. Disponível em: https://www.acefitness. certifications/ace-answers/exam-preparation-blog/3625/axis-of-rotation [Acedido em 26 Abril de 2019].

[63] Ficheiro:Articulações da mão.jpg. (2013). [image] Disponível em : https://pt.wikipedia.org/wiki/Fiche da_mão.jpg [Acedido a 11 Agosto 2019].

[64] Quinn, E. (2019). See the Generally Accepted Values for Normal Range of Motion (ROM). [online] Verywell Health. Disponível em: https://www.verywellhealth.com/what-is- normal-range-of-motion-in-

a-*joint*-3120361 [Acedido em26 Aug. 2019].

[65] Medlej, J. (2014). Human Anatomy Fundamentals: Flexibility and *Joint* Limitations.

[online] Design & Illustration Envato Tuts+. Disponível em : https://design.tutsplus.com/articles/human-anatomy-fundamentals-flexibility-and-*joint*-limitations–vector-25401 [Acedido em 26 Agosto 2019].

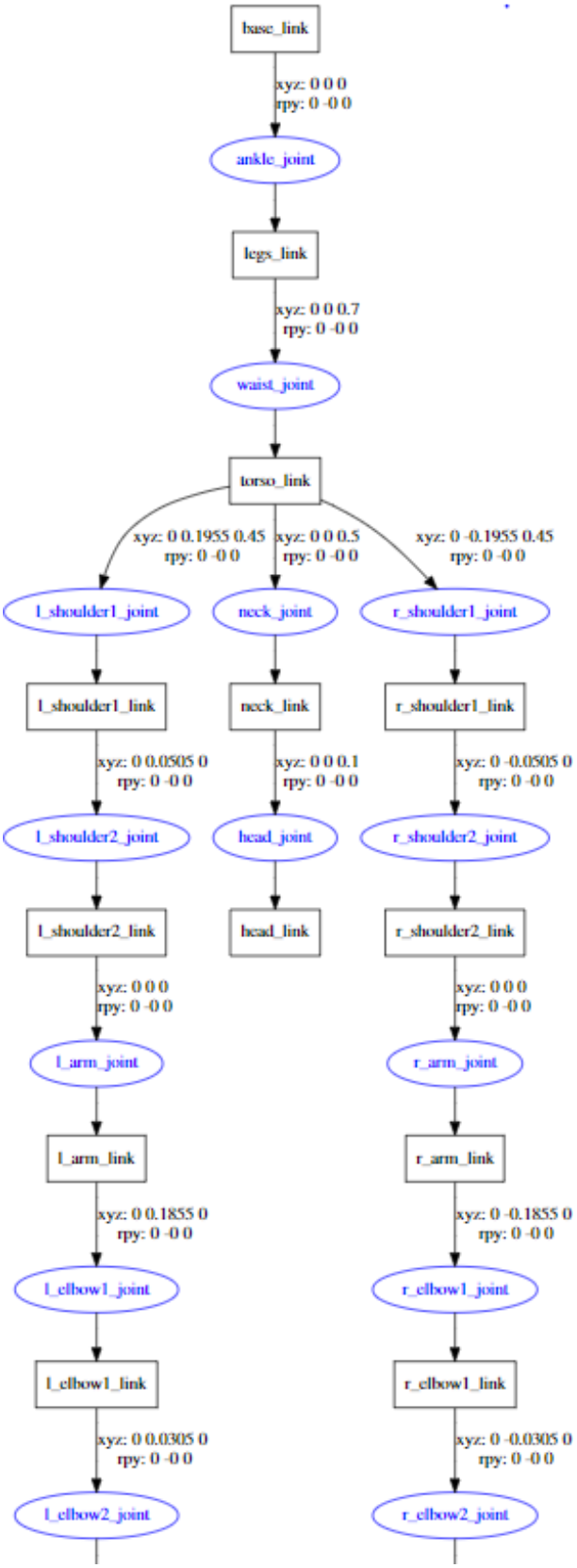
[66] Hamilton, N., Weimar, W. and Luttgens, K. (2012). Kinesiology. 12th ed. New York (NY): McGraw-Hill. [67] Noergaard, K. (2019). mongodb_store - ROS Wiki. [online] Wiki.ros.org. Disponível em: http://wiki.ros.org/mongodb_store [Acedido em 17 Maio 2019].

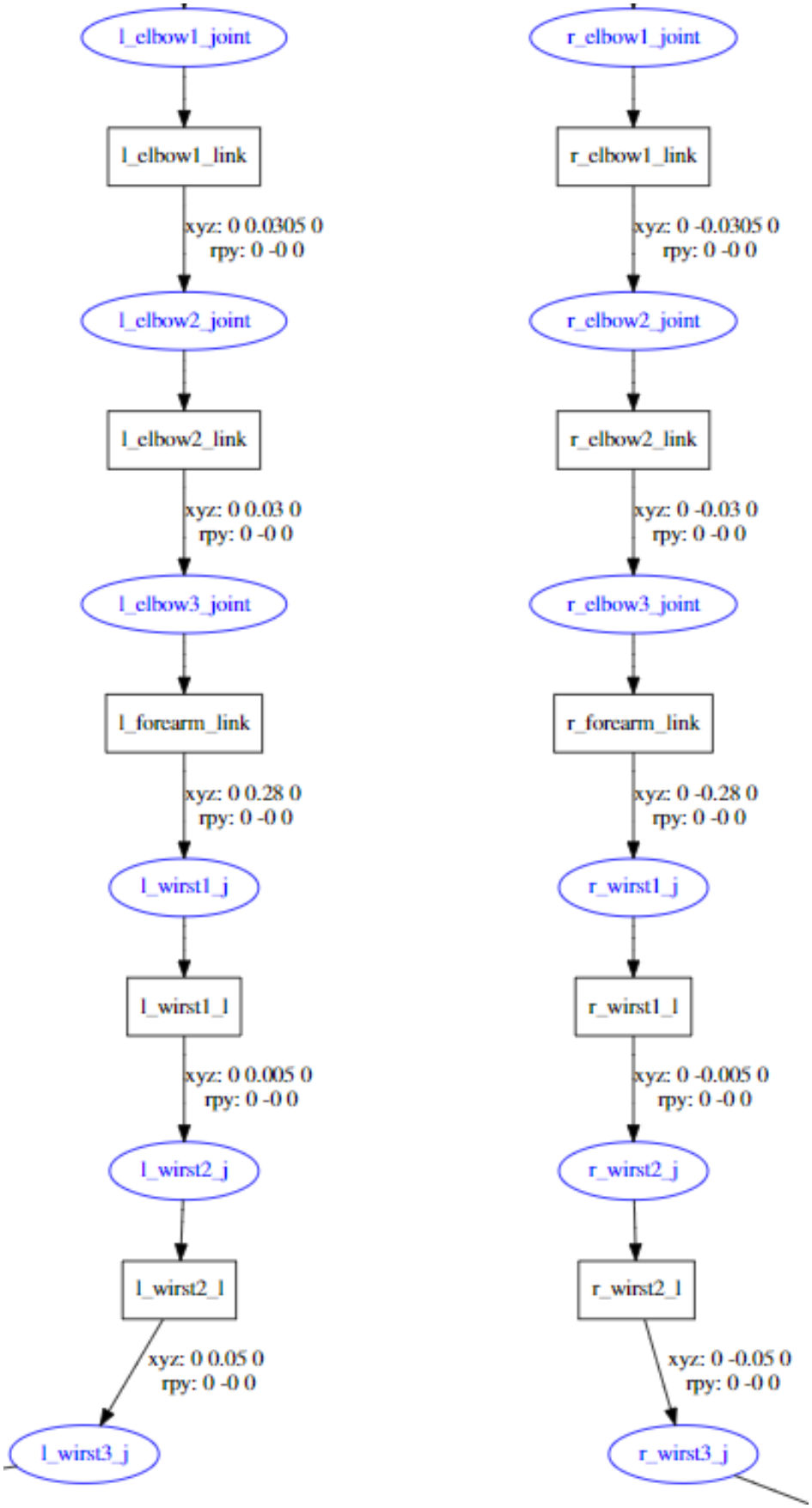
[68] Conley, K. (2011). trajectory_msgs - ROS Wiki. [online] Wiki.ros.org. Disponível em: http://wiki.ros.org/trajectory_msgs [Acedido em18 Janeiro 2019].

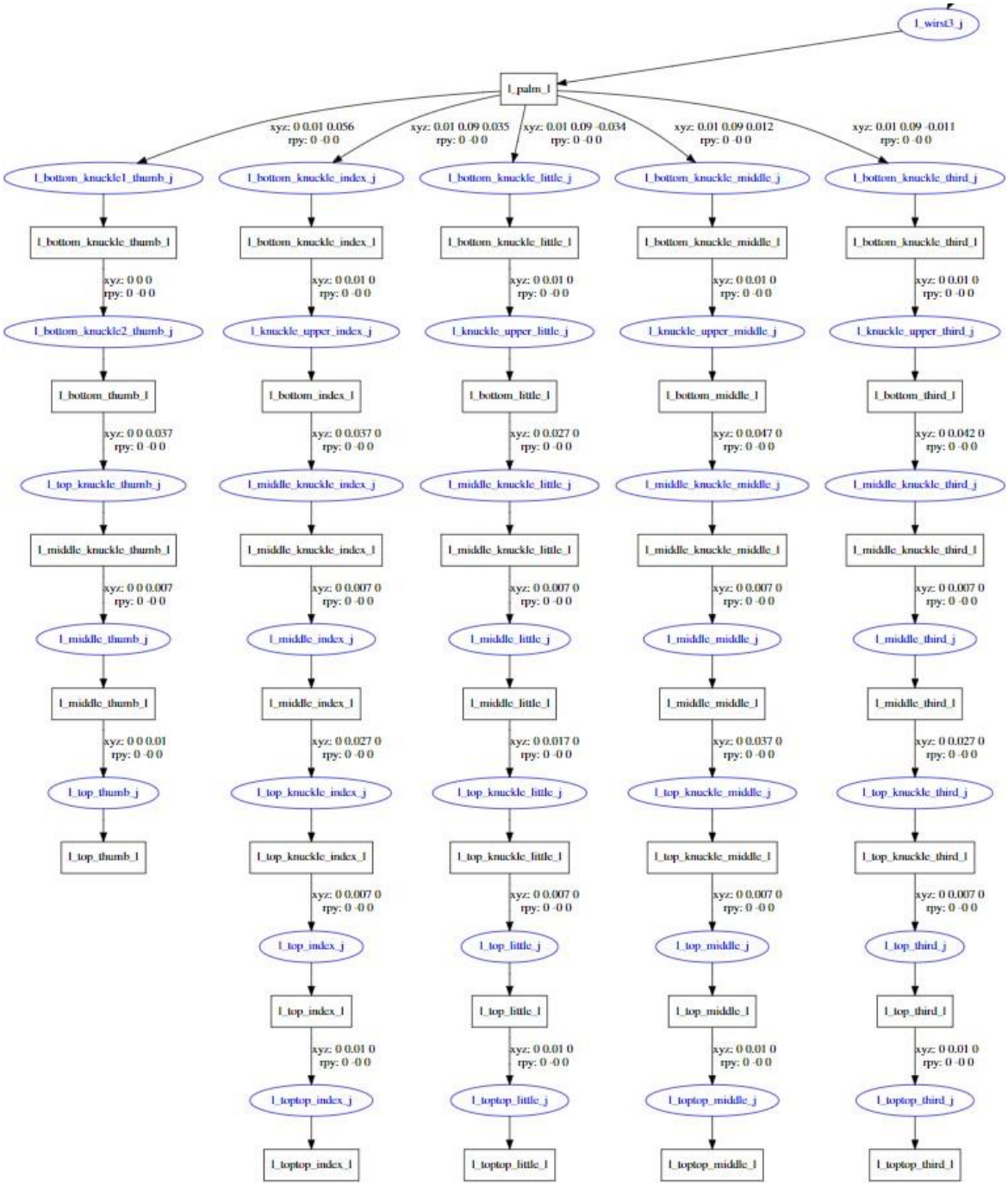
[69] Docs.ros.org. (2019). trajectory_msgs/*Joint*Trajectory Documentation. [online] Disponível em: http://docs.ros.org/melodic/api/trajectory_msgs/html/msg/*Joint*Trajectory.html [Acedido em18 Janeiro 2019].

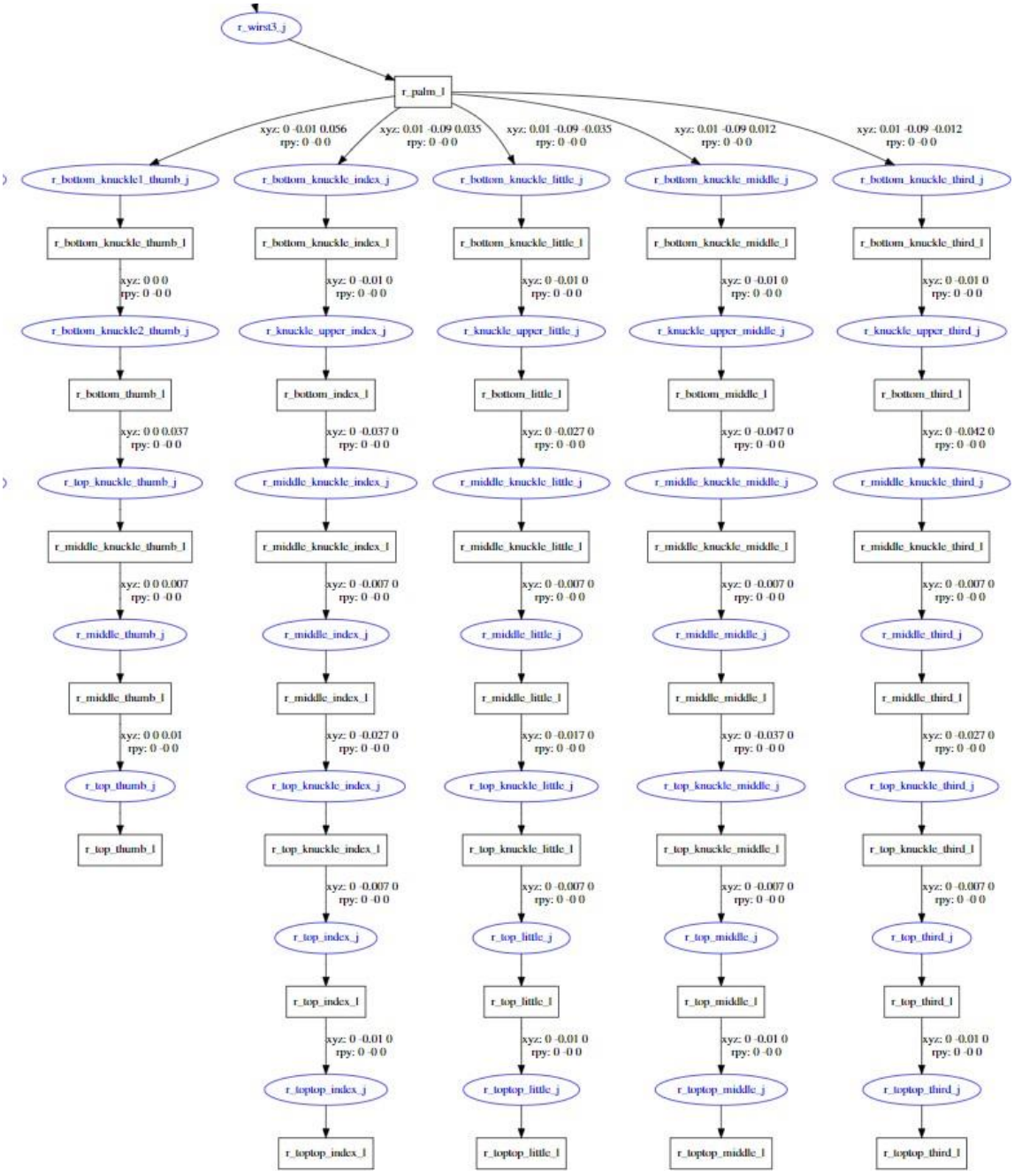
Anexos

A.1 Estrutura do modelo robótico



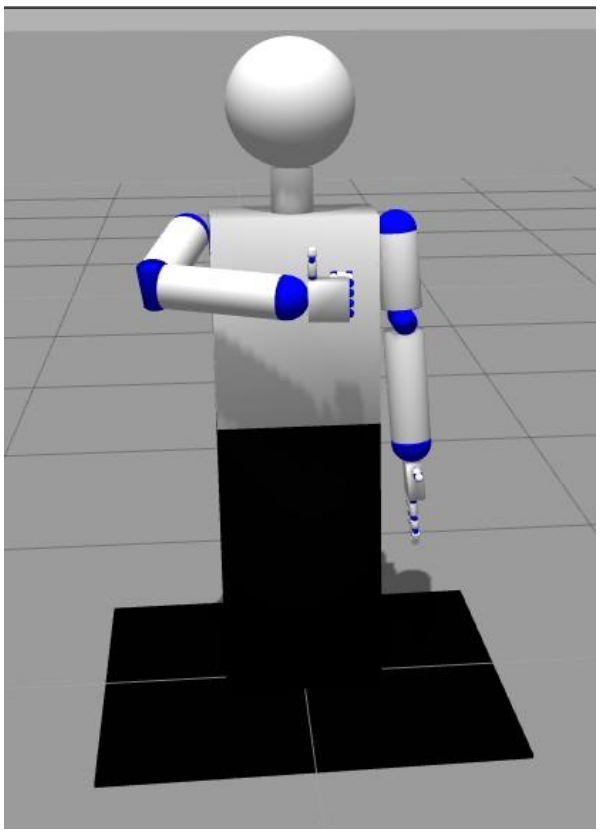

l_elbow1_joint
l_elbow1_link
xyz: 0 0.0305 0
rpy: 0 -0 0
l_elbow2_joint
l_elbow2_link
xyz: 0 0.03 0
rpy: 0 -0 0
l_elbow3_joint
l_forearm_link
xyz: 0 0.28 0
rpy: 0 -0 0
l_wirst1_j
l_wirst1_l
xyz: 0 0.005 0
rpy: 0 -0 0
l_wirst2_j
l_wirst2_l
xyz: 0 0.05 0
rpy: 0 -0 0
l_wirst3_j
r_elbow1_joint
r_elbow1_link
xyz: 0 -0.0305 0
rpy: 0 -0 0
r_elbow2_joint
r_elbow2_link
xyz: 0 -0.03 0
rpy: 0 -0 0
r_elbow3_joint
r_forearm_link
xyz: 0 -0.28 0
rpy: 0 -0 0
r_wirst1_j
r_wirst1_l
xyz: 0 -0.005 0
rpy: 0 -0 0
r_wirst2_j
r_wirst2_l
xyz: 0 -0.05 0
rpy: 0 -0 0
r_wirst3_j


l_wirst3_j
l_palm_l
xyz: 0 0.01 0.056
rpy: 0 -0 0
xyz: 0.01 0.09 0.035
rpy: 0 -0 0
xyz: 0.01 0.09 -0.034
rpy: 0 -0 0
xyz: 0.01 0.09 0.012
rpy: 0 -0 0
xyz: 0.01 0.09 -0.011
rpy: 0 -0 0
l_bottom_knuckle1_thumb_j
l_bottom_knuckle_index_j
l_bottom_knuckle_little_j
l_bottom_knuckle_middle_j
l_bottom_knuckle_third_j
l_bottom_knuckle_thumb_l
l_bottom_knuckle_index_l
l_bottom_knuckle_little_l
l_bottom_knuckle_middle_l
l_bottom_knuckle_third_l
xyz: 0 0 0
rpy: 0 -0 0
xyz: 0 0.01 0
rpy: 0 -0 0
xyz: 0 0.01 0
rpy: 0 -0 0
xyz: 0 0.01 0
rpy: 0 -0 0
xyz: 0 0.01 0
rpy: 0 -0 0
l_bottom_knuckle2_thumb_j
l_knuckle_upper_index_j
l_knuckle_upper_little_j
l_knuckle_upper_middle_j
l_knuckle_upper_third_j
l_bottom_thumb_l
l_bottom_index_l
l_bottom_little_l
l_bottom_middle_l
l_bottom_third_l
xyz: 0 0 0.037
rpy: 0 -0 0
xyz: 0 0.037 0
rpy: 0 -0 0
xyz: 0 0.027 0
rpy: 0 -0 0
xyz: 0 0.047 0
rpy: 0 -0 0
xyz: 0 0.042 0
rpy: 0 -0 0
l_top_knuckle_thumb_j
l_middle_knuckle_index_j
l_middle_knuckle_little_j
l_middle_knuckle_middle_j
l_middle_knuckle_third_j
l_middle_knuckle_thumb_l
l_middle_knuckle_index_l
l_middle_knuckle_little_l
l_middle_knuckle_middle_l
l_middle_knuckle_third_l
xyz: 0 0 0.007
rpy: 0 -0 0
xyz: 0 0.007 0
rpy: 0 -0 0
xyz: 0 0.007 0
rpy: 0 -0 0
xyz: 0 0.007 0
rpy: 0 -0 0
xyz: 0 0.007 0
rpy: 0 -0 0
l_middle_thumb_j
l_middle_index_j
l_middle_little_j
l_middle_middle_j
l_middle_third_j
l_middle_thumb_l
l_middle_index_l
l_middle_little_l
l_middle_middle_l
l_middle_third_l
xyz: 0 0 0.01
rpy: 0 -0 0
xyz: 0 0.027 0
rpy: 0 -0 0
xyz: 0 0.017 0
rpy: 0 -0 0
xyz: 0 0.037 0
rpy: 0 -0 0
xyz: 0 0.027 0
rpy: 0 -0 0
l_top_thumb_j
l_top_knuckle_index_j
l_top_knuckle_little_j
l_top_knuckle_middle_j
l_top_knuckle_third_j
l_top_thumb_l
l_top_knuckle_index_l
l_top_knuckle_little_l
l_top_knuckle_middle_l
l_top_knuckle_third_l
xyz: 0 0.007 0
rpy: 0 -0 0
xyz: 0 0.007 0
rpy: 0 -0 0
xyz: 0 0.007 0
rpy: 0 -0 0
xyz: 0 0.007 0
rpy: 0 -0 0
l_top_index_j
l_top_little_j
l_top_middle_j
l_top_third_j
l_top_index_l
l_top_little_l
l_top_middle_l
l_top_third_l
xyz: 0 0.01 0
rpy: 0 -0 0
xyz: 0 0.01 0
rpy: 0 -0 0
xyz: 0 0.01 0
rpy: 0 -0 0
xyz: 0 0.01 0
rpy: 0 -0 0
l_toptop_index_j
l_toptop_little_j
l_toptop_middle_j
l_toptop_third_j
l_toptop_index_l
l_toptop_little_l
l_toptop_middle_l
l_toptop_third_l


r_wirst3_j
r_palm_l
xyz: 0 -0.01 0.056
rpy: 0 -0 0
xyz: 0.01 -0.09 0.035
rpy: 0 -0 0
xyz: 0.01 -0.09 -0.035
rpy: 0 -0 0
xyz: 0.01 -0.09 0.012
rpy: 0 -0 0
xyz: 0.01 -0.09 -0.012
rpy: 0 -0 0
r_bottom_knuckle1_thumb_j
r_bottom_knuckle_index_j
r_bottom_knuckle_little_j
r_bottom_knuckle_middle_j
r_bottom_knuckle_third_j
r_bottom_knuckle_thumb_l
r_bottom_knuckle_index_l
r_bottom_knuckle_little_l
r_bottom_knuckle_middle_l
r_bottom_knuckle_third_l
xyz: 0 0 0
rpy: 0 -0 0
xyz: 0 -0.01 0
rpy: 0 -0 0
xyz: 0 -0.01 0
rpy: 0 -0 0
xyz: 0 -0.01 0
rpy: 0 -0 0
xyz: 0 -0.01 0
rpy: 0 -0 0
r_bottom_knuckle2_thumb_j
r_knuckle_upper_index_j
r_knuckle_upper_little_j
r_knuckle_upper_middle_j
r_knuckle_upper_third_j
r_bottom_thumb_l
r_bottom_index_l
r_bottom_little_l
r_bottom_middle_l
r_bottom_third_l
xyz: 0 0 0.037
rpy: 0 -0 0
xyz: 0 -0.037 0
rpy: 0 -0 0
xyz: 0 -0.027 0
rpy: 0 -0 0
xyz: 0 -0.047 0
rpy: 0 -0 0
xyz: 0 -0.042 0
rpy: 0 -0 0
r_top_knuckle_thumb_j
r_middle_knuckle_index_j
r_middle_knuckle_little_j
r_middle_knuckle_middle_j
r_middle_knuckle_third_j
r_middle_knuckle_thumb_l
r_middle_knuckle_index_l
r_middle_knuckle_little_l
r_middle_knuckle_middle_l
r_middle_knuckle_third_l
xyz: 0 0 0.007
rpy: 0 -0 0
xyz: 0 -0.007 0
rpy: 0 -0 0
xyz: 0 -0.007 0
rpy: 0 -0 0
xyz: 0 -0.007 0
rpy: 0 -0 0
xyz: 0 -0.007 0
rpy: 0 -0 0
r_middle_thumb_j
r_middle_index_j
r_middle_little_j
r_middle_middle_j
r_middle_third_j
r_middle_thumb_l
r_middle_index_l
r_middle_little_l
r_middle_middle_l
r_middle_third_l
xyz: 0 0 0.01
rpy: 0 -0 0
xyz: 0 -0.027 0
rpy: 0 -0 0
xyz: 0 -0.017 0
rpy: 0 -0 0
xyz: 0 -0.037 0
rpy: 0 -0 0
xyz: 0 -0.027 0
rpy: 0 -0 0
r_top_thumb_j
r_top_knuckle_index_j
r_top_knuckle_little_j
r_top_knuckle_middle_j
r_top_knuckle_third_j
r_top_thumb_l
r_top_knuckle_index_l
r_top_knuckle_little_l
r_top_knuckle_middle_l
r_top_knuckle_third_l
xyz: 0 -0.007 0
rpy: 0 -0 0
xyz: 0 -0.007 0
rpy: 0 -0 0
xyz: 0 -0.007 0
rpy: 0 -0 0
xyz: 0 -0.007 0
rpy: 0 -0 0
r_top_index_j
r_top_little_j
r_top_middle_j
r_top_third_j
r_top_index_l
r_top_little_l
r_top_middle_l
r_top_third_l
xyz: 0 -0.01 0
rpy: 0 -0 0
xyz: 0 -0.01 0
rpy: 0 -0 0
xyz: 0 -0.01 0
rpy: 0 -0 0
xyz: 0 -0.01 0
rpy: 0 -0 0
r_toptop_index_j
r_toptop_little_j
r_toptop_middle_j
r_toptop_third_j
r_toptop_index_l
r_toptop_little_l
r_toptop_middle_l
r_toptop_third_l

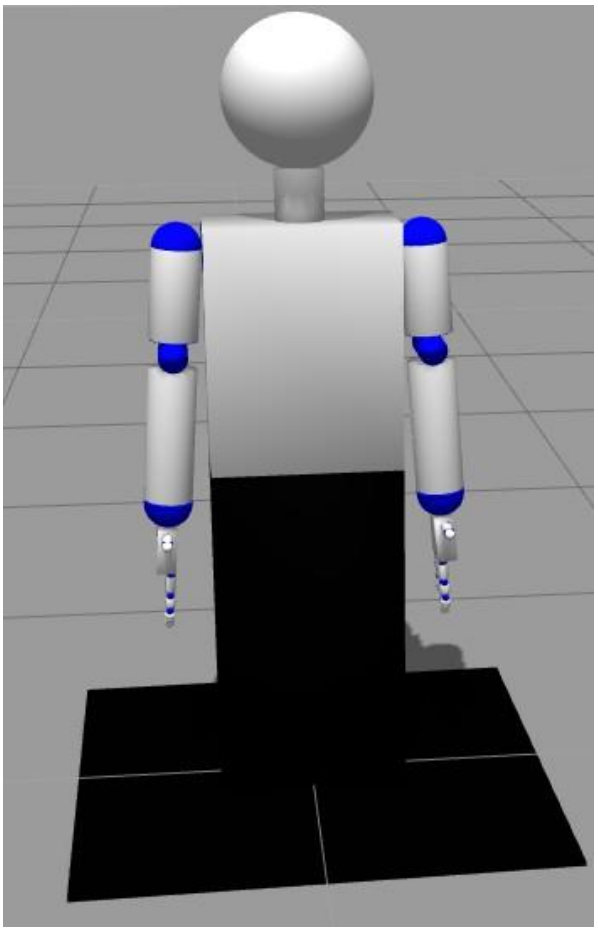
A.2. Especificação dos gestos

1- Letra b



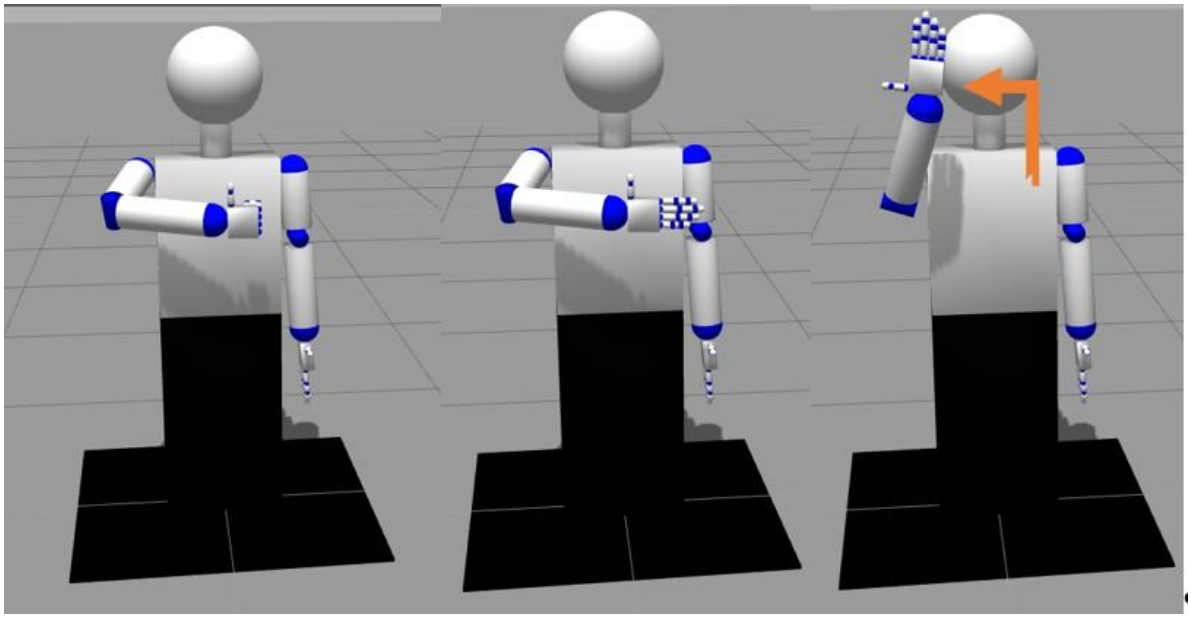
Este gesto é estático. O punho cerrado deve estar no centro do tronco, com o polegar esticado para cima, paralelo à linha do tronco.

2 - Mãos em repouso (posição default)



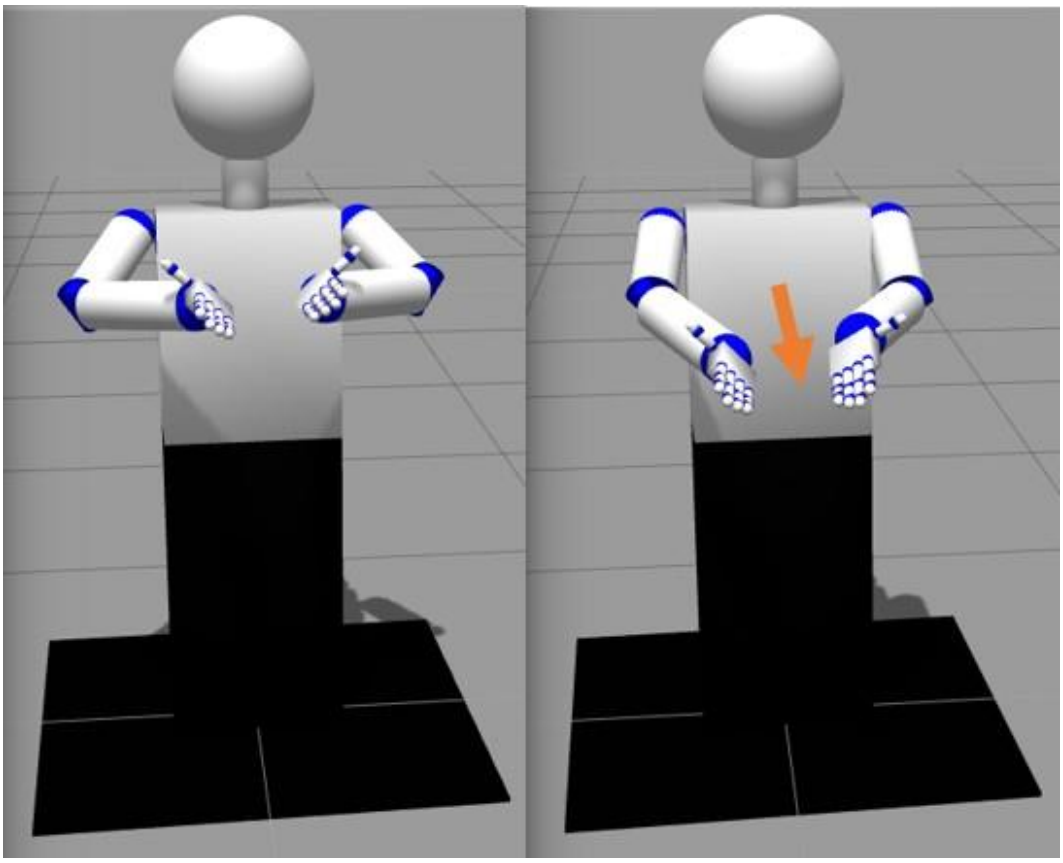
Os braços e as mãos devem estar esticados para baixo, paralelos ao tronco.

3 - "Bom dia"



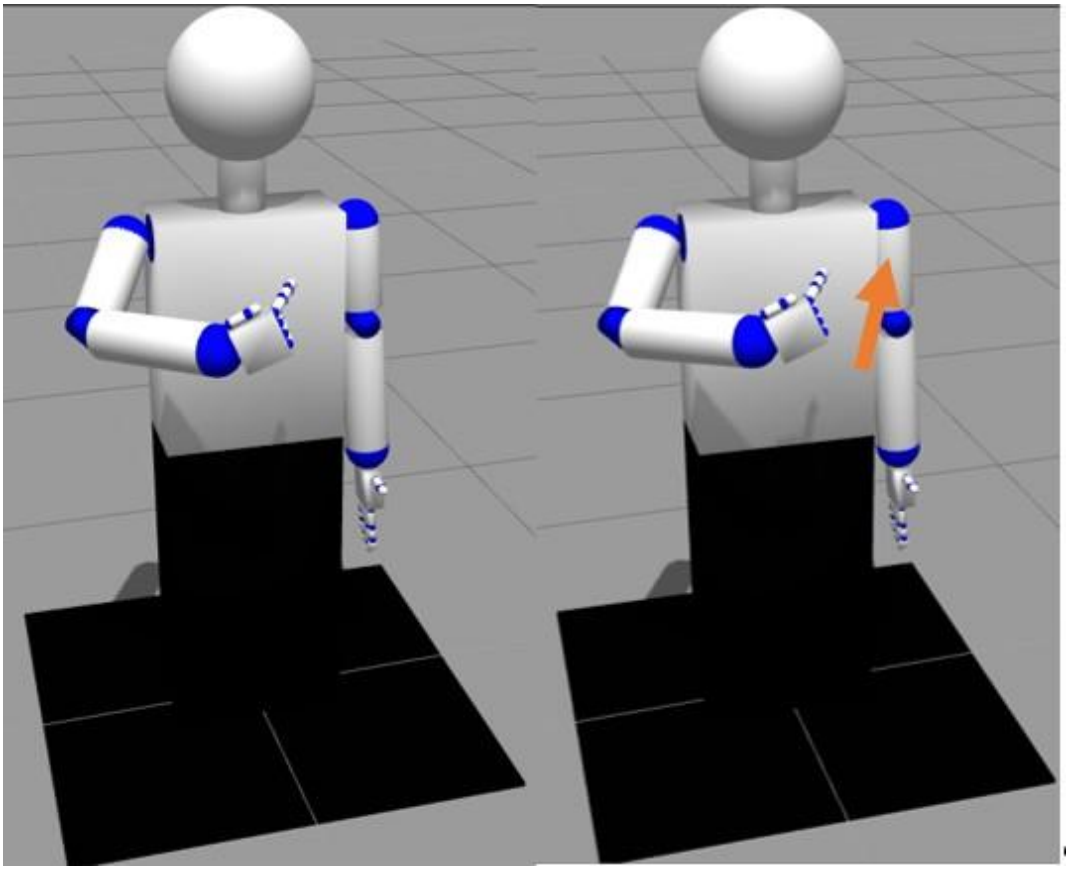
Este gesto é dinâmico, sendo por isso constituído por três partes. Na primeira fase deve-se executar o gesto que representa a letra "b", o punho cerrado deve estar no centro do tronco, com o polegar esticado para cima, paralelo à linha do tronco. Na segunda fase, deve-se esticar todos os dedos da mão. Na terceira e última fase deve-se rodar o braço para cima até perfazer 90 graus.

4 - "Ajudar"



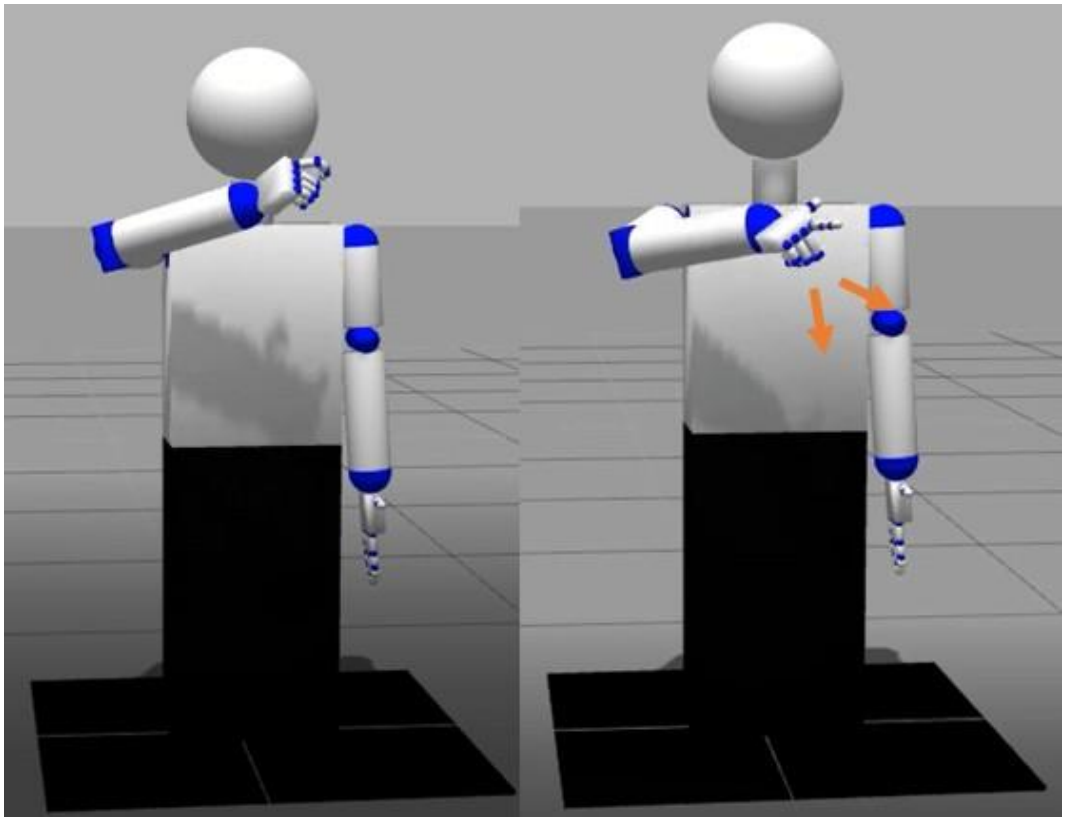
Este gesto é dinâmico, sendo por isso constituído por duas partes. Na primeira fase, ambas as mãos posicionam-se na zona do peito, com os dedos esticados e palmas virada para cima. Na segunda fase, desloca-se ambas as mãos para a frente.

5 - "Eu"



Este gesto é dinâmico, sendo por isso constituído por duas partes. Na primeira fase, a mão posiciona-se na zona do peito, mas afastado do mesmo. A mão está cerrada exceto o dedo indicador, que fica esticado apontando para o peito. Na segunda fase, desloca-se a mão para mais perto do peito.

6 - "Poder"



Este gesto é dinâmico, sendo por isso constituído por duas partes. Na primeira fase, a mão está cerrada e encostada debaixo do queixo. Na segunda fase, a mão desloca-se na direção frente e baixo estando a rodar o punho para a frente, simultaneamente os dedos polegar e indicador desdobram, até ficarem completamente esticados.

A.3. Caraterísticas do computador

Modelo do computador	SATELLITE L755-13W
Modelo do Processador	Intel(R) Core (TM) i5-2410M
Número de Cores	4
Velocidade de Processamento	2.30 GHz
Gráfica	NVIDIA GeForce GT 525M
RAM	4 GB DDR3
Sistema operativo	Linux Ubuntu 14.04 (trusty)
Kernel	4.4.0-141-generic