Capítulo 2

Robôs Socialmente Assistivos: Desenvolvendo Sessões de Terapia Multissensorial com o Robô EVA

Marcelo Marques da Rocha, Sara Luzia de Melo, Jesús Favela e Débora C. Muchaluat Saade

Abstract

Socially Assistive Robots (SARs) have been used to promote social, emotional and cognitive skills. From this perspective, this chapter presents an overview of methodologies and concepts related to SARs applied in the context of healthcare therapy sessions. A SAR called EVA is highlighted. EVA is an open-source robotics platform that has been proposed for therapies for patients with Alzheimer's and children with Autism Spectrum Disorder. In addition, this chapter describes the hardware and software architecture of the EVA robot, its main functionalities and examples of multisensory therapies with the EvaML language, which is based on XML. Therefore, we aim at assisting the development of new multisensory therapy sessions and carrying out simulations using the EvaSIM software.

Resumo

Robôs Socialmente Assistivos (SARs - Socially Assistive Robots) têm sido utilizados para promover habilidades sociais, emocionais ou cognitivas. Nesta perspectiva, este capítulo apresenta uma visão geral das metodologias e conceitos relacionados aos SARs aplicados no contexto de sessões de terapia de saúde. Destaca-se o robô EVA, uma plataforma de robótica open-source proposta para terapias para pacientes com Alzheimer e crianças com Transtorno do Espectro Autista. Além disso, este capítulo descreve a arquitetura de hardware e software do robô EVA, suas principais funcionalidades e exemplos de terapias multissensoriais com a linguagem EvaML, que é baseada em XML. Assim, vislumbra-se auxiliar o desenvolvimento de novas sessões de terapia multissensoriais, bem como realizar simulações utilizando o software EvaSIM.

2.1. Introdução

Tecnologias robóticas têm sido utilizadas em vários ambientes com o objetivo de proporcionar uma melhor qualidade de vida para seus usuários. Assim sendo, robôs são empregados como dispositivos que realizam algum tipo de assistência e, por outro lado, a tecnologia robótica é um método responsável por desenvolver capacidades físicas e cognitivas dos seres humanos. Dessa maneira, há robôs que auxiliam em trabalhos domésticos, e também há aqueles que acompanham e colaboram com pessoas portadoras de algum tipo de deficiência. Contudo, pesquisas têm demostrado novos avanços na tecnologia robótica e no processo de interação humano-robô [Shibata 2004, Shibata 2012].

Robôs Socialmente Assistivos (SARs - *Socially Assistive Robots*) abrangem este novo campo da robótica que compreende uma classe de robôs definida pela interseção entre a classe de robôs assistivos (assistência ao usuário) e a classe de robôs sociais interativos (interação social e não física). Diferente da robótica interativa, que visa estabelecer relações básicas com usuários humanos, a robótica assistiva destaca-se por incluir robôs que proporcionam uma interação social próxima e eficaz a usuários com necessidades especiais, mais especificamente para idosos ou indivíduos com deficiências físicas, cognitivas, emocionais e sociais [Feil-Seifer and Mataric 2005, Tapus et al. 2007].

Como exemplo, SARs possibilitam suporte a pessoas com demência e indivíduos com Transtorno do Espectro Autista (TEA). Além disso, podem atuar como cuidadores juntamente às equipes de saúde. No entanto, segundo [Yang et al. 2018], a robótica social e a robótica médica apresentam grandes desafios nas áreas de desenvolvimento e aplicações. Considerando sessões terapêuticas, julga-se necessário que a interação social seja realizada de maneira personalizada. Para *Alzheimer's Association*[1], a comunicação e interação com pacientes são estratégias eficientes para preservação das habilidades cognitivas. Os trabalhos de [Tapus et al. 2009, Mordoch et al. 2013, Salichs et al. 2016, Woods et al. 2021] apresentaram contribuições na inserção dos SARs em pacientes com doença de Alzheimer. Da mesma forma, em crianças com TEA, para [Duquette et al. 2008], terapias que aumentam a capacidade de imitar ações do robô, ou que despertam o desejo de imitação, estimulam o aprendizado de outras habilidades.

Vale ressaltar que o TEA é uma síndrome do neurodesenvolvimento que apresenta diferentes níveis de intensidade. Esse transtorno é uma condição que compromete fatores cognitivos, em especial, as habilidades sociais, o comportamento e a comunicação [Mesibov et al. 2013]. Entretanto, uma das principais dificuldades enfrentadas pelos indivíduos diagnosticados com TEA é a capacidade da expressão e o reconhecimento das emoções primárias, sendo elas, alegria, tristeza, medo, desgosto, surpresa e raiva [Elder et al. 2006, Annaz et al. 2009].

No âmbito do desenvolvimento das habilidades emocionais em crianças com TEA, SARs têm sido cada vez mais utilizados, principalmente para auxiliar em diagnósticos e tratamentos. Autores de [Mazzei et al. 2011, Cabibihan et al. 2013, Chen et al. 2020, Bartl-Pokorny et al. 2021, Gudlin et al. 2022] constataram efeitos favoráveis nas interações criança-robô e também no reconhecimento de algumas emoções e habilidades emocionais. Assim, a utilização dos SARs é considerada uma solução satisfatória no processo

[1] https://www.alz.org/

de interação personalizada, uma vez que despertam o interesse, atuam como mediadores e motivadores [Barakova and Lourens 2013, Huskens et al. 2013]. Além disso, esses robôs são capazes de monitorar e armazenar dados importantes dos usuários, tais como, nível de atenção, emoções, direção do olhar, dentre outros.

De acordo com [Josué et al. 2020] e [Rocha et al. 2021], a interação multimodal e efeitos sensoriais são relevantes quando adicionados às capacidades dos robôs. Uma integração de efeitos sensoriais de luz na interação humano-robô possibilita produzir terapias imersivas mais significativas, principalmente para crianças. Além da interação por voz, que é frequentemente utilizada por SARs, a interação por vídeo também pode ser inserida com objetivo de contribuir, por exemplo, na captura das emoções expressas pelo usuário.

Nesta perspectiva, o objetivo deste capítulo é apresentar uma visão geral sobre alguns SARs encontrados na literatura e discutir propostas de robôs socialmente assistivos para terapias em saúde. Em seguida, descrever o processo prático, em relação ao desenvolvimento de sessões interativas multissensoriais, para um SAR específico, denominado EVA, que vem sendo proposto para terapias com idosos e crianças. Para isto, serão especificados a arquitetura de hardware e software do robô EVA, suas principais funcionalidades e exemplos de terapias multissensoriais propostas com o robô. O EVA oferece uma linguagem baseada em XML (*eXtensible Markup Language*), chamada EvaML, e um simulador, chamado EvaSIM, para promover o desenvolvimento de novas terapias.

O restante deste capítulo está organizado da seguinte maneira. Na Seção 2.2, são apresentados diferentes modelos e aplicações de SARs em terapias de saúde. O Robô EVA é detalhado na Seção 2.3, bem como suas principais funcionalidades e componentes de hardware e software. O desenvolvimento das sessões de terapia com a linguagem EvaML, os comandos da linguagem e o simulador EvaSIM são descritos na Seção 2.4. Na Seção 2.5, apresenta-se uma atividade prática de como executar scripts EvaML no simulador EvaSIM. E, por fim, na Seção 2.6 as considerações finais do capítulo são apontadas.

2.2. Modelos e Aplicabilidade de SARs

Conforme descrito anteriormente, o conceito de SARs está principalmente relacionado a robôs que fornecem assistência por meio da interação social. Várias abordagens de SARs, que podem ser vistas em [Fasola and Mataric 2012, Scassellati et al. 2012, Robinson et al. 2014, Cruz-Sandoval et al. 2020], são utilizadas para fornecer recursos responsáveis por aprimorar os cuidados e a qualidade de vida dos pacientes diagnosticados com Alzheimer e TEA. Considerando a aparência física do robô, segundo [Martinez-Martin et al. 2020], os robôs possuem as seguintes classes:

- **androide**: robôs parecidos fisicamente com humanos;
- **mascote**: possui formas humanoides, mas aparências abstratas ou caricaturais;
- **mecânico**: formas humanoides com partes visivelmente mecânicas;
- **animais**: aparência física similar a animais de estimação;
- **não humanoide**: não há semelhança com nenhum ser vivo.

Ainda no trabalho de [Martinez-Martin et al. 2020], o desenvolvimento de qualquer produto com a finalidade de auxiliar indivíduos com demência, julga-se necessário não apenas observar os aspectos tecnológicos, mas sim efetuar uma análise apropriada dos fatores que torne o projeto mais inclusivo e humano. Neste contexto, um projeto de SARs responsável por auxiliar no tratamento de demência, os autores sugerem que robôs, além da interação personalizada, sejam preparados para redução de estímulos desnecessários durante a sessão de terapia, tais como ruídos e brilhos excessivos.

Todavia, conforme exposto em [Fong et al. 2003, Matarić et al. 2007], os principais desafios encontrados no desenvolvimento de SARs são referentes à definição do comportamento social dos robôs, à execução de diferentes tarefas, à elaboração de uma comunicação apropriada aos usuários e à adaptação personalizada ao ambiente. Para projetar adequadamente robôs sociais, o trabalho de [Nestorov et al. 2014] apresentou uma taxonomia com recursos necessários no projeto de robôs que auxiliam no cuidado de pessoas com demência. Resumidamente, na taxonomia exposta em [Nestorov et al. 2014], foram definidas cinco principais categorias que devem ser avaliadas, sendo elas:

- **aparência**: a escolha da aparência física do robô conforme sua aplicação;
- **modalidade de interação**: conter múltiplas modalidades interativas;
- **interação inteligente**: comportamento adaptativo e específico do robô;
- **capacidade de tarefa**: assistência emergencial e suporte de tarefas;
- **modo de operação**: capacidade de adaptação dos robôs às mudanças do ambiente e necessidades do usuário.

Em terapias robóticas para o tratamento de indivíduos com TEA, segundo [Ricks and Colton 2010], um robô com forma humana é considerado o mais adequado na aplicabilidade em sessões terapêuticas. Isso se deve ao fato que uma interação com robôs humanoides possibilita estabelecer o envolvimento de crianças com autismo no processo de reproduzir os estados emocionais, bem como o reconhecimento das emoções e, consequentemente, promove o ensino das habilidades emocionais. Entretanto, de acordo com [Costa et al. 2017, Dickstein-Fischer et al. 2018], recomenda-se que o tamanho do robô seja apropriado ao tamanho da criança com objetivo de facilitar o contato visual e, além disso, as expressões faciais dos SARs não devem ser intimidantes durante o processo de intervenções com autismo.

Ao observar as características supracitadas sobre projeto de robôs sociais, constata-se que para utilização dos SARs em sessões de terapias, além de considerar o *design* do robô conforme seu contexto de aplicação, também deverão ser pesquisados, validados e acrescentados ao projeto fatores relacionados à adaptabilidade ao ambiente e/ou interação, comportamentos do robô e diversas modalidades com intuito de proporcionar uma terapia robótica eficiente e personalizada. A seguir são apresentados brevemente alguns robôs aplicados em sessões terapêuticas, tanto comerciais quanto acadêmicos, encontrados na literatura revisada.

2.2.1. Robôs Comerciais

Comumente utilizado em sessões de terapias para promover o ensino das habilidades emocionais, o robô denominado NAO é um robô humanoide da *Softbank* [2]. O NAO é uma ferramenta de programação que tornou-se um padrão em educação e pesquisa, também é utilizado como assistente por empresas e centros de saúde para receber, informar e proporcionar entretenimento aos visitantes. Em [Valentí Soler et al. 2015] um estudo foi conduzido com o objetivo de testar o efeito da introdução do robô NAO em sessões terapêuticas para pacientes com demência em relação às mudanças de comportamento, apatia e qualidade de vida. Na Figura 2.1, apresenta-se o robô NAO, exposto no trabalho de [Erden 2013], utilizado para efetuar o desenvolvimento emocional das posturas do robô relacionadas com três emoções básicas.

Figura 2.1. Robô NAO apresentado no trabalho de [Erden 2013]

Nota-se, na Figura 2.1, que a aparência do robô NAO é bastante amigável. A capacidade do NAO de enfatizar sua visão, compreender o ambiente, a mudança da cor dos olhos e a imitação dos gestos humanos permitem tornar uma sessão terapêutica mais surpreendente [Shamsuddin et al. 2012, Lytridis et al. 2018]. Segundo [Ueyama 2015, Amanatiadis et al. 2017], robôs humanoides, como o NAO, demonstraram eficácia no tratamento de crianças diagnosticadas com TEA.

A Figura 2.2 apresenta um outro robô, denominado Moxie, da empresa norte-americana *Embodied*[3]. Paolo Pirjanian, um ex-cientista da NASA e fundador da empresa *Embodied*, afirma que o robô Moxie foi projetado para estabelecer um relacionamento com crianças e, por meio dessa interação é possível efetuar o reconhecimento e o ensino de diferentes emoções. Segundo [Xiao et al. 2020, Cano et al. 2021], a interação de crianças com o Moxie desenvolve habilidades essenciais, como regulação emocional e gestão de relacionamentos, especialmente em crianças com autismo.

O QTrobot, exposto na Figura 2.3, foi projetado pela LuxAI[4]. Esse robô social é utilizado na educação especial, possibilitando uma interação ativa e comprometida entre crianças e seus pais/cuidadores. Desse modo, o QTrobot fornece uma configuração amigável e eficaz responsável por manter a atenção das crianças e promover o ensino de várias habilidades. O trabalho de [Costa et al. 2017] utilizou o QTRobot para proporcionar o treinamento das habilidades emocionais em crianças com TEA. Nesse estudo,

[2]https://www.softbankrobotics.com/emea/en/nao
[3]https://embodied.com/
[4]https://luxai.com/

Figura 2.2. Aparência física do robô Moxie. Fonte: Embodied - https://embodied.com/

o treinamento foi desenvolvido para adaptar-se ao nível de desenvolvimento dos usuários. Para estimular o ensino das emoções, termos simples foram aplicados e o nível de dificuldade foi acrescentado ao longo das sessões, tornando-o adaptativo ao ambiente.

Figura 2.3. QTRobot utilizado no trabalho de [Costa et al. 2017]

Outro robô humanoide desenvolvido pela *Softbank Robotics*, denominado de Pepper, é exposto na Figura 2.4. O Pepper, conforme detalhado em [Martinez-Martin et al. 2020], possui quase as mesmas articulações de um humano, exceto por sua base móvel. Além disso, contém quatro microfones, dois alto-falantes, duas câmeras RGB e um sensor de profundidade. Considerando promover habilidades sociais, o robô Pepper foi utilizado no trabalho de [Yabuki and Sumi 2018] com objetivo de desenvolver um sistema de apoio à aprendizagem para comunicação e interação personalizada ao diálogo com indivíduos com TEA. Da mesma maneira, o trabalho de [Azuar et al. 2019] utilizou o robô Pepper para modificar a evolução de uma história contada considerando as emoções experimentadas. Para isso, o robô utilizou a emoção expressa pelo usuário, reconhecida por meio de expressões faciais.

Conforme definido em [Shibata 2004], existem quatro categorias relacionadas a aparência do robô: tipo humano, tipo animal familiar, tipo animal desconhecido e tipo animal imaginário/novo tipo de personagem. Projetado especificamente para fins terapêuticos, o robô Paro (*The Seal Robot*), exposto na Figura 2.5, possui uma aparência semelhante a uma foca bebê. Isso faz com que as pessoas possam aceitar o Paro facilmente, sem preconceitos, em sessões terapêuticas [Wada et al. 2008].

A utilização desse robô possibilita trabalhar com os quatro sentidos primários, ou seja, visão (sensor de luz), audição (determinação da direção da fonte sonora e reconhecimento de fala), equilíbrio e o sentido tátil. Dessa maneira, o Paro tem sido inserido em terapias para pacientes com demência. Os resultados da análise feita em [Shibata 2012],

Figura 2.4. Robô Pepper apresentando no trabalho de [Yabuki and Sumi 2018]

Figura 2.5. Características do Paro. Fonte: [Shibata 2012]

relacionados a utilização do robô Paro, constataram que a interação humano-robô estimula o lobo frontal do cérebro de idosos com demência, bem como a alteração dos seus estados emocionais.

2.2.2. Robôs Acadêmicos

Para que se mantenha a percepção das habilidades compreendidas nas sessões de terapia, julga-se necessário que o processo terapêutico seja continuado em casa. Assim, SARs devem ser projetados em tamanhos menores, possuir baixo custo e altos níveis de autonomia [Cao et al. 2015]. Considerando que robôs utilizados em terapia do autismo devem ser simples, o Probolino, apresentado na Figura 2.6, é um dispositivo social robótico portátil de baixo custo com interação através do toque. Conforme descrito em [Saldien et al. 2010], o Probolino parece com um bicho de pelúcia, possui orelhas animadas, olhos, sobrancelhas, pálpebras, boca e pescoço. Uma tela sensível ao toque é localizada em sua barriga. Assim sendo, este robô é capaz de se comunicar e interagir com humanos, expressar atenção e emoções através do olhar e expressões faciais.

Um outro robô com aparência de brinquedo é o robô Bliss, que permite apresentar uma expressão facial mais amigável e não ameaçadora comparado aos robôs humanoides [Santatiwongchai et al. 2016]. A aparência física do Bliss, robô utilizado para atividades terapêuticas, pode ser observada na Figura 2.7. Suas sobrancelhas conseguem ser deslocadas em conjunto com o movimento dos olhos para expressar diferentes emoções e, ainda, possui duas rodas que possibilitam a sua movimentação. Além disso, o Bliss é capaz de exibir luzes LED, sons e exibir imagens que podem ser usadas como ferramenta auxiliar em atividades de aprendizagem.

Figura 2.6. Robô Probolino. Fonte: [Vanderborght et al. 2012]

Figura 2.7. Robô Bliss. Fonte: [Attawibulkul et al. 2019]

No estudo de [Santatiwongchai et al. 2016], os autores efetuaram uma análise da utilização do robô Bliss em atividades de interação para terapia do autismo com a finalidade de fornecer uma melhor interação social para crianças com TEA. Diversos foram os resultados da interação devido aos diferentes níveis do autismo. No entanto, os autores sugerem como trabalhos futuros o desenvolvimento de conteúdos adaptados, para crianças em vários níveis de desenvolvimento, que promovam maior grau de interação durante as sessões de terapia.

Por fim, um outro modelo de robô humanoide, porém com aparência mecânica, conhecido por Troy, é exposto na Figura 2.8. O Troy possui a parte superior do corpo aproximadamente do tamanho de uma criança de quatro anos. O robô possui mais de 30 ações pré-programadas disponíveis para o terapeuta aplicar em sessões de terapia [Manohar et al. 2011]. Essas ações são projetadas para completar tarefas que os terapeutas consideram úteis. Contudo, quando um novo protocolo de terapia é desenvolvido, ações específicas necessárias para completar o protocolo são programadas e disponibilizadas para utilização.

2.2.3. Comparação entre SARs

Dentre as características dos robôs discutidos, observa-se que a aparência física do robô desempenha um papel fundamental no projeto de um SAR e no contexto de suas aplicações. Para o desenvolvimento de sessões terapêuticas, segundo [Ricks et al. 2010, Ueyama 2015, Amanatiadis et al. 2017], robôs humanoides são considerados mais eficientes. Entretanto, na literatura revisada ainda não há um consenso em relação à aparência física no projeto de SARs. A Tabela 2.1 exibe uma comparação entre os robôs comentados neste capítulo.

Figura 2.8. Robô Troy. Fonte: [Ricks et al. 2010]

Tabela 2.1. Análise comparativa dos robôs comerciais e acadêmicos

Robô	Uso	Forma	Aplicação em Saúde	Referência
NAO	comercial	androide	Demência e TEA	[Erden 2013]
Moxie	comercial	mascote	Desenvolvimento social e emocional	[https://embodied.com/]
QTrobot	comercial	androide	TEA	[Costa et al. 2017]
Pepper	comercial	androide	TEA	[Yabuki and Sumi 2018]
Paro	comercial	animal	Demência	[Wada et al. 2008]
Probolino	acadêmico	mascote	TEA	[Cao et al. 2015]
Bliss	acadêmico	mascote	TEA	[Attawibulkul et al. 2019]
Troy	acadêmico	mecânico	TEA	[Ricks et al. 2010]
EVA	acadêmico	mascote	Demência e TEA	[Rocha et al. 2021]

Outros modelos de robôs sociais, aparências, tipos, descrições e publicações relacionadas aos robôs especificados podem ser vistos em [Martinez-Martin et al. 2020]. Na próxima seção, é apresentado um breve histórico do robô social EVA utilizado neste trabalho, suas principais características e funcionalidades.

2.3. Robô EVA

O robô EVA (*Embodied Voice Assistant*) é um robô social e foi desenvolvido originalmente por pesquisadores do CICESE (Centro de Investigación Científica y de Educación Superior de Ensenada), em Baja California no México. O EVA foi desenvolvido como parte de uma tese de doutorado [Cruz Sandoval 2020] e depois foi estendido em outros trabalhos [Mitjans 2020] e [Rocha et al. 2021].

2.3.1. Histórico do Robô EVA

O robô EVA, em sua primeira versão, foi feito em papelão, como pode ser visto na Figura 2.9. Todo o software do robô era executado num Raspberry PI 3 e contava com uma caixa de som e uma placa Matrix Voice que possuía oito microfones e uma matriz de 18 LEDs RGB. Na Seção 2.3.2.1 serão apresentados mais detalhes sobre a placa Matrix Voice.

Essa primeira versão do robô contava com um mecanismo de comunicação não verbal específico de robôs baseados em luz LEDs. O EVA realizava seis animações que usavam esses LEDs, sendo que quatro delas representariam emoções (surpresa, alegria,

Figura 2.9. Primeira versão do robô social EVA [Mitjans 2020]

tristeza e raiva), como pode ser visto na Figura 2.10. As duas outras animações eram usadas para indicar outros estados relacionados com a comunicação verbal do robô, como quando ele estava ouvindo ou falando.

A segunda versão do robô EVA foi toda projetada usando o software Blender [5] e teve seu corpo impresso em 3D utilizando-se o filamento PLA[6]. Nessa versão foram adicionados dois mecanismos de comunicação não verbal: os olhos e o movimento da cabeça. Como não se tinha disponível na época um display com as dimensões adequadas, um smartphone foi usado temporariamente como display para o robô. A Figura 2.11(a) mostra a tela do projeto no Blender, a Figura 2.11(b) mostra o projeto da animação dos olhos do robô no software Unity 3D[7] e a Figura 2.11(c) mostra o novo visual do robô, impresso em 3D.

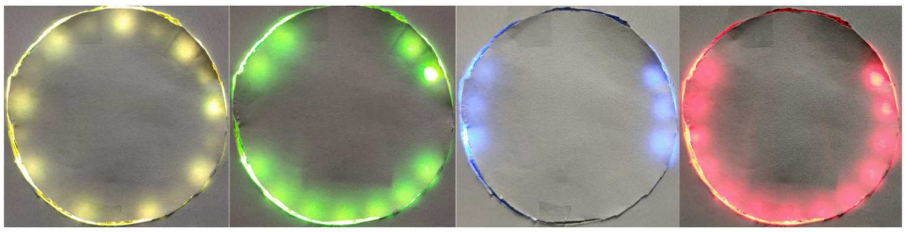

Figura 2.10. Emoções projetadas para o robô EVA (versão 1) [Mitjans 2020]

A princípio foram criadas duas propostas para a síntese de expressões através do olhar do robô, a primeira foi baseada no estudo realizado por [Pollmann et al. 2019] onde se comparou um total de cinco projetos de olhos inspirados em desenhos animados. A segunda proposta de design para os olhos do robô foi baseada no estilo dos quadrinhos japoneses. Após a realização de uma pesquisa informal com algumas pessoas, optou-se por utilizar a segunda proposta, que foi classificada como mais expressiva. Para permitir a movimentação da cabeça do robô foram incorporados ao seu projeto dois

[5]https://www.blender.org/
[6]PLA é um termoplástico biodegradável
[7]https://www.unity.com/

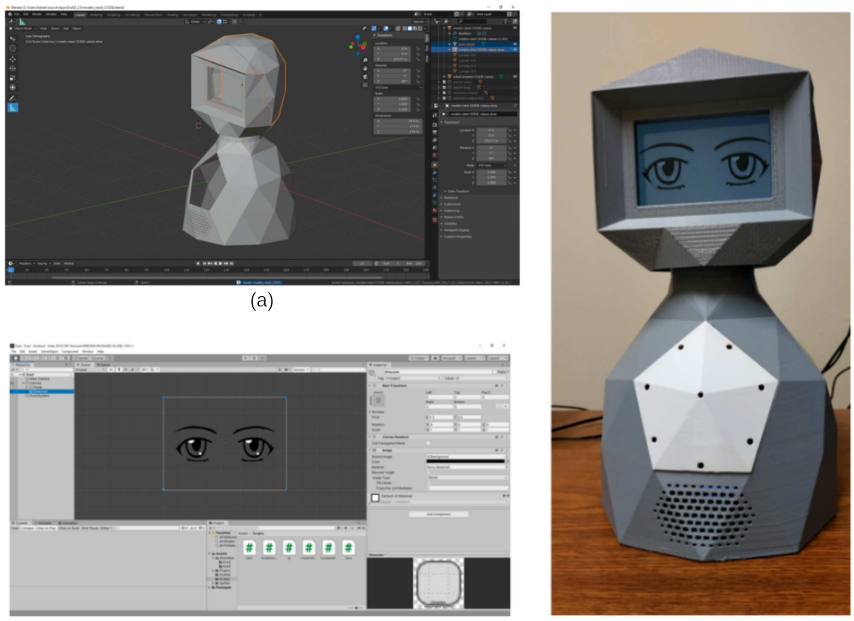

Figura 2.11. (a) Projeto do robô no Blender, (b) projeto dos olhos do robô no Unity 3D e (c) o Robô EVA (versão 2) impresso em 3D [Mitjans 2020]

servomotores. Os movimentos foram projetados levando-se em conta os estudos realizados pelos pesquisadores [Cassell et al. 1994, Pelachaud et al. 1996, Gratch et al. 2006, Li and Chignell 2011, Liu et al. 2012, Marsella et al. 2013].

Na segunda versão do robô, manteve-se a possibilidade do uso dos LEDs. A sua placa de controle principal continuou sendo um Raspberry PI, porém, a versão 3 foi substituída pela versão 4 da mesma.

A terceira versão do EVA foi igualmente projetada no software Blender e posteriormente fabricada em fibra de vidro. Ela herda todas as características funcionais e de hardware da versão anterior. Sua principal característica em relação às versões anteriores está no seu design antropomórfico, apresentando pernas e braços. A Figura 2.12 mostra a versão três do robô EVA fabricada em fibra de vidro.

A versão atual do robô EVA é uma plataforma de robótica *open-source* destinada a auxiliar como ferramenta de apoio à pesquisa em Interação Humano-Robô (IHR). Nessa versão, o corpo do robô EVA voltou a ser impresso em 3D usando um termoplástico biodegradável (PLA) de origem natural. A impressão da estrutura física do EVA utilizou os arquivos com os modelos disponibilizados pelos criadores do robô. Todas as informações necessárias para a montagem do robô EVA encontram-se em seu repositório disponível na web[8]. O software de controle do robô foi construído usando a pilha de software MEAN (*MongoDB*, *Express*, *AngularJS* e *NodeJS*). Os detalhes dos componentes do EVA são apresentados a seguir.

[8]https://github.com/eva-social-robot

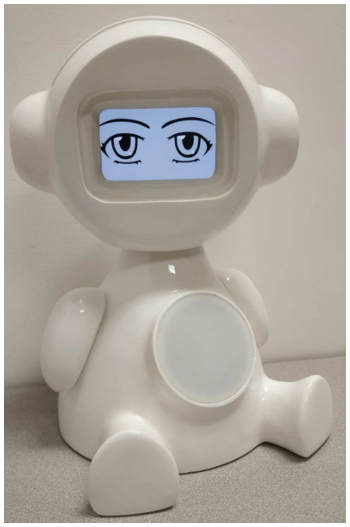

Figura 2.12. Robô EVA (versão 3) fabricado em fibra de vidro [Mitjans 2020]

2.3.2. Arquitetura do Robô

2.3.2.1. Componentes de Hardware

Em relação aos componentes de hardware, a Figura 2.13 exibe a versão básica do robô EVA e seus principais componentes. Esta versão do robô inclui uma interface por voz, uma tela sensível ao toque, um anel de LEDs RGB integrado em seu tórax e servomotores que permitem que sua cabeça se mova com dois graus de liberdade. Os novos elementos, webcam e a lâmpada inteligente, que foram adicionados ao robô em [Rocha et al. 2021] também podem ser vistos na Figura 2.13. Em seguida, será descrito brevemente cada componente de hardware com a sua funcionalidade.

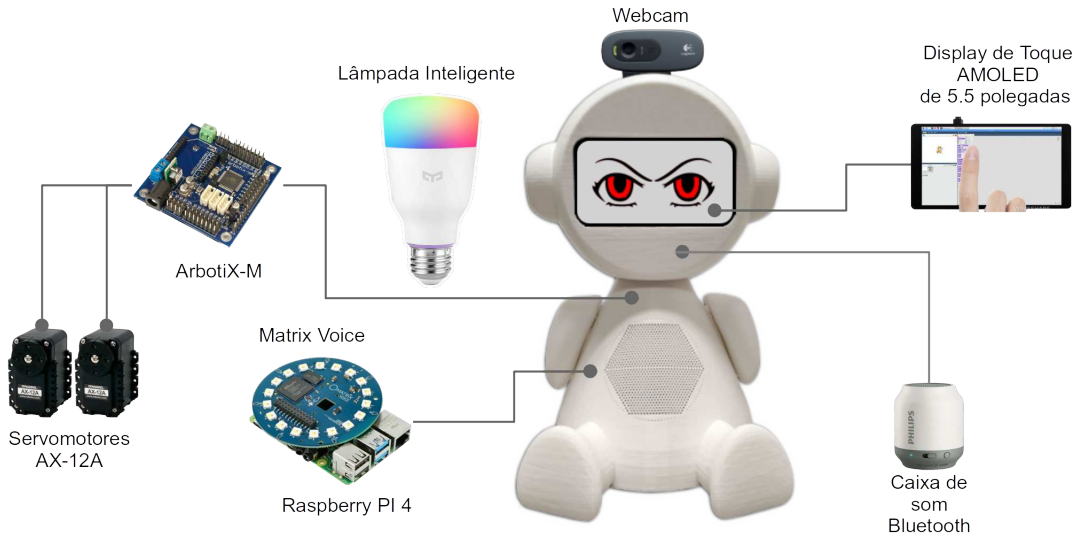

Figura 2.13. Componentes de hardware do robô EVA

- *Tela de Toque*: O EVA tem uma tela de 5.5 polegadas que dá ao robô a capacidade

de sintetizar expressões faciais, sendo elas: *neutra*, *raiva*, *tristeza* e *alegria*.

- *Raspberry PI 4*: Esta é a placa principal do robô, que roda a distribuição Linux para ARMv6 (Raspbian). As demais placas que complementam as funcionalidades do EVA são conectadas ao Raspberry.
- *ArbotiX-M*: Os dois servomotores que são responsáveis pelo movimento da cabeça do robô são controlados por este dispositivo[9]. A ArbotiX-M é uma placa baseada em Arduino e pode ser programada através da IDE do Arduino.
- *Dois Servomotores AX12-A*: O robô usa dois servomotores Dynamixel AX-12A[10] que fornecem 2 graus de liberdade (DoF - *Degree of Freedom*) para a cabeça do EVA.
- *Caixa de Som Bluetooth*: O robô pode falar, usando recursos de *Text-To-Speech*[11] (TTS), pode emitir sons e tocar música usando uma caixa de som *Bluetooth*.
- *Matrix Voice*: Esta placa[12] tem duas funções distintas no projeto do robô EVA. A primeira é funcionar como dispositivo de captura de áudio, captando a voz do usuário, que futuramente será convertida para texto. Para o processo de captura de áudio, ela conta com oito microfones integrados. A placa também é usada como elemento de comunicação não verbal. Algumas animações com cores específicas são executadas usando a matriz de dezoito LEDs RGB da placa.
- *Webcam*: Por meio de uma *webcam*, o robô pode reconhecer as expressões faciais do usuário. O módulo de reconhecimento facial pode retornar as seguintes expressões: "NEUTRAL", "ANGRY", "DISGUST", "FEAR", "SURPRISE", "HAPPY" e "SAD".
- *Lâmpada Inteligente*: Para tornar as sessões de terapia mais atrativas, principalmente para as crianças, mais uma extensão foi adicionada ao robô EVA. Essa extensão dá ao robô a capacidade de controlar uma lâmpada inteligente Xiaomi. O EVA pode se comunicar com a lâmpada via rede sem fio, podendo, ligá-la, desligá-la e definir suas cores.

Como dito anteriormente o robô EVA foi criado por pesquisadores do CICESE, no México. Vários trabalhos usando essas e outras versões intermediárias do robô (mesmo protótipos mais simples) têm sido publicados. Em [Cruz-Sandoval and Favela 2019], os criadores do EVA apresentam os resultados de um estudo utilizando o robô na condução de uma sessão de terapia não farmacológica com pacientes idosos com demência. A Figura 2.14 apresenta uma sessão de terapia de estimulação cognitiva com três pacientes em uma casa de repouso para idosos.

A Tabela 2.2 apresenta os materiais, componentes de hardware, sensores, atuadores e recursos de comunicação verbal e não verbal de cada uma das quatro versões do robô

[9]https://www.trossenrobotics.com/p /arbotix-robot-controller.aspx
[10]https://www.robotis.us/dynamixel-ax-12a/
[11]Conversão de texto para áudio usando o serviço na nuvem do IBM Watson
[12]https://www.matrix.one/products/voice

Figura 2.14. Uma sessão de terapia de estimulação cognitiva com três participantes sendo conduzida pelo robô Eva [Cruz-Sandoval et al. 2020]

EVA descritas neste trabalho, além de apresentar as capacidades de interação multimodal disponíveis em cada modelo.

2.3.2.2. Componentes de Software

O EVA possui uma aplicação web para desenvolvimento de scripts. Ela trabalha com um banco de dados para salvar e carregar os scripts criados e também mantém uma pasta no sistema de arquivos para armazenar arquivos de áudio, que podem ser tocados durante as interações. A aplicação, através de uma porta serial, controla os servomotores que movem a cabeça do robô. O sistema web foi desenvolvido usando a pilha de software MEAN, que inclui as tecnologias: *MongoDB*, *Express*, *AngularJS* e *NodeJS*. A linguagem C++ foi utilizada para o desenvolvimento do firmware de controle dos servomotores, que roda na placa ArbotiX-M, e para o código de controle das animações dos LEDs RGB, que roda na placa Matrix Voice. A Figura 2.15 mostra os componentes de software do robô.

A plataforma EVA inclui um componente para desenvolver sessões interativas [Mitjans 2020]. O usuário pode criar scripts de interação usando uma linguagem de programação visual (VPL - *Visual Programming Language*), definindo o fluxo do script usando sequências, condições e loops. A VPL possui vários elementos que podem ser usados para criar scripts de interação, alguns deles são: um elemento para definir a linguagem a ser usada pelos componentes de fala-para-texto e texto-para-fala, um elemento para controlar a expressão do olhar do robô, outro componente para reconhecimento de voz, um componente para reconhecimento das expressões faciais do usuário usando a *webcam*, e um componente capaz de controlar efeitos sensoriais de luz usando a lâmpada inteligente.

A Figura 2.16 mostra uma interação que foi construída usando a VPL. Por ser uma linguagem de programação visual, a VPL não possui detalhes sintáticos como uma

Tabela 2.2. Modelos do robô EVA

	Versão 1	Versão 2	Versão 3	Versão Atual
Material	Papelão	PLA	Fibra de Vidro	PLA
Placa de controle	Raspberry PI 3	Raspberry PI 4	Raspberry PI 4	Raspberry PI 4
Sensores	Matrix Voice	Matrix Voice	Matrix Voice	Matrix Voice
Atuadores	Caixa de som	Caixa de som, servomotores	Caixa de som , servomotores	Caixa de som , servomotores
Graus de liberdade (DoF)	0	2	2	2
Aparência	Caricatura	Caricatura	Antropomórfica	Antropomórfica
Mecanismo de comunicação não verbal	Luzes LED	Luzes LED, Olhos, Mov. da cabeça	Luzes LED, Olhos, Mov. da cabeça	Luzes LED, Olhos, Mov. da cabeça
Efeitos sensoriais de luz	-	-	-	Lâmpada inteligente Xiaomi
Reconhecimento de expressões faciais	-	-	-	Webcam

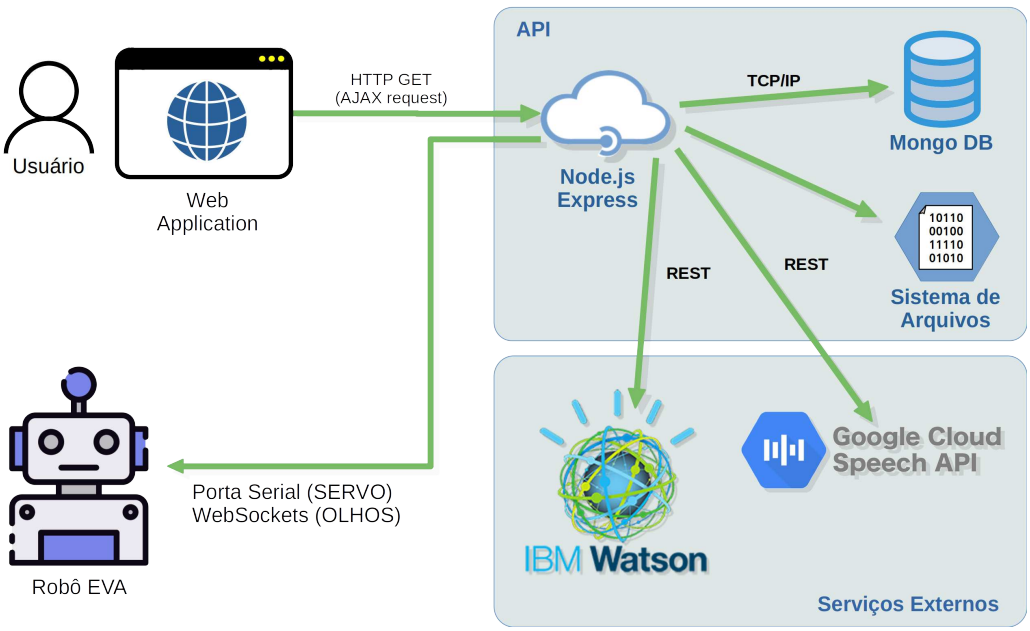

Figura 2.15. Arquitetura de software do robô EVA

linguagem de programação convencional. Ela foi criada com o objetivo de facilitar a construção de scripts de interação por pessoas que não são especialistas em programação. Usando esta ferramenta gráfica, construída com o framework web GoJS[13], um script pode ser desenvolvido de forma simples, apenas arrastando e soltando os elementos de controle da linguagem na interface gráfica.

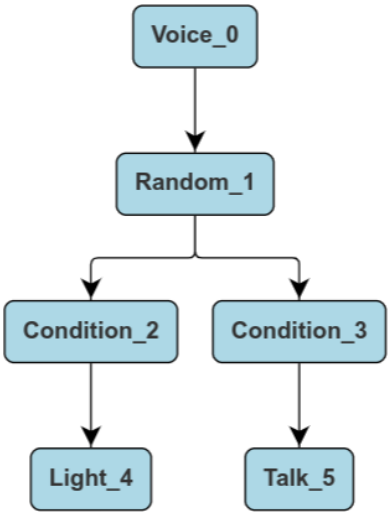

Figura 2.16. Um script na VPL

O fluxo de execução de um script na VPL ocorre de cima para baixo e no caso de elementos condicionais, da esquerda para a direita. Um script na VPL é representado por um grafo onde os nós são os elementos da linguagem (capacidades do robô) e as arestas indicam a sequência do fluxo de execução. Através do uso de elementos condicionais é possível alterar o curso da execução do script dependendo de uma condição. Como pode ser visto na Figura 2.16, o script representado na imagem possui cinco elementos distintos. O elemento *Voice* é utilizado para configurar o idioma e timbre de voz utilizados nos processos de texto-para-fala e fala-para-texto. O elemento *Random* gera um número natural aleatório dentro de um intervalo especificado em seus parâmetros. O elemento *Condition* avalia uma condição, no exemplo avalia o número gerado aleatoriamente pelo comando anterior e, dependendo do resultado, o fluxo de execução do script pode seguir diferentes caminhos. O elemento que segue o caminho da esquerda é o *Light*, que controla a lâmpada inteligente, seu estado (ligado ou desligado) e sua cor. O elemento que segue o caminho da direita é o *Talk*, que faz o EVA falar o texto especificado como parâmetro. Cada elemento da VPL possui um conjunto de parâmetros que são configurados no momento de sua inserção no script.

2.3.3. Montagem Física do Robô

Esta seção apresenta uma visão geral do processo de montagem do robô com algumas imagens reais das conexões das placas acopladas às peças do corpo do robô. Também será apresentada a solução para uma dificuldade que pode ocorrer no processo de montagem dos componentes, a configuração dos IDs (identificadores) dos servomotores. Para

[13]https://gojs.net/latest/index.html

informações mais detalhadas sobre os arquivos dos modelos 3D e um passo-a-passo da montagem dos componentes, acesse o repositório[14] do EVA no Github.

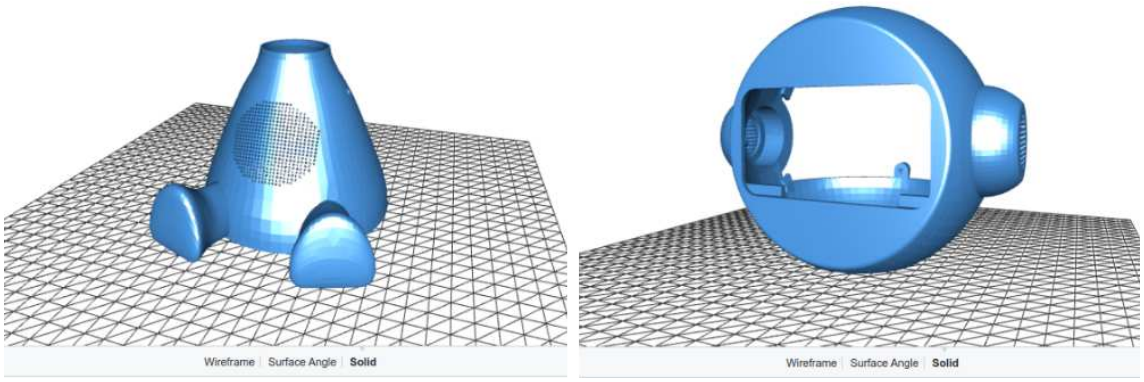

Figura 2.17. Modelos para a impressão 3D

Todo o corpo do robô é impresso em impressora 3D e os criadores do robô disponibilizam todos os arquivos e modelos para a impressão e construção do EVA. A Figura 2.17 mostra algumas imagens dos modelos 3D do corpo e da cabeça do robô.

Com todas as partes do corpo do robô impressas em 3D e todos os componentes de hardware em mãos, pode-se dar início ao processo de junção das partes internas do robô e a conexão das placas, dos cabos, do display e dos dois servomotores. A Figura 2.18(a) apresenta uma visa frontal da placa Matrix Voice com seus 8 microfones e sua matriz de 18 LEDs RGB. A Figura 2.18(b) mostra a parte traseira do display dentro da cabeça do robô. Ao display se conectam três cabos, um cabo de energia, um cabo mini HDMI e um cabo USB que é conectado ao Raspberry PI permitindo que o display funcione como dispositivo de entrada através do toque na tela. A Figura 2.18(c) mostra a Matrix Voice conectada ao Raspberry PI e como elas juntas são fixadas em uma peça interna ao robô. Essa peça, que também foi impressa, serve como suporte para todas as placas e também para os dois servomotores.

A placa ArbotiX-M é a responsável pelo controle dos dois servomotores que movimentam a cabeça do robô. Essa placa pode controlar até três servomotores e pode ser programada utilizando-se a IDE do Arduino. Ela se conecta ao Raspberry PI através de um cabo FTDI-USB que é usado tanto para a transferência do firmware da placa quanto para a comunicação serial. A placa ArbotiX-M também é usada para programar e definir os IDs dos servomotores. A Figura 2.19 mostra esses três componentes.

A placa ArbotiX-M possui três conexões destinadas ao controle de três servomotores. O tipo de conexão usada é a *DaisyChain*, isto é, os servomotores são ligados em série, de maneira sequencial. Para ter o controle individual de cada dispositivo, nesse tipo de conexão, os servomotores precisam ter IDs diferentes. Os servomotores AX-12A podem ter esses parâmetros definidos através da alteração dos seus firmwares. Esse processo de alteração do firmware do servomotores pode ser feito utilizando-se a placa ArbotiX-M.

O processo, que será apresentado neste texto, usa a ArbotiX-M junto com a IDE

[14]https://github.com/eva-social-robot

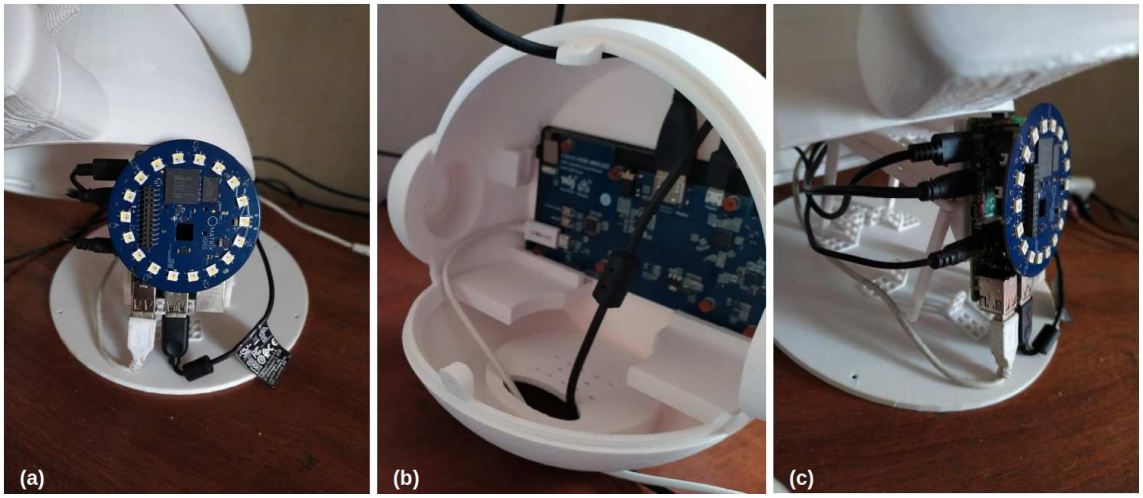

Figura 2.18. (a) Visão frontal da placa Matrix Voice com os 18 LEDs RGB, (b) Montagem do display de toque de 5.5" e (c) Matrix Voice acoplada ao Raspberry PI

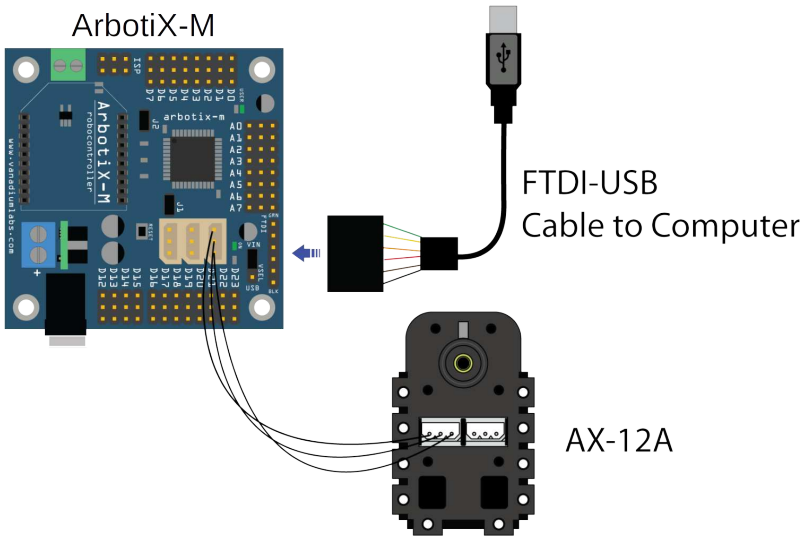

Figura 2.19. Placa ArbotiX-M, servomotor AX-12A e um cabo FTDI-USB

do Arduino, sem utilizar qualquer software intermediário. A ideia é usar uma biblioteca para a ArbotiX-M, que se comunica com o servomotor, e pode escrever e ler nos seus registradores. A Listagem 2.1 mostra o código do sketch[15] que deve ser importado e executado na IDE do Arduino. Este pequeno código faz com que o servomotor com ID=1 seja alterado para ID=2.

```
#include <ax12.h>
#include <BioloidController.h>

void setup()
{
  // param1 = ID do servo
  // param2 = n. do registrador
  // param3 = novo ID

  ax12SetRegister(1, 3, 2);
}

void loop() {}
```

Listagem 2.1. Código para alterar o ID do servomotor AX-12A

O código a seguir pode ser usado para descobrir o ID de um servomotor conectado a ArbotiX-M e alterá-lo. Pode-se ver o código do sketch para a ArbotiX-M na Listagem 2.2. Para descobrir o ID dos servomotores, basta fazer o *upload* do código para a placa ArbotiX-M e abrir o monitor da porta serial da IDE do Arduino, como indicado na Figura 2.20. Após a abertura da janela que monitora a porta serial na IDE, será apresentada uma saída semelhante à da Figura 2.21, exibindo os IDs dos servomotores conectados à placa ArbotiX-M.

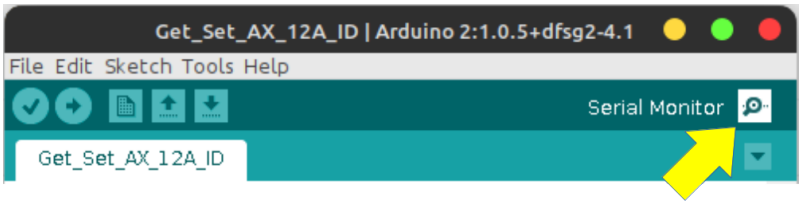

Figura 2.20. Monitor de porta serial da IDE do Arduino

```
/*
Autor: Marcelo Marques da Rocha
Laboratorio MidiaCom - Universidade Federal Fluminense (UFF)

Programa para encontrar o ID do servomotor AX-12A
Para alterar o ID de um servomotor:
1) Rode o programa para descobrir o ID do servomotor
2) Em seguida, descomente a linha com o comando ax12SetRegister(x, y, z
   );
3) Substitua os parametros da funcao de acordo com a sua necessidade*/
```

[15]Um sketch é o nome que é usado para se referir a um programa dentro da IDE do Arduino. É a unidade de código que é carregada e executada em uma placa do tipo Arduino.

Figura 2.21. Saída do programa que mostra o ID de um servomotor na janela de monitoramento da porta serial

```
#include <BioloidController.h>
#include <ax12.h>

void setup()
{
  // param1 = ID do servo
  // param2 = n. do registrador
  // param3 = novo ID
  //ax12SetRegister(10, 3, 2);
  Serial.begin(9600);
}

void loop()
{
  delay(2000);
  Serial.println("Iniciando busca pelo ID do servomotor.");
  Serial.println("-----------------------------------------------");
  Serial.println("ID\tEncontrado");
  Serial.println("-----------------------------------------------");
  int servID = -1;
  for (int i=0; i<=255; i++)
  {
    delay(100);
    // param1 = ID do servo (a verificar)
    // param2 = n. do registrador
    // param3 = numero de bytes lidos
    servID = ax12GetRegister(i, 3, 1);
    Serial.print(i); // imprime a coluna do ID
    if (servID != -1){ // imprime o resultado da busca
      Serial.println("\tOK");
    } else {
      Serial.println("\t---");
    }
```



```
  }
  Serial.println("-----------------------------------------------");
  Serial.println("Fim de busca.");
  while(1) ;
}
```

Listagem 2.2. Código que mostra o ID de um servomotor

Após a correta configuração dos IDs dos servomotores, o firmware de controle do EVA deve ser enviado para a ArbotiX-M.

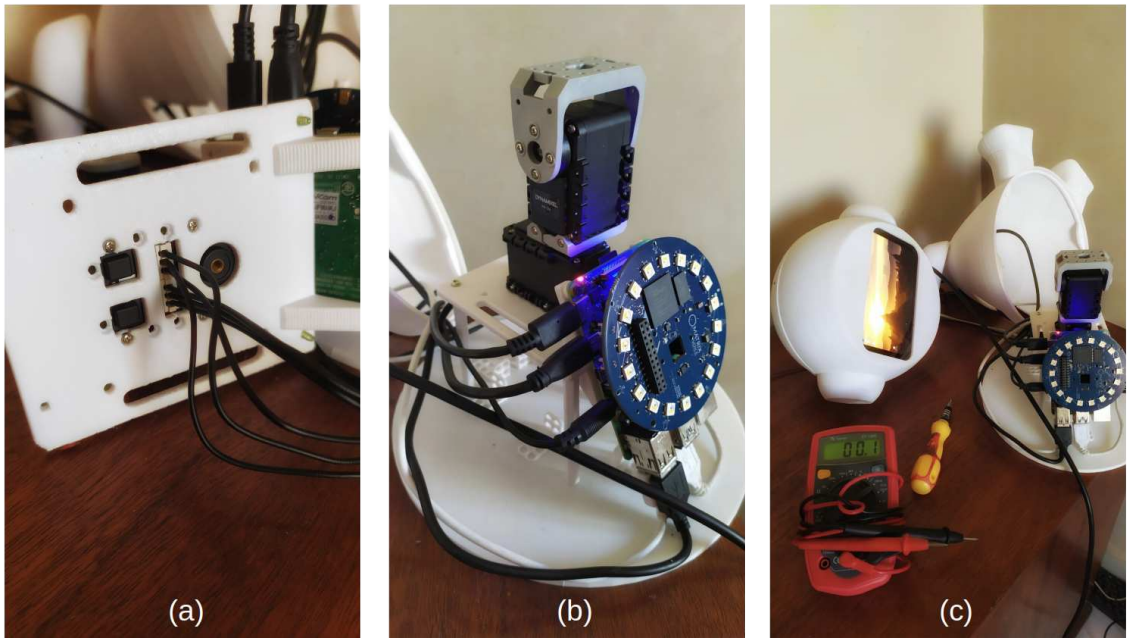

Figura 2.22. Fixação dos servomotores no suporte na parte interna do Eva

A Figura 2.22 mostra a montagem final dos componentes internos do robô. O item (a) mostra os cabos que conectam os dois servomotores à placa ArbotiX-M. A imagem do centro, item (b), apresenta todo o sistema de placas e os dois servomotores conectados um ao outro. O item (c) mostra o momento em que o robô, com seu hardware totalmente instalado, será fechado e estará pronto para a instalação do seu software de controle.

Para instalar o software de controle do EVA, é preciso acessar o terminal de dentro do sistema Raspbian no Raspberry PI, e baixar do repositório do robô no Github os fontes da aplicação. Logo após, seguindo os passos descritos na página do Github, será preciso instalar as dependências do software robô, como o NodeJS, por exemplo. Para que o EVA possa falar, transformando texto em fala, será necessário criar um conta no serviço IBM Watson e baixar uma chave em formato de texto que será usada pela aplicação do robô. O mesmo será necessário para que o robô possa converter fala em texto, utilizando a API do Google.

2.3.4. Funcionalidades Principais do Robô EVA

Em sua versão básica, o robô possui capacidades de comunicação verbal e não verbal. Ele pode falar, gerando um áudio a partir de texto e também pode reconhecer a fala. Para

o processo de TTS (*Text-To-Speech*), texto para fala, o robô usa os recursos cognitivos do IBM-Watson[16] e para o processo STT (*Speech-To-Text*), fala para texto, ele usa os serviços da API do Google[17].

O robô pode expressar emoções através do olhar e essas emoções podem ser enfatizadas com a movimentação da sua cabeça. É possível executar animações com os LEDs que ficam na placa Matrix Voice, que se encontra no tórax do robô. Além dessas capacidades, na sua versão básica, serão apresentadas duas novas extensões adicionadas ao EVA em [Rocha et al. 2021]. Essas novas capacidades de interação são: o controle de efeitos sensoriais de luz, usando uma lâmpada inteligente, e o reconhecimento de expressões faciais usando uma webcam. A Figura 2.23 mostra a imagem de uma criança interagindo com o robô já utilizando essas duas novas extensões.

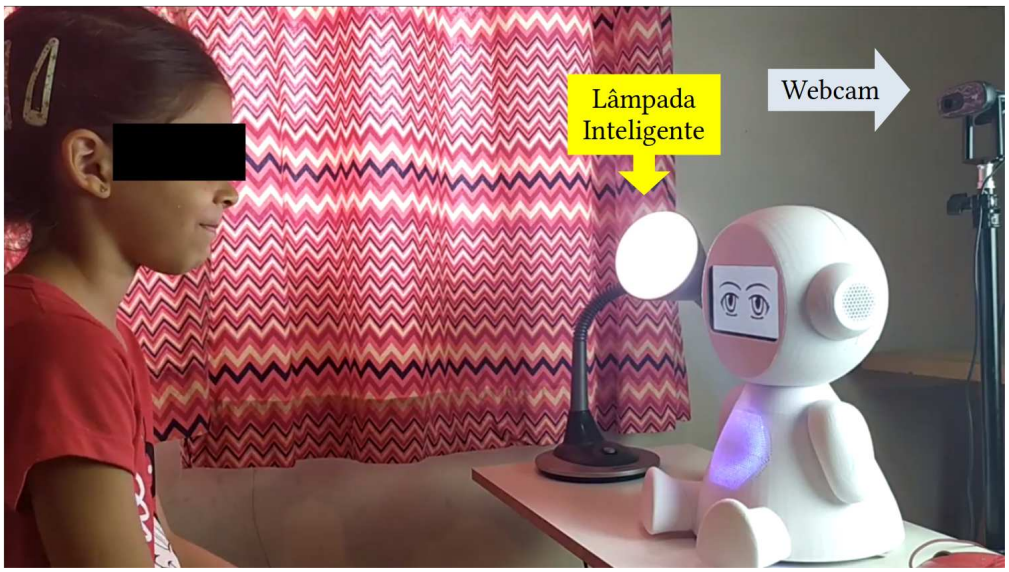

Figura 2.23. Criança interagindo com o robô EVA

2.3.4.1. Expressões do Olhar Usando o Display

O robô EVA pode expressar emoções através do seu olhar utilizando o display de 5.5 polegadas. Ele pode renderizar quatro tipos de expressões. A expressão que representa a emoção *neutra*, expressão padrão do robô, é apresentada assim que o robô é ligado. A segunda expressão é a expressão de *alegria*. Ele pode expressar também *tristeza* e *raiva*. A Figura 2.24 mostra a imagem das quatro expressões do robô.

2.3.4.2. Reconhecimento de Voz

A interação por voz é um recurso importante no processo de interação homem-robô. O EVA pode capturar o áudio da fala do usuário, usando os oito microfones que se encon-

[16]https://www.ibm.com/watson/developercloud/text-to-speech.html
[17]https://cloud.google.com/speech/

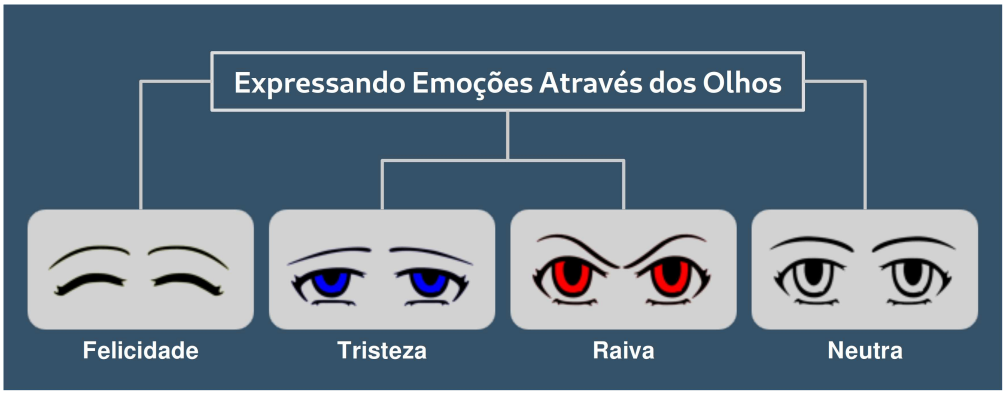

Figura 2.24. Expressões sintetizadas pelo robô EVA

tram na placa Matrix Voice, que fica no seu tórax. Após a captura do sinal de áudio, o arquivo é enviado para a API do Google que transforma o conteúdo da fala em um arquivo de texto. A string retornada é então processada pelo robô e utilizada no processo de comunicação com o usuário.

2.3.4.3. Fala

Utilizando os recursos cognitivos do IBM Watson, o robô consegue transformar texto em fala. O conjunto de comandos da VPL do robô contém com um comando que recebe como parâmetro um texto e envia esse texto para nuvem, que então retorna o áudio do texto sintetizado. O serviço do IBM Watson permite que seja definido o timbre de voz da fala a ser gerada. Existem opções de vozes masculinas e femininas. A escolha do timbre de voz está atrelada diretamente ao idioma usado no processo texto-para-fala do robô.

2.3.4.4. Execução de Arquivos de Áudio

Como foi mostrado na Figura 2.13 o EVA conta com uma caixa de som *Bluetooth*. Esse dispositivo permite que o robô possa tocar arquivos de áudio no formato *wav*. Para ser usado pelo robô, isto é, poder ser acessado pelo comando responsável por tocar áudio, o arquivo de som deve estar dentro da pasta "sonidos" que se encontra no diretório raiz da aplicação de controle do robô dentro do Raspberry PI.

2.3.4.5. Animação dos LEDs RGB

Como um recurso de comunicação não verbal, o robô utiliza a matriz com dezoito LEDs RGB que se encontra na sua placa Matrix Voice. A linguagem visual do robô possui um comando que permite a execução de um conjunto de animações predefinidas usando esses LEDs. Algumas dessas animações estão associadas a outras ações do robô e são executadas junto com os comandos relacionados. Por exemplo, ao falar, o robô executa uma animação com os LEDs na cor azul, e ao escutar, o robô executa uma animação com os LEDs na cor verde.

2.3.4.6. Movimentação da Cabeça

O robô usa dois servomotores Dynamixel AX-12A[18] que fornecem 2 graus de liberdade para a cabeça do EVA. O comando de controle do movimento da cabeça do robô pode fazer com que ele mova a cabeça para cima, para baixo, para a esquerda e para a direita. Para cada movimento, há uma opção que realiza o movimento com menor ou com maior amplitude. Há também a opção de movimentação chamada "sim" que faz com que o robô execute um movimento de sobe e desce com a cabeça, como se estivesse respondendo "sim" com a cabeça. Da mesma maneira, há o movimento de "não". A movimentação da cabeça do robô serve como um componente capaz de aumentar a expressividade do robô, por exemplo quando o robô demonstra uma expressão de raiva através do olhar, essa expressão pode ser acentuada fazendo com que a cabeça do robô se incline para baixo.

2.3.4.7. Controle de Efeitos Sensoriais de Luz

As duas últimas funcionalidades que serão mostradas a seguir foram apresentadas em [Rocha et al. 2021] como uma proposta de aprimorar as capacidades de interação multimodal do robô. A partir dessa proposta, o EVA passou a ter um novo componente que foi adicionado como elemento de primeira classe à sua linguagem de programação visual. Esse elemento foi chamado de *Light*, pois deu ao robô a capacidade de controlar efeitos sensoriais de luz usando uma lâmpada inteligente. A Figura 2.25 mostra o esquema de conexão e controle do robô sobre a lâmpada inteligente. Após a lâmpada estar configurada corretamente na rede wifi, o robô, através de uma conexão TCP, se conecta à lâmpada. Através do comando *Light*, o robô pode ligar e desligar a lâmpada, como também pode definir a sua cor usando uma paleta de cores RGB com 24 bits de cor. O comando de controle e seleção de cor é enviado do robô para a lâmpada como uma string JSON.

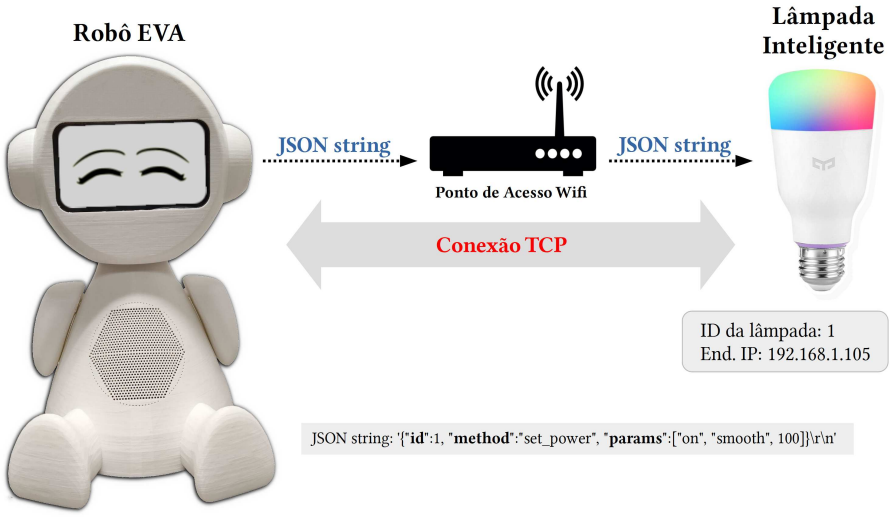

Figura 2.25. Controlando efeitos sensoriais de luz

[18]https://www.robotis.us/dynamixel-ax-12a/

2.3.4.8. Reconhecimento de Expressões Faciais

A capacidade de identificar a emoção do usuário por meio do reconhecimento de expressões faciais e utilizar essas informações dentro do aplicativo é uma facilidade importante, que amplia o poder de interação do robô com o usuário [Valentim et al. 2020]. Para isso, uma segunda extensão foi adicionada à VPL dando ao robô a capacidade de reconhecer expressões faciais. Esse novo componente foi denominado *UserEmotion*. Ele usa os serviços de um módulo externo desenvolvido em Python [Valentim et al. 2020]. Tanto o módulo Python quanto o software EVA rodam no mesmo dispositivo, um Raspberry PI 4.

A comunicação entre o módulo e o robô é feita através de uma conexão TCP. Quando iniciado, o módulo de reconhecimento facial cria um servidor TCP que fica aguardando a conexão com o robô. Após a conexão ser estabelecida o módulo de reconhecimento facial aciona a *webcam*, iniciando assim os processos de captura de imagem e inferência da expressão facial. Ao final do procedimento, o módulo retorna ao cliente TCP (EVA), uma string contendo a expressão identificada. O módulo de reconhecimento facial pode retornar as seguintes expressões: "NEUTRAL", "ANGRY", "DISGUST", "FEAR", "SURPRISE", "HAPPY" e "SAD". Para obter a expressão facial do usuário, através do componente *UserEmotion*, o robô envia uma solicitação ao módulo de reconhecimento facial Python. O processo funciona conforme ilustrado na Figura 2.26.

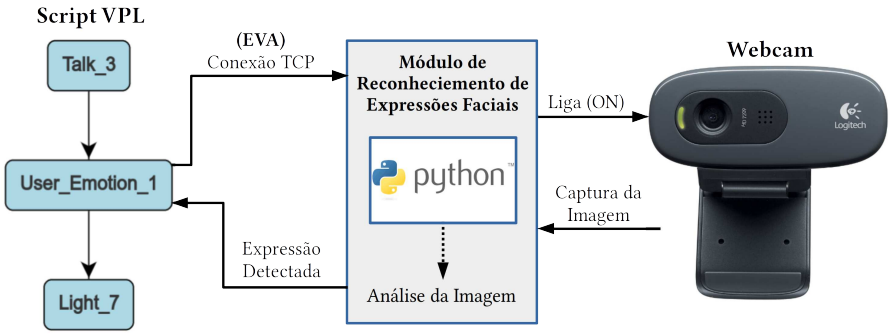

Figura 2.26. Reconhecimento de expressões faciais

2.4. Desenvolvimento de Sessões de Terapia para o Robô EVA

Originalmente, o robô EVA possui uma linguagem de programação visual (VPL), que visa facilitar o desenvolvimento de interações por pessoas sem experiência em programação de computadores. A VPL foi construída usando uma poderosa ferramenta gráfica chamada GoJS[19]. Durante o processo de criação de um script utilizando VPL, o usuário é responsável por inserir os comandos do robô, que são nós no fluxo de execução. No entanto, os elementos são posicionados automaticamente na área gráfica da interface pelo aplicativo GoJS. Os nós do gráfico de script são reposicionados automaticamente após a remoção ou inserção de um nó ou aresta do gráfico. Quando um script se torna grande, esse reposicionamento automático geralmente dificulta a construção do script. À medida que o número de elementos gráficos no script de interação aumenta, a visualização do gráfico se torna muito limitada. Para ter uma visão geral do gráfico, a interface permite

[19]https://gojs.net/latest/index.html

utilizar os recursos de "*zoom in*" e "*zoom out*", porém, com o tamanho reduzido dos elementos, fica muito difícil identificar visualmente cada um deles. É possível selecionar um grupo de nós e arestas e duplicá-los usando o recurso de copiar e colar, o problema, nesse caso, é que o script passa a ter elementos diferentes (que podem ter propriedades diferentes) com o mesmo identificador visual (nome) e isso dificulta muito o processo de edição das propriedades dos elementos duplicados. A observação destas limitações da linguagem visual do robô surgiu durante um processo de experimentação, que investigou a aceitação do uso do robô como ferramenta assistiva aplicada em terapias com pacientes com Transtorno do Espectro Autista (TEA) [Rocha et al. 2021].

Embora o uso de ferramentas gráficas torne um usuário com alguma experiência produtivo, um usuário com habilidades avançadas no domínio da aplicação pode ter sua eficiência reduzida [Novák 2010]. Pensando nestas questões, foi proposta a criação de uma linguagem específica de domínio baseada em XML (*Extensible Markup Language*) para criar sessões de terapia com o robô EVA, com os seguintes objetivos: possibilitar maior controle na entrada e edição de comandos da linguagem e seus respectivos parâmetros; adicionar abstrações de elementos de programação que visam facilitar a construção de scripts; possibilitar o desenvolvimento de scripts independentes da interface de controle do robô, ou seja, utilizando qualquer editor de texto.

A linguagem XML tem muitas vantagens para o desenvolvimento de linguagens específicas de domínio, destacando-se:

1. ser mais legível para não-programadores do que as linguagens de propósito geral.

2. em uma linguagem específica de domínio baseada em XML, a gramática pode ser descrita usando uma DTD (*Document Type Definition*) ou XML *Schema*.

3. É simples analisar a estrutura XML usando o DOM (*Document Object Model*).

2.4.1. EvaML

Com o objetivo de facilitar o desenvolvimento de sessões interativas por pessoas com conhecimento técnico em programação, mas ainda mantendo a legibilidade dos códigos dos scripts, foi criada a EvaML. Uma linguagem baseada em XML para a especificação de sessões interativas usando a plataforma de robótica *open-source* EVA. A EvaML possibilita a criação de scripts de interação para o robô EVA usando apenas um simples editor de texto. Todos os comandos que controlam os elementos de interação multimodal do robô estão presentes na EvaML, entre eles, o componente *Light* (que controla a lâmpada inteligente), os comandos de reconhecimento de voz e o comando *UserEmotion* que possibilita o reconhecimento da expressão facial do usuário através de uma *webcam*. A linguagem também possui elementos para criação e manipulação de variáveis, geração de números aleatórios, controles condicionais usando elementos *switch* e *case* e outros. O *parser* EvaML gera automaticamente um script correspondente no formato JSON que pode ser adicionado ao banco de dados de scrpits do robô e então ser executado por ele.

2.4.1.1. Elementos de um Documento EvaML

A Figura 2.27 apresenta um script EvaML com o seu *elemento raiz* <evaml> e seu atributo name. Este elemento raiz contém os três seguintes elementos: <settings>, <script> e <macros>.

Figura 2.27. Seções de um documento EvaML

Tabela 2.3. Elementos de um documento EvaML (elemento raiz e elementos principais)

Elemento	Atributo	Conteúdo
evaml	name	(settings, script, macros?)
settings		(voice | lightEffects? | audioEffects?)
script		(random* | wait* | talk* | stop* | light* | goto* | userEmotion* | evaEmotion* | useMacro* | listen* | audio* | led* | counter* | switch*)
macros		(macro+)

A Tabela 2.3 mostra o elemento raiz do documento EvaML (<evaml>) e os elementos <settings>, <script> e <macros> que representam as seções do documento. São apresentados os conteúdos e atributos de cada elemento. Na coluna *Atributo*, um atributo sublinhado indica que este é obrigatório. Na coluna de *Conteúdo*, os indicadores de ocorrência são usados para indicar a ordem e o número de vezes que um elemento pode ou deve ocorrer. O caractere "," (vírgula) indica que todos os elementos filhos listados devem ser usados na sequência mostrada. O caractere "|" (barra vertical) indica que qualquer elemento pode ocorrer dentro do elemento pai. O sinal de "+" (sinal de adição), por outro lado, indica que o elemento filho deve aparecer uma ou mais vezes. O caractere

"*" (asterisco) indica que o elemento pode ser usado zero ou mais vezes dentro do elemento pai. O caractere "**?**" (ponto de interrogação) indica que o elemento é opcional, ou seja, o elemento pode não existir ou existe apenas uma ocorrência dele.

Elemento settings - Define algumas características globais do script. É possível definir como será o timbre da voz e o idioma em que o robô irá se comunicar. É possível definir se o código gerado, ao ser executado, irá reproduzir comandos de efeitos de luz, efeitos sonoros ou até mesmo tocar música. Ao configurar esses parâmetros, é possível modificar globalmente o funcionamento do script sem ter que alterar diretamente as definições de seus elementos individuais. A seguir um trecho de código que exemplifica o elemento <settings>.

```
<settings>
  <voice tone="en-US_AllisonV3Voice" />
  <lightEffects mode="ON" />
  <audioEffects mode="ON"/>
</settings>
```

Elemento script - Contém a sequência de comandos que o robô deve executar. Pode-se ver alguns deles no trecho de código a seguir. Na linha 2 do script, se encontra o comando <light> que acende a lâmpada inteligente definindo sua cor como azul. Na próxima linha, o comando <talk> faz o robô dizer algo, por exemplo, se apresentar. O comando <wait>, na linha 4, faz com que a execução do script seja pausada por 2000ms (2s). Na próxima linha, o comando <audio> reproduz um arquivo de áudio chamado "*mario-start*". Então, o robô se despede falando "Bye" e desliga a lâmpada inteligente. O atributo block definido como "TRUE" no elemento <audio> (linha 5) faz com que a execução do script siga somente após o término da reprodução do arquivo de áudio.

```
<script>
  <light state="ON" color="BLUE" />
  <talk>Hi, I am robot EVA</talk>
  <wait duration="2000" />
  <audio source="mario-start" block="TRUE" />
  <talk>Bye</talk>
  <light state="OFF" />
</script>
```

Elemento macros - É uma das abstrações criadas na linguagem EvaML. Como pode ser visto no próximo trecho de código, é possível criar **macros** que podem ser referenciadas dentro do elemento <script>. Uma macro tem o atributo id que serve para identificá-la. Estas macros podem ser usadas dentro da seção <script> usando o comando <useMacro>. O atributo macro do comando <useMacro> faz referência ao elemento <macro> definido dentro de <macros> . Durante o processo de *parsing* do documento EvaML, as macros são expandidas com seu conteúdo na seção <script>. Não há limite para o número de macros criadas, nem para o número de referências a essas macros no script. Como pode ser visto na Tabela 2.3, o elemento **macros** não é obrigatório.

```
<script>
```



```
  <useMacro macro="START" />
</script>
<macros>
  <macro id="START">
    <talk>Hello, I'm robot Eva.</talk>
    <talk>What is your name?</talk>
    <talk>Let us play one more time?</talk>
  </macro>
</macros>
```

2.4.1.2. Comandos da Linguagem EvaML

Antes de apresentar a descrição de cada comando da linguagem EvaML, a Tabela 2.4 mostra os elementos da EvaML que representam os comandos da linguagem. Seus atributos e o que cada elemento pode conter também são listados. Para tal representação, será utilizada a mesma notação utilizada na Tabela 2.3. A descrição de cada símbolo usado pode ser vista na Seção 2.4.1.1. Será apresentada a seguir uma breve descrição de cada comando da EvaML, para uma descrição mais detalhada sobre os comandos da linguagem e exemplos de scripts com cada elemento, acesse o manual da EvaML disponível neste link[20]..

<voice> - Como um elemento de configuração da linguagem, este comando possui apenas um atributo que define, ao mesmo tempo, o timbre de voz (seu gênero) a ser usado pelo robô, e o idioma que será utilizado durante o processo de conversão de texto para fala do serviço IBM Watson. Ele possui apenas o atributo tone, e como já foi citado anteriormente, quando um atributo, na coluna de atributos das Tabelas 2.3 e 2.4 se encontra sublinhado, seu uso é obrigatório. Como é um comando de configuração, deve estar contido no elemento <settings>. A Tabela 2.5 exibe uma pequena lista com algumas opções de vozes para o robô.

<random> - Este comando gera um número inteiro aleatório no intervalo fechado [min, max]. O atributo min, representa o limite inferior, e o atributo max, representa o limite superior do número aleatório a ser gerado. O valor gerado pela função aleatória é armazenado em uma região especial da memória do robô, que funciona como um vetor. O caractere **$** acessa o elemento no final desse vetor.

<wait> - Este comando pausa a execução do script pelo intervalo de tempo definido no seu atributo duration. A unidade de tempo usada é o milissegundo.

<talk> - Uma das capacidades de interação multimodal do robô EVA é a fala e através do uso do comando <talk> o robô pode falar um texto especificado. Ao se definir o texto a ser falado é possível utilizar o conteúdo da memória do robô como parte do texto, utilizando-se, no corpo do texto, o caractere **$**, que referencia uma área especial da

[20]https://github.com/midiacom/eva-robot/blob/master/EvaML-Reference-Manual/EvaML-Reference-Manual.pdf

Tabela 2.4. Elementos de um documento EvaML (Comandos)

Elemento	Atributo	Conteúdo
voice	tone	empty
lightEffects	mode	empty
audioEffects	mode	empty
random	id, min, max,	empty
wait	id, duration	empty
talk	id	text
stop		empty
light	id, state, color	empty
goto	target	empty
motion	id, type	empty
userEmotion	id	empty
evaEmotion	id, emotion	empty
useMacro	macro	empty
listen	id	empty
audio	id, source, block	empty
led	id, animation	empty
counter	id, var, op, value	empty
switch	id, var	(case+, default?)
macro	id	(random* | wait* | talk* | stop* | light* | goto* | userEmotion* | evaEmotion* | listen* | audio* | led* | counter* | switch*)
case	op, value	(random* | wait* | talk* | stop* | light* | goto* | userEmotion* | evaEmotion* | useMacro* | listen* | audio* | led* | counter* | switch*)
default		(random* | wait* | talk* | stop* | light* | goto* | userEmotion* | evaEmotion* | useMacro* | listen* | audio* | led* | counter* | switch*)

Tabela 2.5. Vozes para o serviço de texto para fala do IBM Watson

Código	Gênero	Idioma
pt-BR_IsabelaV3Voice	feminino	português do Brasil
en-US_AllisonV3Voice	feminino	inglês dos EUA
en-US_EmilyV3Voice	feminino	inglês dos EUA
en-US_HenryV3Voice	masculino	inglês dos EUA
es-LA_SofiaV3Voice	feminino	espanhol latinoamericano
es-ES_EnriqueV3Voice	masculino	espanhol

memória do robô[21] ou a notação **#var** que referencia o conteúdo de uma variável definida pelo usuário.

<stop> - Este comando é muito simples e como o nome sugere, ele interrompe a execução do script.

<light> - O controle da lâmpada inteligente pode ser feito usando este comando. O seu atributo state pode assumir os valores "ON" e "OFF" e o atributo color, define a cor da lâmpada. Essa cor pode ser indicada usando a representação hexadecimal RGB "#00ff00" ou algumas das cores da lista predefinida: "WHITE", "BLACK", "RED", "PINK", "GREEN", "YELLOW", "BLUE".

<goto> - Este comando altera o fluxo de execução do script para o comando com id referenciado em seu atributo target. O atributo id define o rótulo que será usado como valor no atributo target do comando <goto>.

<motion> - Conforme mencionado na Seção 2.3.4.6, o robô pode mover sua cabeça e o comando <motion> é responsável por controlar este movimento. O comando possui o atributo type que pode assumir os seguintes valores: "YES", "NO", "CENTER", "LEFT", "RIGHT", "UP", "DOWN", "ANGRY", "2UP", "2DOWN", "2LEFT" e "2RIGHT". Ao se definir type para "YES", faz com que o robô execute um movimento "*sim*" com a cabeça. Usar o valor "NO" faz com que o robô sinalize um "*não*" com sua cabeça. As opções "LEFT", "RIGHT", "UP" e "DOWN" são responsáveis por movimentar a cabeça do robô nas direções correspondentes. O valor "ANGRY" faz com que o robô incline a cabeça para baixo. A opção "CENTER" faz com que a cabeça do robô retorne à sua posição original centralizada. Os valores "2UP", "2DOWN", "2LEFT" e "2RIGHT" executam os movimentos nas direções correspondentes, porém, com o dobro da amplitude do movimento.

<userEmotion> - O robô é capaz de reconhecer expressões faciais através de uma *webcam*. Ele faz isso usando um módulo externo escrito em Python que roda como um serviço dentro do mesmo dispositivo que roda o software do robô, um Rasperry PI 4. O processo de captura da expressão facial do usuário é o seguinte. O EVA envia uma solicitação de reconhecimento facial para o módulo Python, o módulo recebe a solicitação, ativa a *webcam* e retorna as seguintes expressões[22] como uma string: "NEUTRAL", "ANGRY", "DISGUST", "FEAR", "SURPRISE", "HAPPY" e "SAD". Essa resposta pode ser usada no restante do script sendo acessada através da variável **$**. A Figura 2.26 ilustra o processo de comunicação do robô com o módulo *Python*.

<evaEmotion> - Este comando controla a apresentação das expressões do olhar do robô na tela de 5.5 polegadas. O seu atributo emotion pode ter os seguintes valores: "HAPPY", "SAD", "ANGRY" e "NEUTRAL". Estas expressões, respectivamente traduzidas para o português, podem ser vistas na Figura 2.24.

<macro> - Como foi visto na Seção 2.4.1.1 uma *macro* define uma sequência de comandos que podem ser referenciados dentro do elemento <script> usando o comando <useMacro>. O elemento <macro> tem apenas o atributo id que é usado para identificá-

[21] A explicação detalhada do uso do caractere **$** pode ser vista na Seção 2.3.4, Figura 2.2, do Manual da EvaML

[22] O simulador EvaSIM trabalha com cinco expressões que serão apresentadas na Seção 2.4.2.1

lo. É importante saber que um elemento <macro> não pode conter a definição de outra *macro* e não pode, por meio de um comando <useMacro>, referenciar outra *macro*.

<useMacro> - Já foi apresentado como as macros podem ser definidas usando o comando <macro>. O comando <useMacro> faz com que o código da macro, referenciado em seu atributo macro, seja expandido pelo código onde o comando <useMacro> está declarado. Este processo de expansão de macro ocorre na primeira etapa do processo de *Parsing* da linguagem EvaML e será explicado na Seção 2.4.1.3. O comando <useMacro> possui apenas o atributo macro que se refere à macro que será expandida.

<listen> - Conforme mencionado anteriormente, o robô pode reconhecer a voz humana e para isso utiliza o serviço de conversão de fala para texto (STT) da API na nuvem do Google. A captura de áudio é feita usando os 8 microfones da placa Matrix Voice. O áudio capturado é enviado para a nuvem, processado, e então, o texto resultante do processo de STT é retornado para ser utilizado pelo software do robô. A resposta, em formato de string, pode ser referenciada através do caractere **$**.

<audio> - Este comando reproduz um arquivo de áudio contido na pasta "*sonidos*"[23]. O seu atributo source deve indicar apenas o nome do arquivo, sem o caminho da pasta e sem a sua extensão. O EVA só reproduz arquivos no formato *wav*. O seu atributo block pode ter os seguintes valores, "TRUE" ou "FALSE". Esses valores definem se a execução do arquivo de áudio deve bloquear a execução do script, ou seja, o comando que vem após o comando <audio> só será executado após o término da reprodução do mesmo.

<led> - Este comando controla a animação[24] com os LEDs no peito do robô EVA. O seu atributo animation pode assumir os seguintes valores: "HAPPY" (verde), "SAD" (azul), "ANGRY" (vermelho), "STOP" (sem cor/desligado), "SPEAK" (azul), "LISTEN" (verde) e "SURPRISE" (amarelo).

<counter> - A criação de uma variável na memória do EVA é feita através deste comando. Com ele é possível criar, inicializar e realizar operações matemáticas sobre essas variáveis. O atributo var define o nome variável, enquanto o atributo op define o tipo de operação, podendo assumir os seguintes valores: "=" (**atribuição**), "+" (**adição**), "*" (**multiplicação**), "/" (**divisão**) e "%" (**módulo**). O atributo value define o valor a ser atribuído à variável, no caso de uma atribuição, ou o valor a ser usado nas operações aritiméticas. Para referenciar o valor de uma variável criada pelo usuário, diferentemente do caractere "**$**", é necessário utilizar o caractere "**#**" antes do nome da variável. Essa notação deve ser seguida quando uma variável definida pelo usuário for utilizada dentro de um texto no elemento <talk> e no atributo value dos comandos <case>. No atributo var de um comando <switch>, o nome da variável é utlizado sem o caractere "**#**".

<switch> - Este comando define a variável que será comparada com os valores definidos nos comandos <case>. Para isso, seu atributo var deve conter o nome da variável a ser comparada, por exemplo: <switch var="$"> ou <switch var="x">. O atributo

[23] A pasta "sonidos" está localizada no diretório raiz do aplicativo do robô (no Raspberry PI) ou no diretório raiz do simulador do EVA.

[24] Algumas animações dos LEDs estão associadas a outros comandos do robô e são ativadas automaticamente com eles, como por exemplo os comandos <talk> e <listen>.

var do comando <switch> determina com qual variável as comparações serão feitas nos comandos <case> e pode assumir os valores "**$**" ou qualquer outro nome de variável que tenha sido declarada anteriormente.

<case> - O comando <case> especifica uma sequência de comandos que serão executados se a condição definida em seus atributos for verdadeira. O comando <case> possui o atributo op, que define o tipo de comparação ou operador lógico que será processado, e o atributo value, que contém o valor a ser comparado com a variável definida no atributo var do comando <switch>. O conteúdo de value pode ser uma constante (um número ou uma string), o caractere "**$**" ou outra variável usada no script. É importante lembrar que, no atributo value, para se referir ao valor de uma variável declarada anteriormente, deve-se usar o caractere "**#**" antes do nome da variável. O atributo op pode assumir três valores, que determinam três tipos diferentes de comparação: "**exact**" (exata), "**contain**" (contém) e "**math**" (matemática).

A comparação do tipo "**exact**" é usada para comparar o conteúdo do atributo value do comando <case> com o valor referenciado por "**$**". O valor especificado no atributo value será comparado exatamente com o valor referenciado por "**$**", caractere por caractere. Esta comparação não diferencia letras maiúsculas de minúsculas.

A comparação do tipo "**contain**" é usada para comparar o conteúdo do atributo value do comando <case> com o valor referenciado por "**$**". Mas neste caso, a comparação será verdadeira se o valor contido no atributo value do comando <case> for igual ao valor em "**$**" ou uma substring do mesmo. Esta comparação não diferencia letras maiúsculas de minúsculas.

As comparações de tipo "**exact**" e "**contain**" só devem ser usadas quando o atributo var do comando <switch> for "**$**", caso contrário, uma mensagem de erro será lançada pelo *parser*. Essas comparações são baseadas em strings e não em números.

Para realizar uma comparação matemática no script, deve-se utilizar os símbolos de igualdade, desigualdade, maior que, menor que, etc. Na EvaML, devido ao conflito gerado pelo uso do caractere "<" no processo de *parsing*, os operadores de comparação do tipo "**math**" (matemático) são os seguintes: "**eq**", "**lt**", "**gt**", "**lte**", "**gte**" e "**ne**", que representam, respectivamente, *igual*, *menor que*, *maior que*, *menor ou igual*, *maior ou igual* e *diferente*. O uso da comparação matemática assume que o conteúdo do atributo value do comando <case> é um número.

<default> - Conforme mostrado na Tabela 2.4, o comando <default> compõe o grupo de elementos (comandos) que são filhos do elemento <switch> e seu uso não é obrigatório. Se usado, deve vir como o último filho de um elemento <switch>, após todos os comandos <case>. O bloco de comandos contido em <default> será executado se nenhuma condição avaliada nos comandos <case> for verdadeira.

2.4.1.3. Processo de *Parsing* de um Script EvaML

O processo de *parsing* de um documento EvaML ocorre em quatro etapas, que são executadas por quatro módulos e podem ser vistas na Figura 2.29. Essas etapas serão explicadas em detalhes a seguir.

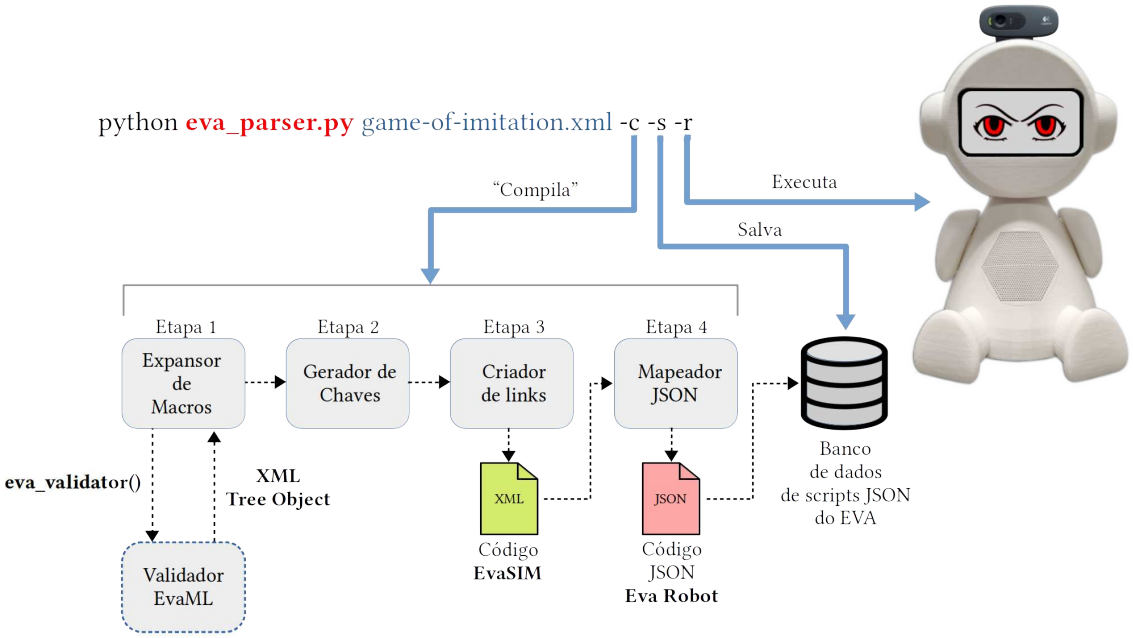

Figura 2.28. Processo de *parsing* e exemplo de linha de comando

A primeira etapa consiste no processo de expansão das macros referenciadas dentro do elemento <script>. Nesta etapa, os comandos contidos nas macros são expandidos dentro do script. Esta expansão é feita nos locais do script onde as macros são referenciadas, através do comando <useMacro>.

O script que o robô EVA executa é representado como um grafo com nós e arestas (links) codificados em JSON. Os nós são os comandos da linguagem (ou ações que o robô pode executar) e os links determinam o fluxo de execução do script. Esta segunda etapa do *parser* executa a tarefa de criação das chaves identificadoras dos nós do grafo. Elas vão identificar cada nó, de maneira unívoca. Esses nós são conectados por meio de links que usam essas chaves para referenciá-los.

A terceira etapa é responsável por analisar o fluxo de execução do script. É nessa etapa que o *Parser* identifica o fluxo de execução do script, com seus possíveis desvios, saltos e repetições, gerando os links que formam o grafo de execução do script. Como saída desta etapa, tem-se o arquivo XML destinado ao simulador do robô EVA, chamado de EvaSIM.

Na quarta etapa, o *parser* mapeia os comandos e links resultantes da etapa anterior usando os templates JSON que representam as estruturas dos comandos do robô. O *parser* identifica o elemento XML com seus atributos, processando cada comando de maneira específica. Como resultado desta última etapa, é gerado um arquivo JSON que pode ser enviado para o banco de dados de scripts do robô e executado.

Parâmetros do *parser* - O *parser* pode ser executado usando três parâmetros que são apresentados na Tabela 2.6. Usando apenas o parâmetro "**-c**", o analisador gera dois arquivos, um no formato EvaSIM e outro no formato JSON, que é o código a ser executado no robô. Utilizando o parâmetro "**-s**", o arquivo JSON gerado é inserido diretamente no

banco de dados do EVA. E finalmente, usando o parâmetro "**-r**", faz com que o robô inicie imediatamente o script. A Figura 2.28 mostra um exemplo da linha de comando usando o *Parser* com seus parâmetros. Os parâmetros "**-s**" e "**-r**" só tem funcionalidade se forem executados de dentro do diretório da aplicação do robô no Raspberry PI.

Tabela 2.6. Parâmetros do *Parser* EvaML

Parâmetro	Definição
-c	Interpreta o script EvaML gerando um arquivo XML no formato do simulador EvaSIM e um arquivo JSON
-s	Salva/Insere o arquivo JSON gerado no banco de dados do robô
-r	Faz com que o robô EVA rode o script imediatamente

2.4.1.4. Processo de Validação de um Script EvaML

A linguagem EvaML foi especificada utilizando-se uma especificação em XML Schema. O propósito de um esquema é definir e descrever uma classe de documentos XML usando essas construções para restringir e documentar o significado, uso e relacionamentos de suas partes constituintes: tipos de dados, elementos e seus conteúdos, atributos e seus valores. Um XML Schema fornece um meio para definir a estrutura, conteúdo e semântica de documentos XML.

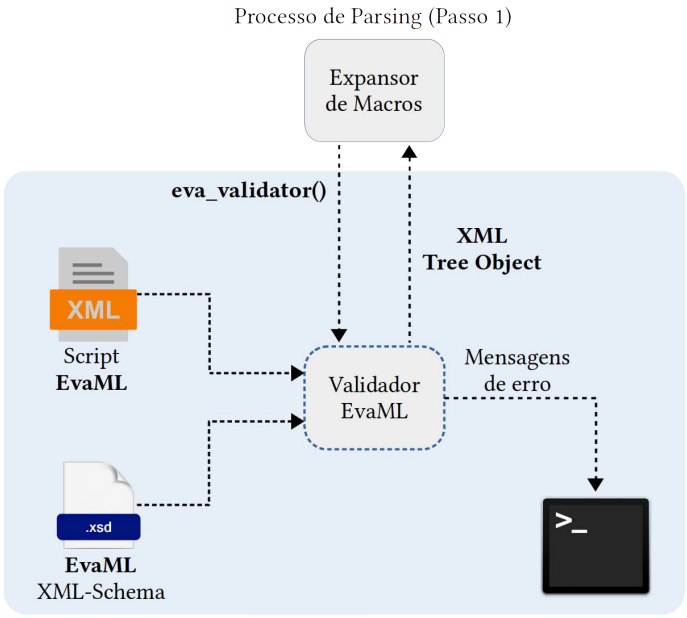

Figura 2.29. Processo de validação de um documento EvaML usando XML Schema

O processo de validação do script EvaML ocorre em paralelo e é iniciado pelo módulo *Expansor de Macros* que usa o módulo auxiliar *Validador EvaML*. A Figura 2.29 apresenta os elementos envolvidos neste processo. A etapa 1 do *parser* chama a função de validação (*eva_validator*) no módulo auxiliar passando como parâmetro o nome do script a ser validado. Dentro do módulo auxiliar está indicada a URL do arquivo XML

Schema que especifica a linguagem EvaML. O módulo auxiliar é responsável por exibir as mensagens do validador XML no terminal do sistema. Durante o processo de validação podem ocorrer três casos distintos:

1. O script EvaML é bem-formado e também válido, o módulo auxiliar retorna um objeto *XML Tree* para o módulo de expansão de macros dando prosseguimento ao processo de *parsing*.

2. O script não é bem-formado e o validador imprime uma mensagem sinalizando o problema no script indicando a possível linha do código onde o problema foi identificado. O módulo auxiliar retorna um objeto nulo para o módulo de expansão de macros, que interrompe o processo de *parsing*.

3. O script é bem-formado, porém, não é válido, ou seja, não segue a gramática definida no arquivo XML Schema da linguagem. O módulo de validação mostra no terminal todos os problemas identificados e também retorna um objeto nulo para o módulo de expansão de macros, interrompendo o processo de *parsing* do script.

O processo de validação do script EvaML usando o arquivo XML Schema da linguagem é capaz de indicar várias falhas no script. Ele pode indicar a falta de um elemento <voice> na seção <settings>, pode acusar a falta de algum atributo que tenha seu uso definido como obrigatório, pode indicar a ocorrência de duplicidade em atributos definidos como únicos, como no caso do atributo id, e pode verificar a corretude de muitos outros critérios definidos na especificação XML Schema.

2.4.2. EvaSIM - O Software Simulador do Robô EVA

Com o objetivo de fornecer um ambiente de teste para os scripts EvaML, foi desenvolvido um software que simula o robô EVA, chamado de EvaSIM. Um requisito importante que foi levado em consideração no desenvolvimento do EvaSIM é que ele fosse portável, por esse motivo, a linguagem Python foi escolhida para o seu desenvolvimento.

A interface gráfica do EvaSIM foi desenvolvida utilizando-se *Tkinter*[25], uma das bibliotecas mais populares para criar interfaces gráficas de usuário (GUI - *Graphical User Interface*)) em Python. O pacote *Tkinter* é uma fina camada orientada a objetos sobre a biblioteca *Tcl/Tk*.

A Figura 2.30 mostra a interface gráfica de usuário do EvaSIM. O layout da aplicação foi definido usando três objetos do tipo *Frame* da biblioteca *Tkinter*. O primeiro *Frame*, à esquerda da janela da aplicação, contém um objeto *Canvas* onde são desenhadas as figuras que representam os componentes do robô. O segundo *Frame*, que fica no centro da janela, contém dois elementos importantes da aplicação. Em sua parte superior, há um grupo de botões que são responsáveis por controlar o EvaSIM. Abaixo do grupo de botões de controle, há um objeto *Text* que é usado para criar um emulador de terminal para o EvaSIM. Neste terminal são apresentadas diversas ações e estados do robô simulado. O terceiro *Frame* à direita contém duas tabelas que exibem os valores das variáveis do sistema e do usuário durante a execução do script. Uma descrição de cada elemento é

[25]https://docs.python.org/3/library/tkinter.html

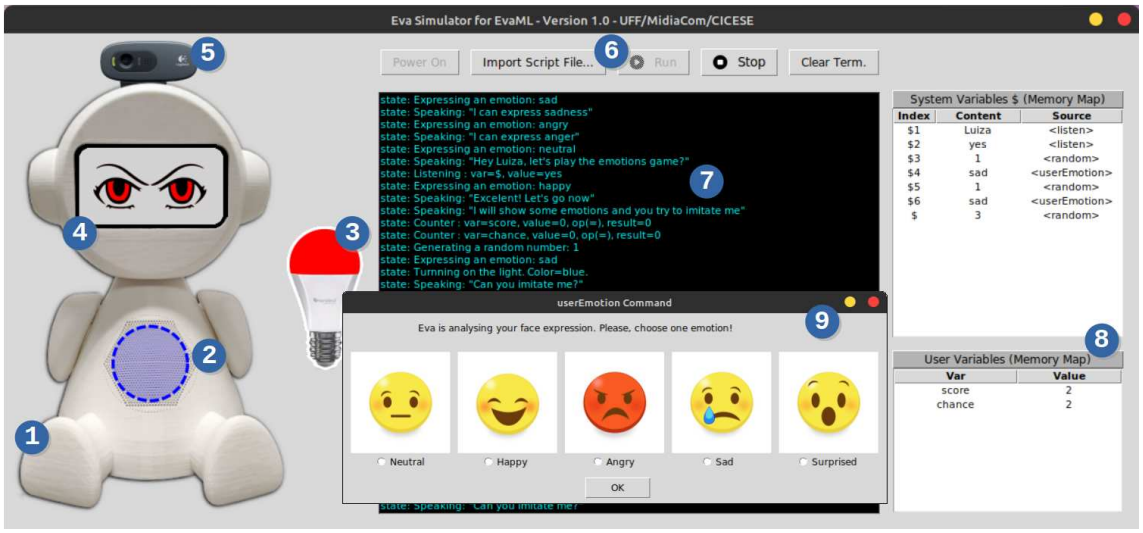

Figura 2.30. Elementos da interface gráfica de usuário do EvaSIM

apresentada a seguir, usando os números indicados dentro dos círculos azuis como referência:

1. Apresenta o corpo do robô EVA impresso em 3D.

2. Representa os LEDs RGB que fazem parte da placa Matrix Voice, localizada no tórax do robô. Esse elemento, como no robô físico, pode assumir várias cores e essas cores variam de acordo com a ação que o robô está realizando no momento.

3. É a representação gráfica da lâmpada inteligente que pode assumir os estados ligado ou desligado, podendo também apresentar as cores definidas no script do robô.

4. Representa a tela de 5,5 polegadas do robô. Este elemento gráfico pode mostrar as expressões do olhar do EVA, apresentando as quatro expressões disponíveis no robô físico.

5. Apresenta a webcam integrada ao robô.

6. O conjunto de botões que controlam o simulador. Da esquerda para a direita, o primeiro botão é aquele que inicia o simulador do robô fazendo com que ele entre em modo *stand by*, aguardando o carregamento de um script. O segundo botão abre uma janela de diálogo de abertura de arquivo, que permite carregar um script. O terceiro botão executa o script carregado. Este botão só é habilitado após o carregamento de um script. O próximo botão interrompe um script em execução e, por último, o botão que limpa o emulador de terminal.

7. Apresenta a emulação de um terminal onde são apresentadas algumas informações importantes, como as ações que estão sendo realizadas pelo robô, detalhes das operações com variáveis, cores e estados da lâmpada inteligente, o texto que o robô está falando, algumas mensagens de alerta e possíveis mensagens de erro no script.

8. Apresenta as tabelas do mapa de memória do simulador. Essas duas tabelas destinam-se a mostrar dinamicamente os valores das variáveis do sistema e do usuário durante a execução do script. A tabela superior mostra os valores das variáveis do sistema que armazenam as respostas obtidas nos processos de interação com usuário, como captura de voz e reconhecimento de expressões faciais. Esse conjunto de variáveis também contém os valores gerados pelo comando de geração de números aleatórios da VPL. Essas variáveis do sistema são indexadas usando o caractere "$". Como a memória do robô está repleta de valores de diferentes origens, a tabela superior possui, além das colunas de índice e conteúdo, uma coluna extra que mostra a origem do valor dessa variável. A segunda tabela apresenta as variáveis criadas pelo desenvolvedor do script, com seus nomes e seus respectivos valores.

9. Esta janela é apresentada sempre que o EvaSIM encontra um comando de reconhecimento facial no script. Ela permite ao usuário escolher, usando o mouse, uma das expressões apresentadas na forma de *emoji*, indicando ao simulador sua expressão facial naquele momento.

2.4.2.1. Simulando o Reconhecimento de Expressões Faciais

Como visto na Figura 2.30, item (9), o EvaSIM simula o processo de reconhecimento facial. Esta função foi implementada utilizando-se uma janela com expressões faciais representadas por emojis. No simulador, as respostas que podem ser fornecidas através da janela que representa as expressões faciais do usuário são: "NEUTRAL", "HAPPY", "ANGRY", "SAD" e "SURPRISED". A execução do comando <userEmotion>, responsável por capturar a expressão do usuário através da webcam, abre uma janela com um conjunto de expressões faciais. O usuário, através do mouse, pode indicar sua expressão facial. Esta resposta é processada pelo simulador da mesma forma que o robô físico. Um pequeno vídeo mostrando a simulação do *Jogo da Imitação* [Rocha et al. 2021] no EvaSIM pode ser encontrado neste link: https://youtu.be/OfGelKZIA9c.

2.4.2.2. Simulando o Reconhecimento de Voz

O EVA pode se comunicar com o usuário por meio de interação por voz, capturando o áudio das respostas e transformando-o em texto usando a API do Google que fica na nuvem. Para facilitar esse processo, dentro do simulador, esse tipo de interação multimodal foi representado por uma caixa de texto, onde o usuário pode responder usando texto escrito ao invés de fala. A Figura 2.31 mostra a simulação do processo de captura de voz do robô.

2.4.2.3. Simulando a Movimentação da Cabeça do Robô

O EvaSIM foi projetado para ser uma ferramenta leve e que exigisse baixo poder computacional para ser executada. Portanto, nenhuma animação sofisticada foi implementada nesta versão 2D do simulador. O robô físico, conforme apresentado na Seção 2.3.4.6,

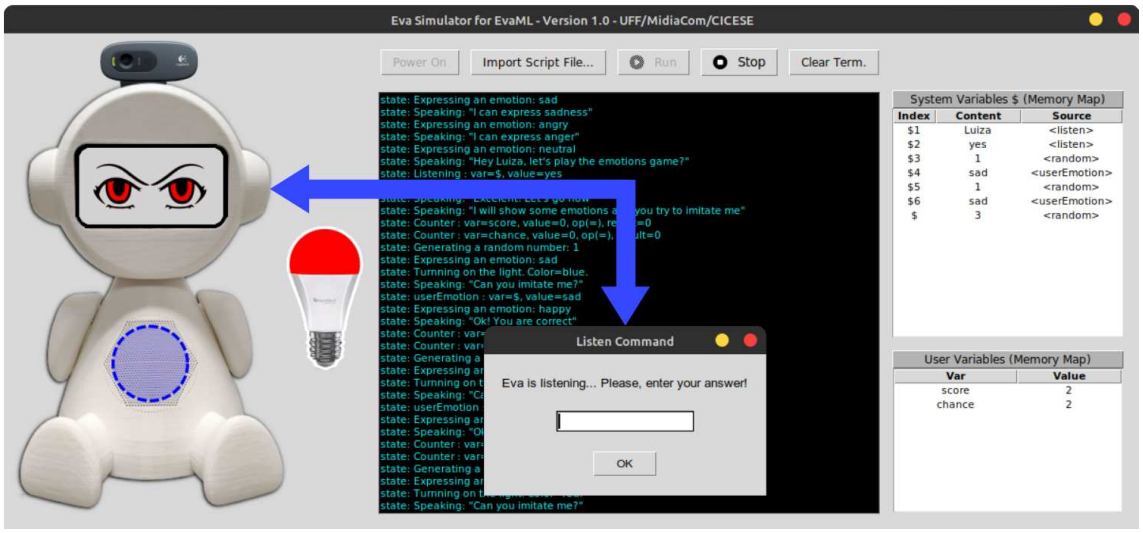

Figura 2.31. Simulando o reconhecimento de voz

pode mover sua cabeça. Este movimento utilizado em conjunto com outros elementos pode aumentar a expressividade do robô. Ao executar um script para o robô e encontrar um elemento <motion>, o EvaSIM utiliza o terminal para indicar que um movimento está sendo executado, mostrando também o tipo de movimento realizado. A Figura 2.32 mostra um exemplo da mensagem no terminal indicando a execução do elemento <motion> e o valor do atributo type.

Figura 2.32. Indicando o movimento da cabeça do robô

A Figura 2.33 apresenta com mais detalhes a emulação de um terminal onde são apresentadas algumas informações importantes sobre a execução do script. Nesse exemplo, pode-se ver a seleção de voz, o estado e a cor da lâmpada inteligente sendo definidos, os textos sendo falados, a captura do nome de usuário através do comando <listen> e a manipulação da variável **x**.

2.5. Atividade Prática

Nesta seção será apresentado um passo-a-passo do como executar o *parsing* de um script EvaML. Também serão mostrados os arquivos que são gerados a partir deste processo. Após o processo de *parsing*, será visto como importar e rodar o código EvaML no simulador EvaSIM.

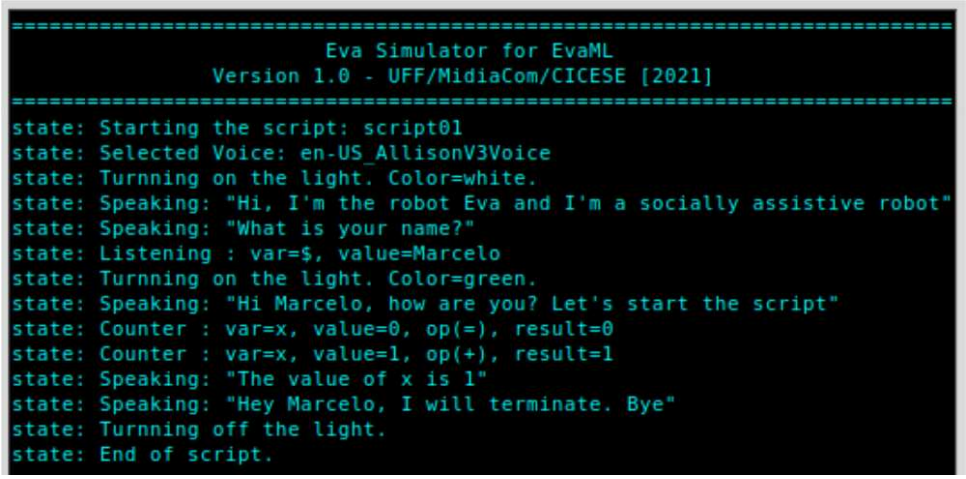

Figura 2.33. Emulador de terminal

2.5.1. Fazendo o *Parsing* de um Script

Antes de dar início ao processo de *parsing*, será apresentada uma IDE muito útil no processo de desenvolvimento dos scripts usando a linguagem EvaML.

Qualquer editor de texto pode ser usado para escrever um script EvaML, mas é altamente recomendado um editor que dê suporte a escrita de código XML e que possua o recurso de validar o código usando como referência um arquivo XML Schema, indicando possíveis inconsistências, na própria janela do editor. Existem vários editores de código disponíveis gratuitamente. A IDE que será utilizada nesta atividade é o *Visual Studio Code*[26]. Para dar o suporte adequado à validação de XML usando um arquivo XML Schema, em tempo real, será preciso instalar a extensão XML *Language Support by Red Hat*. A instalação é muito simples. Acesse a área de extensões no VS Code, que fica na barra de atividades do lado esquerdo da interface do VS Code e clique no ícone Extensões, ou use o atalho do teclado (*Ctrl+Shift+X*). Isso mostrará uma lista das extensões do VS Code. Na barra de pesquisa, digite "XML Language Support by Red Hat". Para instalar uma extensão, selecione o botão Instalar. Depois disso, o VS Code dará suporte a código XML e a validação usando um arquivo XML Schema.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <evaml name="script01" xsi:noNamespaceSchemaLocation="EvaML-Schema/
    evaml_schema.xsd"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
4   <settings>
5   </settings>
6   <script>
7   </script>
8 </evaml>
```

Listagem 2.3. Template de código para a atividade prática

O nome do arquivo do script que será usado neste exemplo é *my_script_file.xml*. Para a execução dos comandos que serão apresentados neste texto, assumiu-se que o

[26]https://code.visualstudio.com/

arquivo do script, o arquivo *eva_parser.py* e o código do *eva_sim.py* estão na mesma pasta.

Para iniciar o processo de *parsing* do arquivo *my_script_file.xml*, é preciso digitar a seguinte linha de comando:

```
python3 eva_parser.py my_script_file.xml -c
```

Se tudo correr bem, poderá ser vista a saída no terminal, como mostra a Figura 2.34. Pode-se ver que todas as 4 etapas do processo de *parsing* foram bem-sucedidas.

```
marcelo@note-mint:~/Dropbox/Estudo XML$ python3 eva_parser.py my_script_file.xml -c
Step 01 - Processing Macros...
Step 02 - Generating Elements keys...
step 03 - Creating the Elements <link>...
step 04 - Mapping XML nodes and links to a JSON file...
marcelo@note-mint:~/Dropbox/Estudo XML$
```

Figura 2.34. Saída no terminal do processo de *Parsing*

Após a execução do *eva_parser.py* tem-se como saída um código XML para o simulador EvaSIM e um código para rodar no robô em formato JSON. Durante o processo de *parsing*, alguns arquivos intermediários são criados, como pode ser visto na Figura 2.35. É importante notar que o nome dos arquivos de saída do tipo EvaML e JSON é baseado no atributo name do elemento raiz <evaml> do elemento *my_script_file.xml* script. Fique atento a isso!

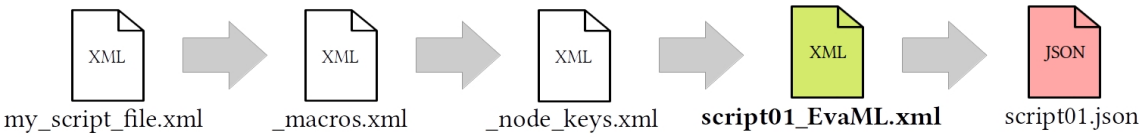

Figura 2.35. Arquivos gerados no processo de *Parsing*

2.5.2. Importando e Executando um Script EvaML

Dando continuidade ao processo que foi iniciado na Seção 2.5.1, quando o processo de *Parsing* do arquivo *my_script_file.xml* gerou o arquivo *script01_EvaML.xml*. O objetivo agora é executar o código EvaML no EvaSIM. A Figura 2.36 mostra a imagem dos botões da interface do simulador.

Os itens citados neste parágrafo são referentes à Figura 2.36. Para começar a usar o EvaSIM você precisa clicar no botão "Power On" (**1**). O EvaSIM falará um texto de saudação e aguardará que um script seja carregado em sua memória. Um script pode ser carregado pressionando-se o botão "Import Script File..." (**2**), que abrirá a caixa de diálogo de abertura de arquivo. Após importar o arquivo, o script poderá ser executado

Figura 2.36. Operando o EvaSIM

clicando-se no botão "Run" (**3**). O botão "Stop" (**4**), se clicado, interrompe a execução do script e o botão "Clear Term." (**5**) limpa o emulador de terminal do EvaSIM.

Agora que já foi apresentado como funciona o EvaSIM, importe o código EvaML gerado pelo parser (*script01_EvaML.xml*) e execute-o, verificando se tudo funciona como esperado.

2.6. Considerações Finais

Este capítulo apresentou o conceito de robôs socialmente assistivos (SARs) usados em terapias de saúde. O texto discutiu características desses robôs, tais como aparência, modalidade de interação, interação inteligente, capacidade de realização de tarefas e modo de operação.

Foram apresentados diversos trabalhos acadêmicos e produtos comerciais de robôs para utilização em terapias, em especial para idosos com doenças neurodegenerativas como Alzheimer e crianças com Transtorno do Espectro do Autismo.

O capítulo deu atenção especial ao robô EVA, que oferece uma plataforma *open-source*, facilitando o desenvolvimento de trabalhos acadêmicos na área. Foram apresentados os componentes de hardware e software do EVA, sua funcionalidade de interação via voz, controle de uma lâmpada inteligente para integração de efeitos de luz na sessão de terapia e também sua capacidade de reconhecimento de expressões faciais do usuário.

O capítulo também apresentou a linguagem EvaML, baseada em XML, para desenvolvimento de sessões de terapia multissensoriais e o simulador EvaSIM, como ferramentas para facilitar o uso do robô em experimentos acadêmicos e até mesmo práticas clínicas. Foram disponibilizados aos leitores um *parser* EvaML e o simulador EvaSIM, de modo a permitir seu uso prático pela comunidade.

Espera-se que este trabalho tenha motivado os leitores a investigar o tema e avançar a pesquisa na área de SARs.

Referências

[Amanatiadis et al. 2017] Amanatiadis, A., Kaburlasos, V. G., Dardani, C., and Chatzichristofis, S. A. (2017). Interactive social robots in special education. In *2017 IEEE 7th international conference on consumer electronics-Berlin (ICCE-Berlin)*, pages 126–129. IEEE.

[Annaz et al. 2009] Annaz, D., Karmiloff-Smith, A., Johnson, M. H., and Thomas, M. S.

(2009). A cross-syndrome study of the development of holistic face recognition in children with autism, down syndrome, and williams syndrome. *Journal of experimental child psychology*, 102(4):456–486.

[Attawibulkul et al. 2019] Attawibulkul, S., Sornsuwonrangsee, N., Jutharee, W., and Kaewkamnerdpong, B. (2019). Using storytelling robot for supporting autistic children in theory of mind. *International Journal of Bioscience, Biochemistry and Bioinformatics*, 9(2):100–108.

[Azuar et al. 2019] Azuar, D., Gallud, G., Escalona, F., Gomez-Donoso, F., and Cazorla, M. (2019). A story-telling social robot with emotion recognition capabilities for the intellectually challenged. In *Iberian Robotics conference*, pages 599–609. Springer.

[Barakova and Lourens 2013] Barakova, E. and Lourens, T. (2013). Interplay between natural and artificial intelligence in training autistic children with robots. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pages 161–170. Springer.

[Bartl-Pokorny et al. 2021] Bartl-Pokorny, K. D., Pykała, M., Uluer, P., Barkana, D. E., Baird, A., Kose, H., Zorcec, T., Robins, B., Schuller, B. W., and Landowska, A. (2021). Robot-based intervention for children with autism spectrum disorder: a systematic literature review. *IEEE Access*.

[Cabibihan et al. 2013] Cabibihan, J.-J., Javed, H., Ang, M., and Aljunied, S. M. (2013). Why robots? a survey on the roles and benefits of social robots in the therapy of children with autism. *International journal of social robotics*, 5(4):593–618.

[Cano et al. 2021] Cano, S., González, C. S., Gil-Iranzo, R. M., and Albiol-Pérez, S. (2021). Affective communication for socially assistive robots (sars) for children with autism spectrum disorder: A systematic review. *Sensors*, 21(15):5166.

[Cao et al. 2015] Cao, H.-L., Pop, C., Simut, R., Furnemónt, R., De Beir, A., Van de Perre, G., Esteban, P. G., Lefeber, D., and Vanderborght, B. (2015). Probolino: A portable low-cost social device for home-based autism therapy. In *International Conference on Social Robotics*, pages 93–102. Springer.

[Cassell et al. 1994] Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, T., Douville, B., Prevost, S., and Stone, M. (1994). Animated conversation: rule-based generation of facial expression, gesture & spoken intonation for multiple conversational agents. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 413–420.

[Chen et al. 2020] Chen, H., Park, H. W., and Breazeal, C. (2020). Teaching and learning with children: Impact of reciprocal peer learning with a social robot on children's learning and emotive engagement. *Computers & Education*, 150:103836.

[Costa et al. 2017] Costa, A. P., Steffgen, G., Lera, F. R., Nazarikhorram, A., and Ziafati, P. (2017). Socially assistive robots for teaching emotional abilities to children with autism spectrum disorder. In *3rd Workshop on Child-Robot Interaction at HRI*.

[Cruz Sandoval 2020] Cruz Sandoval, D. (2020). *Robot conversacional como apoyo a intervenciones no farmacológicas para adultos mayores con demencia Conversational robot to support non-pharmacological interventions for people with dementia*. Tesis de doctorado en ciencias, Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California. 125pp.

[Cruz-Sandoval and Favela 2019] Cruz-Sandoval, D. and Favela, J. (2019). A Conversational Robot to Conduct Therapeutic Interventions for Dementia. *IEEE Pervasive Computing*, 18(2):10–19.

[Cruz-Sandoval et al. 2020] Cruz-Sandoval, D., Morales-Tellez, A., Sandoval, E. B., and Favela, J. (2020). A social robot as therapy facilitator in interventions to deal with dementia-related behavioral symptoms. In *2020 15th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 161–169. IEEE.

[Dickstein-Fischer et al. 2018] Dickstein-Fischer, L. A., Crone-Todd, D. E., Chapman, I. M., Fathima, A. T., and Fischer, G. S. (2018). Socially assistive robots: current status and future prospects for autism interventions. *Innovation and Entrepreneurship in Health*, 5:15–25.

[Duquette et al. 2008] Duquette, A., Michaud, F., and Mercier, H. (2008). Exploring the use of a mobile robot as an imitation agent with children with low-functioning autism. *Autonomous Robots*, 24(2):147–157.

[Elder et al. 2006] Elder, L. M., Caterino, L. C., Chao, J., Shacknai, D., and De Simone, G. (2006). The efficacy of social skills treatment for children with asperger syndrome. *Education and Treatment of Children*, pages 635–663.

[Erden 2013] Erden, M. S. (2013). Emotional postures for the humanoid-robot nao. *International Journal of Social Robotics*, 5(4):441–456.

[Fasola and Mataric 2012] Fasola, J. and Mataric, M. J. (2012). Using socially assistive human–robot interaction to motivate physical exercise for older adults. *Proceedings of the IEEE*, 100(8):2512–2526.

[Feil-Seifer and Mataric 2005] Feil-Seifer, D. and Mataric, M. J. (2005). Defining socially assistive robotics. In *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, pages 465–468. IEEE.

[Fong et al. 2003] Fong, T., Nourbakhsh, I., and Dautenhahn, K. (2003). A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3-4):143–166.

[Gratch et al. 2006] Gratch, J., Okhmatovskaia, A., Lamothe, F., Marsella, S., Morales, M., van der Werf, R. J., and Morency, L.-P. (2006). Virtual rapport. In *International Workshop on Intelligent Virtual Agents*, pages 14–27. Springer.

[Gudlin et al. 2022] Gudlin, M., Ivanković, I., and Dadić, K. (2022). Robots used in therapy for children with autism spectrum disorder. *American Journal of Multidisciplinary Research & Development (AJMRD)*, 4(03):33–39.

[Huskens et al. 2013] Huskens, B., Verschuur, R., Gillesen, J., Didden, R., and Barakova, E. (2013). Promoting question-asking in school-aged children with autism spectrum disorders: Effectiveness of a robot intervention compared to a human-trainer intervention. *Developmental neurorehabilitation*, 16(5):345–356.

[Josué et al. 2020] Josué, M., Montevecchi, E., Abreu, R., Barreto, F., Santos, J., and Muchaluat-Saade, D. C. (2020). Ambientes multissensoriais aplicados à saúde: desenvolvimento de aplicações e tendências futuras. In *Livro de Minicursos do SBCAS 2020*, chapter 2. SBC.

[Li and Chignell 2011] Li, J. and Chignell, M. (2011). Communication of emotion in social robots through simple head and arm movements. *International Journal of Social Robotics*, 3(2):125–142.

[Liu et al. 2012] Liu, C., Ishi, C. T., Ishiguro, H., and Hagita, N. (2012). Generation of nodding, head tilting and eye gazing for human-robot dialogue interaction. In *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 285–292. IEEE.

[Lytridis et al. 2018] Lytridis, C., Vrochidou, E., Chatzistamatis, S., and Kaburlasos, V. (2018). Social engagement interaction games between children with autism and humanoid robot nao. In *The 13th international conference on soft computing models in industrial and environmental applications*, pages 562–570. Springer.

[Manohar et al. 2011] Manohar, V., al Marzooqi, S., and Crandall, J. W. (2011). Expressing emotions through robots: a case study using off-the-shelf programming interfaces. In *2011 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 199–200, Japão. IEEE.

[Marsella et al. 2013] Marsella, S., Xu, Y., Lhommet, M., Feng, A., Scherer, S., and Shapiro, A. (2013). Virtual character performance from speech. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 25–35.

[Martinez-Martin et al. 2020] Martinez-Martin, E., Escalona, F., and Cazorla, M. (2020). Socially assistive robots for older adults and people with autism: An overview. *Electronics*, 9(2):367.

[Matarić et al. 2007] Matarić, M. J., Eriksson, J., Feil-Seifer, D. J., and Winstein, C. J. (2007). Socially assistive robotics for post-stroke rehabilitation. *Journal of neuroengineering and rehabilitation*, 4(1):1–9.

[Mazzei et al. 2011] Mazzei, D., Lazzeri, N., Billeci, L., Igliozzi, R., Mancini, A., Ahluwalia, A., Muratori, F., and De Rossi, D. (2011). Development and evaluation of a social robot platform for therapy in autism. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4515–4518. IEEE.

[Mesibov et al. 2013] Mesibov, G. B., Adams, L. W., and Klinger, L. G. (2013). *Autism: Understanding the disorder*. Springer Science & Business Media.

[Mitjans 2020] Mitjans, A. A. (2020). Affective computation in human-robot interaction. Master thesis, Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California.

[Mordoch et al. 2013] Mordoch, E., Osterreicher, A., Guse, L., Roger, K., and Thompson, G. (2013). Use of social commitment robots in the care of elderly people with dementia: A literature review. *Maturitas*, 74(1):14–20.

[Nestorov et al. 2014] Nestorov, N., Stone, E., Lehane, P., and Eibrand, R. (2014). Aspects of socially assistive robots design for dementia care. In *2014 IEEE 27th International Symposium on Computer-Based Medical Systems*, pages 396–400. IEEE.

[Novák 2010] Novák, M. (2010). Easy implementation of domain specific language using xml. In *Proceedings of the 10th Scientific Conference of Young Researchers (SCYR 2010), Košice, Slovakia*, volume 19.

[Pelachaud et al. 1996] Pelachaud, C., Badler, N. I., and Steedman, M. (1996). Generating facial expressions for speech. *Cognitive science*, 20(1):1–46.

[Pollmann et al. 2019] Pollmann, K., Tagalidou, N., and Fronemann, N. (2019). It's in your eyes: Which facial design is best suited to let a robot express emotions? In *Proceedings of Mensch und Computer 2019*, pages 639–642.

[Ricks and Colton 2010] Ricks, D. J. and Colton, M. B. (2010). Trends and considerations in robot-assisted autism therapy. In *2010 IEEE international conference on robotics and automation*, pages 4354–4359. IEEE.

[Ricks et al. 2010] Ricks, D. J., Colton, M. B., and Goodrich, M. A. (2010). Design and evaluation of a clinical upper-body humanoid robot for autism therapy. In *In Proceedings of the 2010 International Conference on Applied Bionics and Biomechanics, Venice, Italy*, pages 14–16.

[Robinson et al. 2014] Robinson, H., MacDonald, B., and Broadbent, E. (2014). The role of healthcare robots for older people at home: A review. *International Journal of Social Robotics*, 6(4):575–591.

[Rocha et al. 2021] Rocha, M., Valentim, P., Barreto, F., Mitjans, A., Cruz-Sandoval, D., Favela, J., and C., M.-S. D. (2021). Towards enhancing the multimodal interaction of a social robot to assist children with autism in emotion regulation. In *Proceedings of the 15th EAI International Conference on Pervasive Computing Technologies for Healthcare*.

[Saldien et al. 2010] Saldien, J., Goris, K., Vanderborght, B., Vanderfaeillie, J., and Lefeber, D. (2010). Expressing emotions with the social robot probo. *International Journal of Social Robotics*, 2(4):377–389.

[Salichs et al. 2016] Salichs, M. A., Encinar, I. P., Salichs, E., Castro-González, Á., and Malfaz, M. (2016). Study of scenarios and technical requirements of a social assistive robot for alzheimer's disease patients and their caregivers. *International Journal of Social Robotics*, 8(1):85–102.

[Santatiwongchai et al. 2016] Santatiwongchai, S., Kaewkamnerdpong, B., Jutharee, W., and Ounjai, K. (2016). Bliss: Using robot in learning intervention to promote social skills for autism therapy. In *Proceedings of the International Convention on Rehabilitation Engineering & Assistive Technology*, i-CREATe 2016, Midview City, SGP. Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre.

[Scassellati et al. 2012] Scassellati, B., Admoni, H., and Matarić, M. (2012). Robots for use in autism research. *Annual review of biomedical engineering*, 14:275–294.

[Shamsuddin et al. 2012] Shamsuddin, S., Yussof, H., Ismail, L., Hanapiah, F. A., Mohamed, S., Piah, H. A., and Zahari, N. I. (2012). Initial response of autistic children in human-robot interaction therapy with humanoid robot nao. In *2012 IEEE 8th International Colloquium on Signal Processing and its Applications*, pages 188–193. IEEE.

[Shibata 2004] Shibata, T. (2004). An overview of human interactive robots for psychological enrichment. *Proceedings of the IEEE*, 92(11):1749–1758.

[Shibata 2012] Shibata, T. (2012). Therapeutic seal robot as biofeedback medical device: Qualitative and quantitative evaluations of robot therapy in dementia care. *Proceedings of the IEEE*, 100(8):2527–2538.

[Tapus et al. 2007] Tapus, A., Mataric, M. J., and Scassellati, B. (2007). Socially assistive robotics [grand challenges of robotics]. *IEEE robotics & automation magazine*, 14(1):35–42.

[Tapus et al. 2009] Tapus, A., Tapus, C., and Mataric, M. J. (2009). The use of socially assistive robots in the design of intelligent cognitive therapies for people with dementia. In *2009 IEEE international conference on rehabilitation robotics*, pages 924–929. IEEE.

[Ueyama 2015] Ueyama, Y. (2015). A bayesian model of the uncanny valley effect for explaining the effects of therapeutic robots in autism spectrum disorder. *PloS one*, 10(9):e0138642.

[Valentí Soler et al. 2015] Valentí Soler, M., Agüera-Ortiz, L., Olazarán Rodríguez, J., Mendoza Rebolledo, C., Pérez Muñoz, A., Rodríguez Pérez, I., Osa Ruiz, E., Barrios Sánchez, A., Herrero Cano, V., Carrasco Chillón, L., et al. (2015). Social robots in advanced dementia. *Frontiers in aging neuroscience*, 7:133.

[Valentim et al. 2020] Valentim, P. A., Barreto, F., and Muchaluat-Saade, D. C. (2020). Possibilitando o reconhecimento de expressões faciais em aplicações ginga-ncl. In *Anais Estendidos do XXVI Simpósio Brasileiro de Sistemas Multimídia e Web*, pages 53–56. SBC.

[Vanderborght et al. 2012] Vanderborght, B., Simut, R., Saldien, J., Pop, C., Rusu, A. S., Pintea, S., Lefeber, D., and David, D. O. (2012). Using the social robot probo as a social story telling agent for children with asd. *Interaction Studies*, 13(3):348–372.

[Wada et al. 2008] Wada, K., Shibata, T., Musha, T., and Kimura, S. (2008). Robot therapy for elders affected by dementia. *IEEE Engineering in medicine and biology magazine*, 27(4):53–60.

[Woods et al. 2021] Woods, D., Yuan, F., Jao, Y.-L., and Zhao, X. (2021). Social robots for older adults with dementia: A narrative review on challenges & future directions. In *International Conference on Social Robotics*, pages 411–420. Springer.

[Xiao et al. 2020] Xiao, W., Li, M., Chen, M., and Barnawi, A. (2020). Deep interaction: Wearable robot-assisted emotion communication for enhancing perception and expression ability of children with autism spectrum disorders. *Future Generation Computer Systems*, 108:709–716.

[Yabuki and Sumi 2018] Yabuki, K. and Sumi, K. (2018). Learning support system for effectively conversing with individuals with autism using a humanoid robot. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 4266–4270. IEEE.

[Yang et al. 2018] Yang, G.-Z., Bellingham, J., Dupont, P. E., Fischer, P., Floridi, L., Full, R., Jacobstein, N., Kumar, V., McNutt, M., Merrifield, R., et al. (2018). The grand challenges of science robotics. *Science robotics*, 3(14):eaar7650.

