UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Nícolas de Carvalho Gehm

ROBÔ HUMANOIDE BEO: UMA PROPOSTA DE FERRAMENTA PARA O ENSINO DE PROGRAMAÇÃO À CRIANÇAS

Santa Maria, RS
2018

Nícolas de Carvalho Gehm

ROBÔ HUMANOIDE BEO: UMA PROPOSTA DE FERRAMENTA PARA O ENSINO DE PROGRAMAÇÃO À CRIANÇAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Bacharel em Engenharia de Computação.

ORIENTADOR: Prof. Rodrigo da Silva Guerra

Santa Maria, RS
2018

Gehm, Nícolas de Carvalho
ROBÔ HUMANOIDE BEO: UMA PROPOSTA DE FERRAMENTA PARA O ENSINO DE PROGRAMAÇÃO À CRIANÇAS / Nícolas de Carvalho Gehm.- 2018.
71 p.; 30 cm

Orientador: Rodrigo da Silva Guerra
Tese (livre-docência) - Universidade Federal de Santa Maria, Centro de Tecnologia, RS, 2018

1. Robótica 2. Robô humanoide 3. Robótica Educacional 4. Ensino de Programação I. Guerra, Rodrigo da Silva II. Título.

Sistema de geração automática de ficha catalográfica da UFSM. Dados fornecidos pelo autor(a). Sob supervisão da Direção da Divisão de Processos Técnicos da Biblioteca Central. Bibliotecária responsável Paula Schoenfeldt Patta CRB 10/1728.

End. Eletr.: nicolas.gehm@gmail.com

Nícolas de Carvalho Gehm

ROBÔ HUMANOIDE BEO: UMA PROPOSTA DE FERRAMENTA PARA O ENSINO DE PROGRAMAÇÃO À CRIANÇAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Bacharel em Engenharia de Computação.

Aprovado em 20 de dezembro de 2018:

Rodrigo da Silva Guerra, Dr. (UFSM)
(Presidente/Orientador)

Fernando Tello Gamarra, Dr. (UFSM)

Susana Cristina dos Reis, Dr. (UFSM)

Santa Maria, RS
2018

AGRADECIMENTOS

Agradeço à Universidade Federal de Santa Maria por ter me acolhido e dado todo suporte necessário para que chegasse aonde estou hoje. À Engenharia de Computação e seu corpo docente, direção e secretaria pelo apoio e compreensão ao longo dos anos de graduação. À Coordenadoria de Ações Educacionais e todos seus profissionais que me ajudaram nessa caminhada de graduação. Agradeço à empresa Qiron Robotics, que me deu a oportunidade de crescer profissionalmente e desenvolver este trabalho baseado no robô Beo e ao meu professor orientador Rodrigo da Silva Guerra, pelo apoio no desenvolvimento deste trabalho e compreensão nos momentos de dificuldade.

Agradeço a todas as pessoas que fizeram parte dessa caminhada de alguma maneira ou outra, em especial a minha companheira e namorada Camilla Compagnoni a qual sempre me apoiou e se fez presente de alguma maneira. Aos meus amigos pelo incentivo e amizade presentes todos os dias. Aos meus pais e irmã, pelo amor, suporte, compreensão e apoio incondicional sempre presente, mesmo nos momentos mais difíceis. Gratidão pelas oportunidades, crescimento pessoal e profissional e conhecimento adquirido ao longo desses anos de graduação.

Você nunca sabe que resultados virão da sua ação. Mas se você não fizer nada, não existirão resultados.

(Mahatma Gandhi)

RESUMO

ROBÔ HUMANOIDE BEO: UMA PROPOSTA DE FERRAMENTA PARA O ENSINO DE PROGRAMAÇÃO À CRIANÇAS

AUTOR: Nícolas de Carvalho Gehm
ORIENTADOR: Rodrigo da Silva Guerra

Este trabalho apresenta o Beo, robô humanoide móvel com nove graus de liberdade. Beo foi construído para ser utilizado no ensino de programação à crianças com intuito de disponibilizar uma ferramenta que permita ao aluno aprofundar seu conhecimento sem as limitações existentes na montagem de equipamentos com a utilização de kits. São apresentados alguns trabalhos desenvolvidos no meio acadêmico relacionados ao tema, descrevendo diferentes ferramentas para serem utilizadas no ensino de programação. As funcionalidades do robô são apresentadas e discutidas. Por fim, espera-se que a utilização desta ferramenta no ensino de programação seja capaz de atender ao seu propósito inicial de oferecer uma plataforma simples e intuitiva para iniciantes, mas que permita o aprofundamento dos conhecimentos na área de programação para usuários mais experientes.

Palavras-chave: Robótica. Robô humanoide. Robótica Educacional. Ensino de Programação.

ABSTRACT

HUMANOID ROBOT BEO: A TOOL PROPOSAL FOR CHILDREN TO LEARN CODING

AUTHOR: Nícolas de Carvalho Gehm
ADVISOR: Rodrigo da Silva Guerra

This work presents Beo, a mobile humanoid robot with 9 degrees of freedom. Beo was built to be used to teach coding to children and young people with the intention of providing a tool that allows the student to deepen his or her knowledge without the limitations of having to assemble equipment with the use of kits. Some related works developed in the academic environment are presented, describing different tools to be used to teach programming. The robot functionalities are presented and discussed. Finally, it is expected that the use of this tool in teaching programming will be able to meet its initial purpose of offering a simple and intuitive platform for beginners, but that allows the deepening of knowledge in the programming area for more experienced users.

Keywords: Robotics. Humanoide Robot. Educacional Robotics. Programming teaching.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

bit	*Binary Digit*
$byte$	*Binary Term*
CAD	*Computer Aided Design*
CRC	*Cyclic Redundancy Check*
CSI	*Camera Serial Interface*
DC	*Direct Current*
DSI	*Display Serial Interface*
$DTMF$	*Dual Tone Multi Frequency*
$GPIO$	*General Purpose Input/Output*
$HDMI$	*High Definition Multimedia Interface*
$HTML5$	*HyperText Markup Language 5*
I^2C	*Inter-integrated Circuit*
IDE	*Integrated Development Environment*
IR	*Infrared*
LED	*Light Emitting Diode*
$LIRC$	*Linux Infrared Remote Control*
MIT	*Massachusetts Institute of Technology*
$OLED$	*Organic Light-Emitting Diode*
PLA	*Polylactic Acid*
RAM	*Random Access Memory*
RPG	*Role-playing Game*
$UART$	*Universal Asynchronous Receiver/Transmitter*
USB	*Universal Serial Bus*
VPL	*Visual Programming Language*
uSD	*Micro Secure Digital*

LISTA DE SÍMBOLOS

GB	Gigabytes
$Mbps$	*Mega Bits per Second*
MHz	Megahertz
mm	Milímetros
rpm	Rotações por minuto
V	Volts

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, no contexto acadêmico percebe-se um esforço que vem sendo feito pela comunidade acadêmica quanto ao desenvolvimento de ambientes de aprendizagem e programação que tornem o aprendizado mais fácil, agradável e eficiente (CARBAJAL; BARANAUSKAS, 2015). Esse esforço iniciou-se com o surgimento da linguagem Logo, criada por Papert, na década de 1960, representando a primeira linguagem de programação elaborada com intuito de ser utilizada por crianças (PAPERT, 1980). Após esta, um trabalho de grande impacto em desenvolvimento é o uso da proposta Scratch, pelo grupo de pesquisa do MIT, cujo principal objetivo, de acordo com Maloney et al. (2010), é se apresentar como ferramenta de introdução à programação para as pessoas que nunca tiveram experiência alguma programando, através da utilização de uma linguagem visual de blocos, mais intuitiva e de fácil compreensão.

Além disso, muitas outras ferramentas vêm sendo desenvolvidas e estudadas para aplicação em contextos de ensino, desde a criação de novas linguagens de programação, até a utilização de mecanismos físicos como robôs e kits de eletrônica. Há uma grande quantidade de estudos e trabalhos sendo desenvolvidos na comunidade científica na área, com a preocupação em fornecer instrumentos para o ensino de algoritmos e de programação que sejam cada vez mais didáticos e atraentes aos alunos.

Com isso, surge a robótica educacional, que é uma área em expansão tanto no contexto acadêmico quanto escolar. De acordo com diversos pesquisadores, a robótica poderá se tornar uma ferramenta para contribuir para uma educação inovadora e libertadora (FREIRE; SHOR, 2008). O potencial da tecnologia disponível nos kits robóticos auxilia no desenvolvimento de práticas pedagógicas, pois resgata motivação, colaboração e a formação de competências tão essenciais à construção do conhecimento. A robótica cumpre a função de despertar o desejo de aprender e propiciar a utilização de diversos conceitos de várias disciplinas, como “[...] a Matemática, [...] Ciências, [...] Língua Portuguesa [...] Expressões dramáticas [...]” (RIBEIRO, 2006), visando ampliar a produção do conhecimento de maneira autônoma e interdisciplinar.

Percebe-se uma evolução em relação às ferramentas que vem sendo desenvolvidas para respectiva aplicação no ensino de programação para as crianças. Inicialmente, com o surgimento das primeiras linguagens de programação, não havia preocupação com o aprendizado das crianças nessa área, o que começou a mudar com

Segundo Alimisis (2013), as principais teorias por trás da Robótica Educacional são o construcionismo e construtivismo. Ackermann (2004) ratifica que ambas teorias andam lado a lado e apresentam importâncias equivalentes no que diz respeito à aprendizagem humana.

Para Ackermann (2001), o Construtivismo, teoria de aprendizagem proposta por

Jean Piaget, explica que o conhecimento nada mais é que as experiências adquiridas pelo indivíduo através da interação deste com o mundo, as pessoas e as coisas, e não simplesmente uma informação transmitida a partir de um ponto e entregue a outro, sendo codificada, memorizada e recuperada.

Alimisis (2013) explica que a teoria de Piaget fala que a chave para as crianças construirem seu conhecimento é a partir da manipulação de artefatos, ideia à qual Papert adicionou que a construção do conhecimento ocorre especialmente em contextos em que a criança está engajada na construção do artefato, seja ele um castelo de areia na praia ou um artefato tecnológico.

No Brasil, Albuquerque et al. (2007) introduz o conceito de Robótica Pedagógica Livre. A proposta consiste em utilizar-se de robôs e princípios de construtivismo no processo pedagógico de construção do conhecimento. A Robótica Pedagógica Livre se diferencia pela sua ênfase no emprego de soluções livres, incluindo o uso de *softwares* livres (Linux e seus aplicativos), e incentivando o uso da sucata de equipamentos eletrônicos para a construção de kits.

O uso de robôs físicos, e não apenas jogos virtuais, permite ao jovem experimentar em primeira pessoa, tocando e manipulando objetos reais. Nesse tipo de aprendizagem, reduz-se ou até elimina-se a necessidade de utilização de símbolos, gravuras, animações que, indiretamente, representam um conceito ou ideia. O próprio robô se manifesta diretamente, representando sua própria essência. Por meio da brincadeira, manipulando o robô, e executando tarefas e desafios, o jovem se envolve de forma multimodal, e em comunidade, permitindo assim um processo mais natural de aquisição de conhecimento.

Neste trabalho descrevemos o desenvolvimento do Beo, um robô didático projetado para aplicações voltadas ao ensino de programação e robótica à crianças já alfabetizadas, sendo assim capazes de ler e escrever, ou seja, crianças entre seis e sete anos de idade, aproximadamente. Em contraste à proposta da Robótica Pedagógica Livre, o Beo não é feito de sucata e não é oferecido na forma de kit. O Beo é um robô humanoide sofisticado, que vem pronto para ser programado.

No Brasil, em diversas escolas, a prática de robótica vem surgindo como atividade extra-curricular cada vez em séries iniciais. Em geral, esses projetos envolvem tarefas em que os jovens exploram princípios muito básicos de eletrônica, mecânica e programação para criação de robôs utilizando sucata ou kits de montagem (ou ainda kits híbridos). Entretanto, observa-se que, apesar da grande capacidade da criança, há uma grande barreira que limita a complexidade do equipamento que essa criança consegue construir com as próprias mãos. As limitações não são apenas ligadas à falta de conhecimentos avançados de projeto mecânico e de *software*, mas também estão relacionadas ao custo e ao tempo disponível para a atividade extra-curricular.

Em sua concepção, o Beo busca ocupar esse nicho, em que a criança que aprendeu conceitos básicos de mecânica, eletrônica e programação, possa dar continuidade à sua

aprendizagem, aprofundando seu conhecimento sem precisar estar limitado àquilo que ele mesmo constrói através dos kits. Como analogia, comparamos com o caso de crianças com talento para música. Certamente, pode-se ensiná-lo como funciona o princípio básico da acústica de um violão permitindo à criança construir seu próprio instrumento utilizando kits, ou materiais reciclados. Entretanto, após adquirir essa experiência inicial, para que o jovem avance de forma séria em sua habilidade musical, seria preferível que a criança tivesse acesso a um instrumento real, em toda sua plenitude. Esse é um exemplo bastante ilustrativo, pois conhecemos diversos casos de crianças que demonstram impressionante capacidade musical.

Além de ser uma ferramenta sofisticada, o Beo também apresenta uma forma antropomórfica. Esse antropomorfismo permite ao jovem se relacionar com o robô como um "alguém", ao invés de um "algo". Com cabeça, braços e expressões faciais, ele personifica um personagem. Um urso de pelúcia representa um urso, e uma boneca representa um bebê ou uma mulher. Um carrinho representa um automóvel, e um boneco representa um super-herói. Mais que um simples boneco, o Beo é o próprio robô que ele representa: ele é o próprio robô, ou seja, representa a si próprio, sua própria essência.

O robô Beo é um projeto da Qiron Robotics, empresa incubada dentro da Universidade Federal de Santa Maria, localizada em Santa Maria, Rio Grande do Sul. A empresa trabalha no desenvolvimento de robôs e no ensino de programação e robótica para crianças. O projeto Beo foi iniciado em 2016, período em que o autor entrou na empresa como estagiário e fez parte do grupo de desenvolvimento do robô humanoide. Além da atuação no planejamento e programação do robô, o autor atuou como instrutor nas aulas de programação e robótica ministradas, pela empresa, para crianças, utilizando o próprio robô como ferramenta de ensino em algumas aulas.

1.1 JUSTIFICATIVA

Aparentemente, o ensino de programação atualmente, é o ensino de inglês de décadas atrás. Uma habilidade indispensável hoje em dia para quem quer concorrer a vagas de emprego em diversas empresas. Os avanços tecnológicos cada vez mais rápidos em nossa sociedade fazem com que muitos empregos sumam e muitos outros novos surjam de tempos em tempos, como é o caso dos trabalhadores da indústria, os quais, em muitos casos, foram substituídos por máquinas. De acordo com um estudo realizado pela consultoria PwC (2017), a robótica e a inteligência artificial podem eliminar uma boa parte dos empregos em um futuro próximo. Esse estudo mede o potencial impacto que a automação terá sobre no Reino Unido e em outras grandes economias, avaliando se robôs irão roubar empregos atualmente ocupado por pessoas.

Porém, essas mudanças no mercado de trabalho trazem novas oportunidades, no-

vas atividades e profissões, pois, por exemplo, as máquinas precisam sair de algum lugar. Torna-se então necessário que haja alguém com conhecimento técnico suficiente para produzir tais máquinas e programá-las. Hoje em dia, a maioria dos componentes eletrônicos que as pessoas têm em casa possuem algum tipo de programação e cada vez mais utensílios, antes apenas mecânicos, trazem um sistema embarcado consigo atualmente. O mercado de trabalho está mudando, o que exige que o profissional mude também, se prepare de uma maneira diferente, adquira novas habilidades.

A ideia, portanto, é começar a preparar esse profissional desde sua infância, introduzindo a ele lógica e conceitos de da área de programação. De acordo com o Fórum Econômico Mundial, 65% das crianças hoje no ensino fundamental vão exercer uma função que ainda não existe (FORUM, 2016). Conforme Bellizia (2017), *General Manager* na Microsoft Brasil, muitos dos empregos que ainda não existem nascerão nas áreas de ciência, tecnologia, engenharia e matemática e, para minimizar impacto sobre os empregos que correm o risco de serem substituídos, novas habilidades e conhecimentos precisam ser desenvolvidos. Bellizia (2017) traz ainda que, a Microsoft vem tomando diversas ações de apoio à educação, especialmente no que diz respeito ao aprendizado de programação, para crianças, adolescentes e adultos.

Portanto, uma maneira que pode ajudar a amenizar os impactos sobre os empregos que serão automatizados é o ensino de programação às crianças, ajudando a prepará-las para o futuro, quando forem disputar vagas de emprego ou até mesmo entrar em uma instituição de ensino superior em um contexto social que será incerto. Esse ensino precisa se tornar uma experiência rica e cativante para os alunos, pois as novas gerações estão rodeadas por tecnologia e aprendem de uma maneira totalmente digital, o que muda o papel da tecnologia dentro da sala de aula **??**.

1.2 OBJETIVOS

1.2.1 Objetivos Gerais

Propor uma ferramenta para o ensino de programação para crianças, já alfabetizadas, de forma completa, ou seja, que não possua a limitação do artefato construído pela criança quando essa utiliza um kit de eletrônica/robótica. Partindo da ideia de manipulação de artefatos de Piaget, apresentar também uma ferramenta que seja física, palpável, assim como são os kits e robôs, possibilitando dessa forma às crianças, a manipulação de objetos reais durante o processo de aprendizagem.

1.2.2 Objetivos Específicos

Planejar, montar, programar e testar um robô humanoide sobre rodas, com sensores e atuadores que possam ser controlados por crianças em aulas de programação. Disponibilizar uma maneira de controlar a ferramenta de forma lúdica e intuitiva para as crianças, ou seja, oferecer junto à ferramenta uma interface de programação que utilize uma linguagem de programação visual.

2 REVISÃO BIBLIOGRÁFICA

No desenvolvimento da ferramenta completa, inúmeras áreas do conhecimento se fizeram presentes. Porém, em relação aos conhecimentos necessários à parte que foi especificamente desenvolvida pelo autor foram, principalmente, conhecimentos relacionados à protocolos de comunicação, tanto a nível de *software* quando de *hardware*. Além disso, como a proposta da ferramenta é ser utilizada no ensino de programação à crianças, é interessante também entender o funcionamento básico de um ambiente de programação com suporte à linguagem visual. Portanto, esses dois assuntos são abordados nessa seção do trabalho.

2.1 FUNDAMENTAÇÃO TEÓRICA

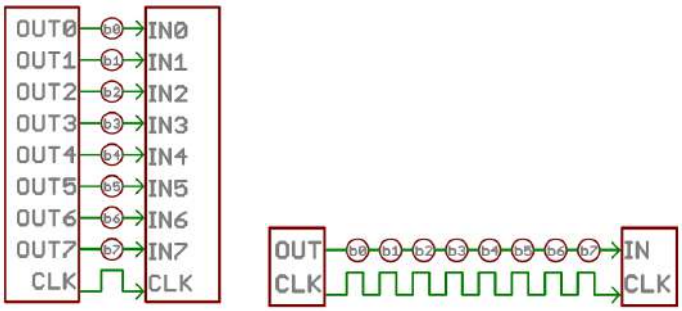
2.1.1 Protocolos de Comunicação

Segundo Stallings (2007), um protocolo de comunicação é arquitetado em camadas de forma a suportar a troca de dados entre sistemas, em que cada uma das camadas possui seu próprio protocolo, com regras bem definidas para a transmissão dos dados. As camadas são organizadas em um pilha vertical, sendo cada nível da pilha responsável por um subconjunto de tarefas do conjunto completo de funções necessárias para realizar a comunicação por completo. Idealmente, cada camada é totalmente independente das demais, ou seja, mudanças feitas em uma delas não exigirão mudanças nas demais. Ainda segundo Stallings (2007), um protocolo é um conjunto de regras ou convenções que determinam como são formatados os dados a serem transmitidos em uma comunicação entre dois sistemas, dos quais ambos devem ser compostos pelo mesmo conjunto de camadas.

A troca de informações, utilizando um protocolo, pode ser realizada de maneira serial ou paralela, como apresentado na Fig. 2.1, e de forma síncrona ou assíncrona. No caso de comunicações síncronas, há um sinal de *clock* compartilhado para sincronização do sinal transmitido com o recebido, o que não ocorre na comunicação assíncrona, a qual é realizada sem suporte de um sinal de *clock* externo. Contudo, a ausência de um sinal de relógio significa um esforço maior na confiabilidade do sistema. Cada tipo de modo de comunicação pode ser separado ainda de acordo com o sentido de transmissão dos dados, ou seja, se o dispositivo pode transmitir e receber dados simultaneamente ou não. No caso de dispositivos que podem apenas exercer uma das funções, receber ou transmitir dados, são denominados *Simplex*. No caso em que os dispositivos podem tanto receber quanto enviar dados, executando uma tarefa por vez, chamam-se dispositivos *Half-duplex*

e, os dispositivos que são capazes de exercer as duas funções de enviar e receber dados simultaneamente recebem o nome de *Full-duplex*.

Figura 2.1 – À esquerda, interface paralela, transferindo vários bits ao mesmo tempo. À direita, interface serial, transmitindo um bit por vez



Fonte: (LINDBLOM, 2012)

Entre os principais protocolos utilizados pelo robô Beo estão UART, I²C e Dynamixel 2.0. O protocolo UART, tem sua comunicação de forma serial assíncrona *Full-duplex*. Sua sigla significa *Universal Asynchronous Receiver/Transmitter*, tem sua denominação de universal indicando que o formato dos dados transmitidos assim como a taxa com que são enviados é configurável. Já o protocolo I²C apresenta comunicação serial síncrona *Half-duplex*, cuja sigla significa *Inter-integrated Circuit*, requisita dois meios físicos para transmitir dados e é utilizado para curtas distâncias. Em especial, haverá um foco maior no protocolo Dynamixel 2.0 devido ao tempo dedicado a ele na produção do robô.

2.1.1.1 Protocolo Dynamixel 2.0

Protocolo de comunicação serial assíncrono Half-duplex, o Protocolo Dynamixel 2.0 foi desenvolvido pela empresa Robotis para ser utilizado na comunicação de qualquer dispositivo com os motores Dynamixel produzidos pela mesma, cuja aplicação é focada no âmbito da robótica. O protocolo funciona de acordo com um sistema de mestre-escravo, no qual os motores representam o lado escravo e apenas irão iniciar algum tipo de transmissão de dados no meio de comunicação apenas se for solicitado por um mestre. A transmissão de dados aos motores é feita através de pacotes de instrução, os quais representam todo e qualquer comando enviado para os dispositivos motores. Sua estrutura é apresentada na imagem 2.2, a qual mostra sua composição e a ordem com que as informações são organizadas dentro do pacote.

O atributo "Header" indica o início do pacote de dados, o "Packet ID" contém o número identificador do disposivito ao qual a mensagem é destinada, suportando um total de 253 valores distintos, considerando que os valores 253 e 255 não são usados e o valor 254 indica que é uma mensagem para todos os dispositivos conectados ao barramento de

Figura 2.2 – Estrutura do pacote de instruções

Header1	Header2	Header3	Reserved	Packet ID	Length1	Length2	Instruction	Param	Param	Param	CRC1	CRC2
0xFF	0xFF	0xFD	0x00	ID	Len_L	Len_H	Instruction	Param 1	...	Param N	CRC_L	CRC_H

Fonte: (Robotis INC, 2018)

dados. "Packet Length" informa o número de campos que a mensagem possui a partir de si mesmo, contabilizando portanto o código da instrução, presente em "Instruction", o número de parâmetros utilizados pela instrução mais os campos com valores para os cálculos do CRC, o qual verifica se o pacote foi corrompido durante sua transmissão. É importante falar que cada dos campos do pacote de instruções é composto por dados de 8 bits cada.

As instruções que os motores aceitam são apresentadas, com seus respectivos valores em hexadecimal e descrição, na Fig. 2.3.

Figura 2.3 – Instruções existentes no protocolo Dynamixel 2.0

Value	Instructions	Description
0x01	Ping	Instruction that checks whether the Packet has arrived to a device with the same ID as Packet ID
0x02	Read	Instruction to read data from the Device
0x03	Write	Instruction to write data on the Device
0x04	Reg Write	Instruction that registers the Instruction Packet to a standby status; Packet is later executed through the Action command
0x05	Action	Instruction that executes the Packet that was registered beforehand using Reg Write
0x06	Factory Reset	Instruction that resets the Control Table to its initial factory default settings
0x08	Reboot	Instruction to reboot the Device
0x10	Clear	Instruction to reset certain information
0x55	Status(Return)	Return Instruction for the Instruction Packet
0x82	Sync Read	For multiple devices, Instruction to read data from the same Address with the same length at once
0x83	Sync Write	For multiple devices, Instruction to write data on the same Address with the same length at once
0x92	Bulk Read	For multiple devices, Instruction to read data from different Addresses with different lengths at once
0x93	Bulk Write	For multiple devices, Instruction to write data on different Addresses with different lengths at once

Fonte: (Robotis INC, 2018)

Uma vez recebido um pacote de instrução, os motores executam o comando contido nele e retornam, sempre, um pacote de status, o qual irá conter qualquer informação solicitada pelo mestre ou uma resposta padrão informando que a mensagem foi recebida e executada com sucesso. O padrão deste pacote é visto em 2.4.

Figura 2.4 – Estrutura do pacote de status

Header1	Header2	Header3	Reserved	Packet ID	Length1	Length2	Instruction	ERR	PARAM	PARAM	PARAM	CRC1	CRC2
0xFF	0xFF	0xFD	0x00	ID	Len_L	Len_H	Instruction	Error	Param 1	...	Param N	CRC_L	CRC_H

Fonte: (Robotis INC, 2018)

Aqui novamente há o campo "Header", que é responsável pela indicação do início

da transmissão de um conjunto de dados; "Packet ID", identificando o motor que está respondendo a mensagem e o campo "Instruction", informando que este pacote é uma resposta à instrução presente neste campo. Além desses campos padrões, há agora o parâmetro "Error" para indicar qualquer falha que tenha ocorrido durante o processo de comunicação, seja um pacote recebido corrompido ou com algum dado incorreto ou em um formato inesperado. E, por último, as regiões destinadas a parâmetros, caso a mensagem necessite acrescentar algum dado solicitado ao motor e os campos CRC's.

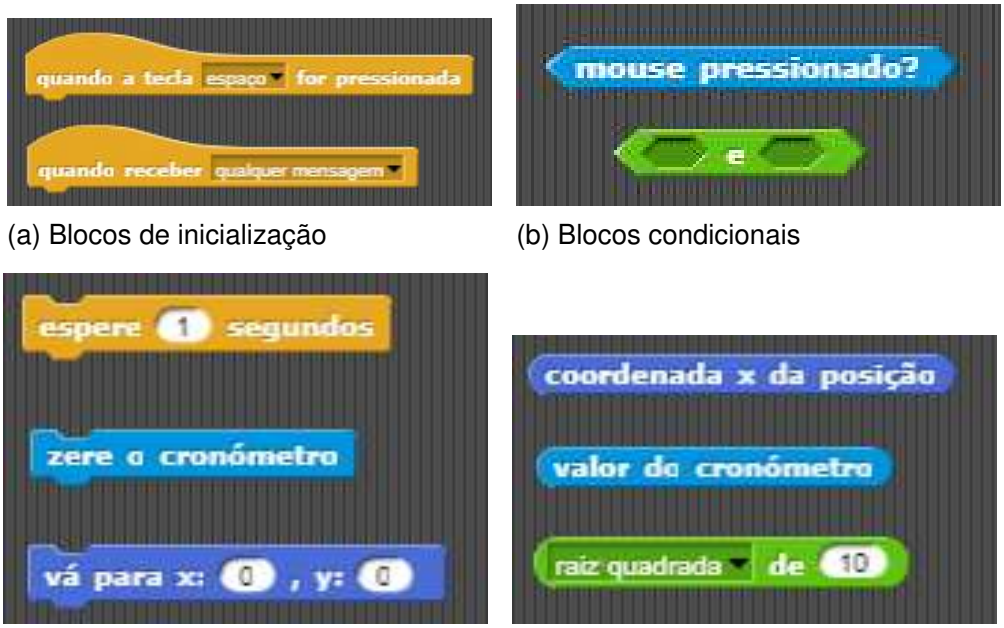
2.1.2 *Software* Snap!: Ambiente de programação para crianças

O programa Snap! é um ambiente de desenvolvimento de algoritmos voltado à introdução de conceitos de programação às crianças. Ele é uma re-implementação estendida do programa Scratch, o qual foi desenvolvido por um grupo vinculado ao Instituto de Tecnologia de Massachusetts (MIT) e utiliza blocos como linguagem de programação (HARVEY; MöNIG,). A principal diferença entre os dois *softwares* é que a re-implementação utiliza HTML5 ao invés de *Adobe Flash Player*. *Adobe Flash Player* é uma plataforma multimídia de desenvolvimento de aplicações que contenham animações, áudio e vídeo, necessária para a execução do Scratch. *Adobe Flash Player* só está disponibilizado para plataformas com arquitetura compatível com o Intel $x86$, motivando a escolha do Snap! como alternativa para este projeto, já que a placa controladora do robô Beo não utiliza arquitetura Intel. Porém, na versão Scratch 3.0 que, ainda está sob desenvolvimento, e, não foi lançada oficialmente, o *Adobe Flash Player* será substituído por HTML5 e, portanto, o Scratch poderá vir a ser utilizado no humanoide Beo em um futuro próximo.

A partir desse momento, o funcionamento do Snap! e suas funcionalidades serão apresentadas com base no trabalho de Harvey e Mönig (), no qual é apresentado um manual completo da ferramenta. De acordo com o autor, pode-se dividir os comandos/blocos em oito categorias: Movimentos, Aparência, Som, Caneta, Controle, Sensores, Operadores e Variáveis. Cada categoria possui uma cor diferente associada a seus blocos, assim como contém diferentes tipos de blocos relacionados ao seu tema, os quais por sua vez possuem formatos diferentes, cada um de acordo com a sua função. Basicamente existem quatro formatos diferentes de blocos e, portanto, quatro funções básicas diferentes. A imagem a seguir apresenta os quatro tipos diferentes de formato de blocos.

Os blocos da categoria mostrada na Fig. 2.5(a) indicam a condição que fará com que o programa seja inicializado, ou seja, os demais blocos conectados a eles serão executados apenas após a condição destes primeiros ser satisfeita, dando partida à execução dos blocos em sequência. Os blocos da Fig. 2.5(b) são condicionais, ou seja, eles sempre indicarão se uma condição é verdadeira ou falsa. Como, por exemplo, os blocos da imagem verificam se o mouse foi pressionado ou não, retornando verdadeiro caso afirmativo

Figura 2.5 – Os diferentes formatos de blocos



(a) Blocos de inicialização (b) Blocos condicionais

(c) blocos de comando (d) Blocos de dados

Fonte: Elaborada pelo autor

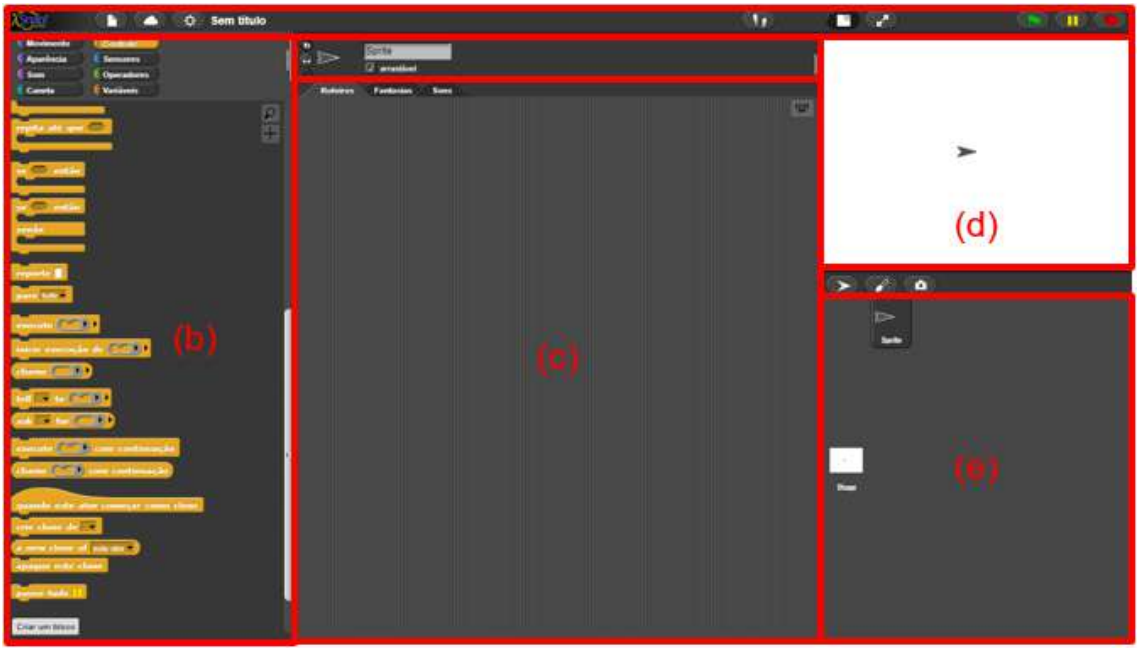
e falso no caso negativo. A Fig. 2.5(c) apresenta blocos de comandos, as quais, assim como o nome sugere, são blocos que executam ações dentro do contexto da ferramenta. Por último, os blocos de formato arredondado presentes em (d) contêm informações e as reportam sempre que solicitadas, semelhante ao comportamento de variáveis de memória em linguagens de programação textuais como C++ e Python.

O ambiente de desenvolvimento pode ser separado em cinco partes distintas: barra de ferramentas, paleta de blocos, área destinada à programação, palco e local de visualização de atores e palco. A seguir, na Fig. 2.6, é visualmente definida a localização de cada uma dessas cinco regiões.

A IDE tem sua estruturação baseada em atores e palco, ou seja, toda programação que é feita, é relacionada aos atores ou ao palco, no qual esses atores se encontram. Um programa desenvolvido por essa plataforma pode ter inúmeros atores, porém sempre possuirá apenas um palco. Cada um desses membros do programa possui sua própria programação e seu próprio fenótipo, ou seja, sua própria aparência, a qual pode ser desenhada livremente ou pode ser representada por alguma imagem pronta, através do *upload* da mesma ao programa.

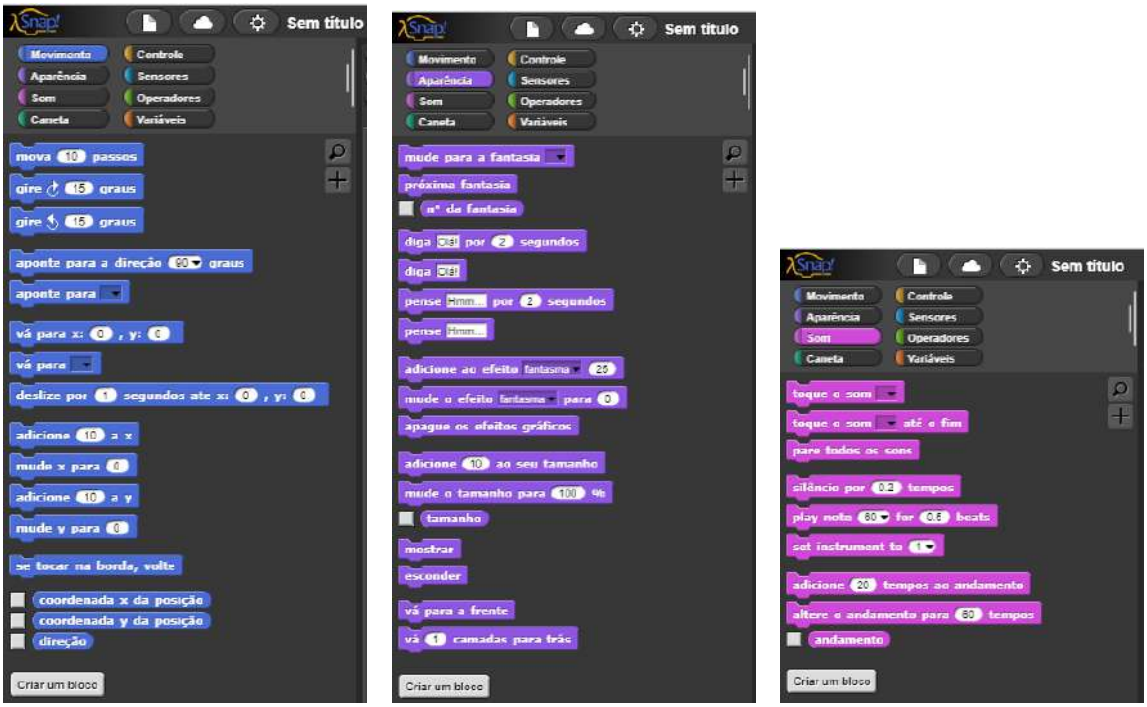
Os blocos das categorias Movimentos, Aparência e Som, que já vêm por padrão na ferramenta são mostrados na imagem 2.7. Essas categorias, como seus nomes sugerem, controlam funcionalidades relativas a movimentos, aparência e som dos objetos presentes no programa. Esses objetos ou atores vão aparecer na área do palco e vão ter qualquer tipo de movimentação controlada pelos blocos azuis e suas aparências pelos roxo, con-

Figura 2.6 – Ambiente de desenvolvimento da ferramenta Snap! e suas seções. (a) Barra de ferramentas. (b) Paleta de blocos. (c) Área destinada à programação. (d) Palco. (e) Local de visualização de atores e palco



Fonte: Elaborada pelo autor

Figura 2.7 – Blocos padrões do *software* Snap! pertencentes às categorias Movimentos, Aparência e Som

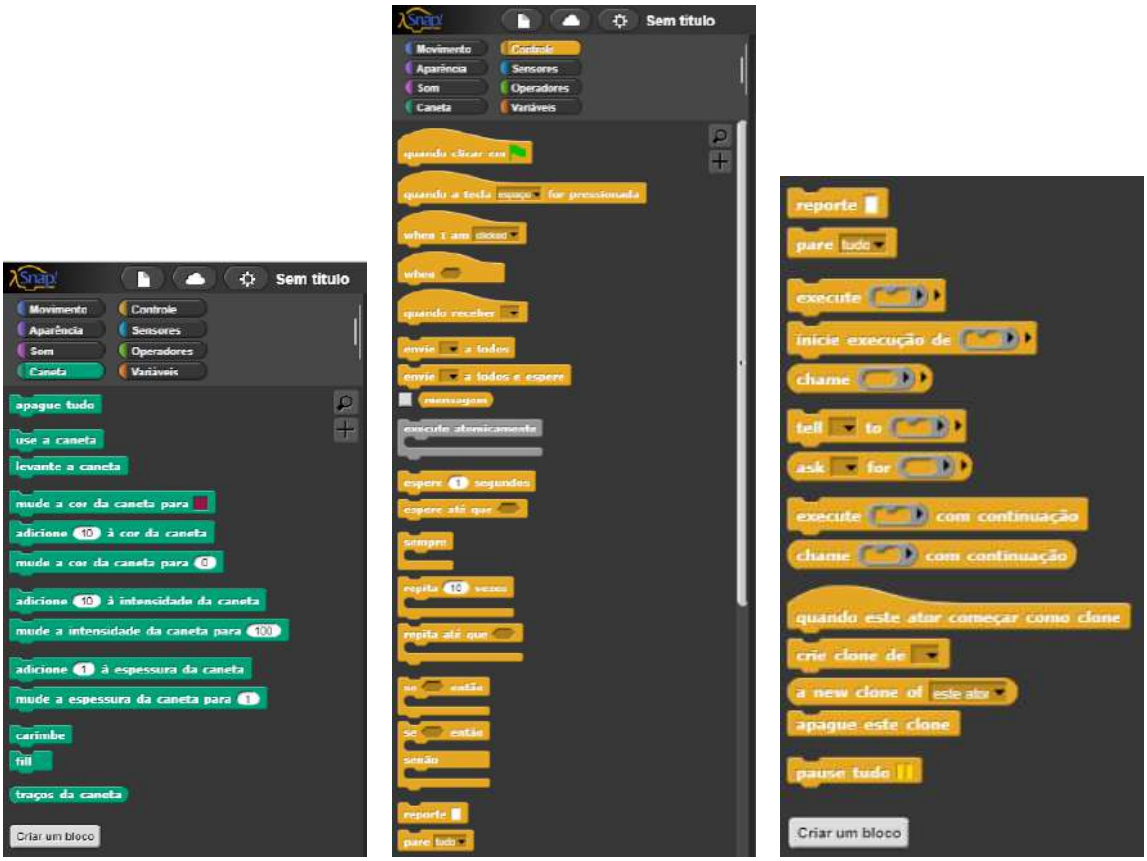


Fonte: Elaborada pelo autor

forme os blocos da figura acima. Os sons emitidos pelo programa são controlados de acordo com as funcionalidades presentes nos blocos roxo claro.

Características relativas ao controle do fluxo do código criado são possibilitadas pela categoria Controle, a qual tem seus blocos em amarelo. Essa categoria também permite a troca de mensagens entre os membros do programa de forma a transmitir mensagem a todos os membros do programa ou para algum em específico. Existe a categoria chamada Caneta, cuja cor é verde escuro, que possibilita a implementação de efeitos como o de uma caneta a serem feitos na área do palco, fazendo com que o objeto que contém estes blocos de programação se comporte conforme uma caneta, possibilitando o controle de cor, espessura e intensidade do traço feito. As funcionalidades disponíveis por padrão dessas duas categorias são apresentadas na imagem 2.8.

Figura 2.8 – Blocos padrões do *software* Snap! pertencentes às categorias Caneta e Controle

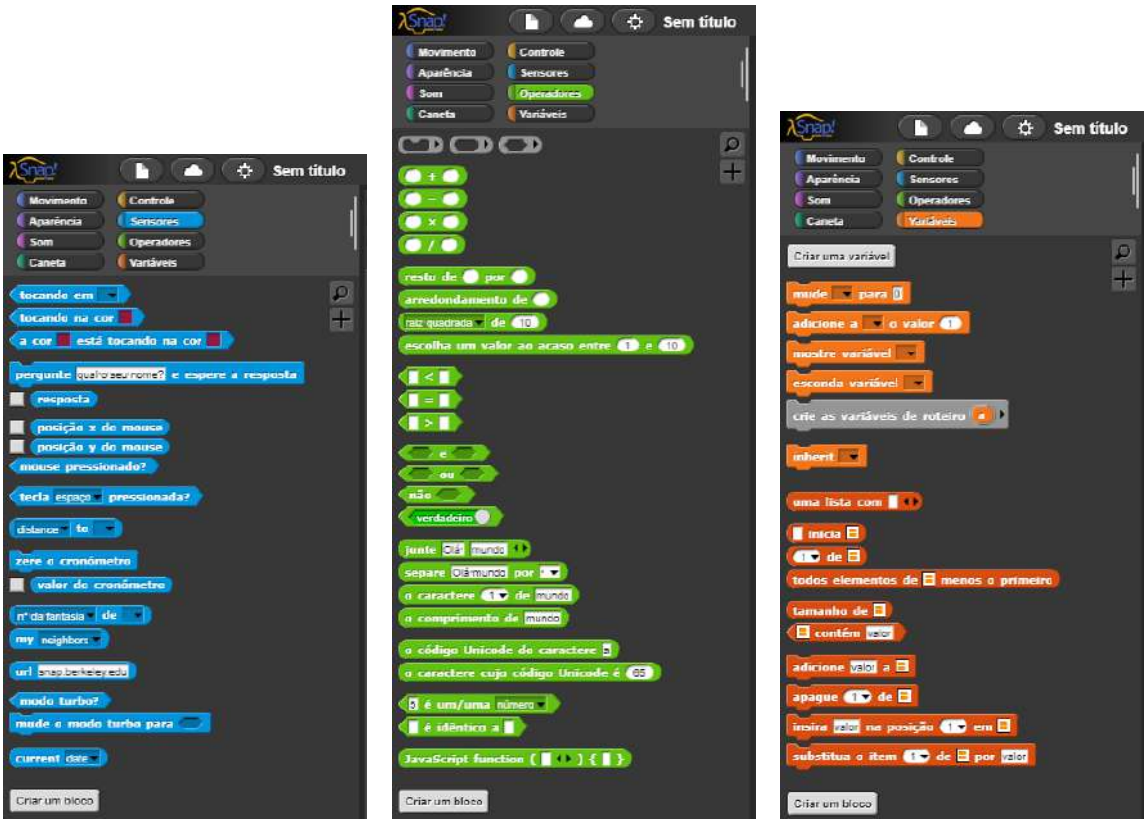


Fonte: Elaborada pelo autor

Os conteúdos de blocos das categorias de Sensores, Operadores e Variáveis são mostrados na imagem 2.9. A categoria Sensores, de cor azul clara, trabalha principalmente com base em eventos, como o clicar do mouse, o encontro entre dois atores do programa ou, ainda, o encontro de um ator com uma cor em específico. A categoria de Operadores oferece o suporte a operações matemáticas básicas como multiplicação, divisão, soma e subtração. Essa categoria ainda dá suporte aos operadores de comparação “maior que”, “menor que” e “igual a”, além de algumas outras funcionalidades. Por último, a categoria laranja, denominada Variáveis, fornece estruturas capazes de gerenciar variáveis de me-

mória, criando-as, atribuindo valores a elas e verificando os valores armazenados em cada uma delas, além também de possibilitar a criação e edição de listas.

Figura 2.9 – Blocos padrões do *software* Snap! pertencentes às categorias Sensores, Operadores e Variáveis



Fonte: Elaborada pelo autor

Os blocos diferem um pouco entre palco e atores, pois não faz sentido, por exemplo, verificar se o palco está tocando em algum ator. Porém, quando se trata da programação de um ator, dependendo do contexto do programa que está sendo criado, é coerente analisar se esse ator está em contato com algum outro. No caso, o palco não dispõe de todos os blocos que um ator dispõe, porém um ator possui todos os blocos que um palco possui.

2.2 TRABALHOS RELACIONADOS

Devido ao vasto número de propostas de ferramentas já apresentadas por pesquisadores para o ensino de programação e seus conceitos básicos para crianças e adolescentes, não há viabilidade de se falar sobre todas as soluções existentes, e nem mesmo é este o foco do trabalho. Contudo, analisando algumas das soluções propostas por pesquisadores nessa área, é possível notar semelhanças e diferenças, entre elas em relação ao tipo de linguagem de programação utilizada, a presença de um *software* personalizado

para a montagem dos algoritmos e, ainda, o uso de um material físico de apoio, como kits de robótica ou um robô propriamente dito. Essas soluções serão apresentadas aqui, discutidas e analisadas com relação à proposta de ferramenta trazida por este trabalho.

Dentre os trabalhos apresentados nesta seção, alguns vão parcialmente ao encontro da proposta deste estudo quando da utilização de um ambiente de programação visual, utilizando linguagem de blocos, como Scratch. Enquanto que outros vão de encontro, devido à falta de uma ferramenta física com a qual os alunos possam trabalhar durante as aulas de programação.

As abordagens utilizadas nas soluções que fazem uso apenas de meios virtuais como ferramenta de ensino, apresentam principalmente metodologias envolvendo programação de jogos. Porém, há a utilização de meios como sites e interfaces de desenvolvimento puras, voltadas para o público jovem. Como exemplo de site há o code.org, discutido em (DANTAS; COSTA, 2013), que é uma ferramenta online cujo objetivo é reunir e organizar soluções de ensino-aprendizagem de programação de outras plataformas em um só local, separando-as por nível e conteúdos trabalhados. Por reunir diferentes plataformas, o site contém aulas para diversas linguagens e conceitos de programação para diversos públicos, desde crianças até jovens e adultos.

Em relação a ambientes de ensino com jogos, alguns trabalhos apresentados são GrubiBots, o qual apresenta um robô carrinho virtual que é programado em linguagem de blocos, mostrado em Oliveira et al. (2014), que é voltado para o público infantil. O RoboCode que utiliza programação Java para controlar tanques robôs em um campo de batalha, visto em Amaral, Silva e Pantaleao (2015), que é voltado para alunos do Ensino Médio. O KidCoder dispõe tanto da interface de programação visual para comandar um robô avatar virtual através de um jogo estilo RPG, descrito por Galdino, Neto e Costa (2015), com aplicação destinada a alunos de disciplinas de programação do ensino superior. Ainda, há também alguns trabalhos citados por Medeiros, Silva e Aranha (2013), que apresentam uma revisão sistemática da literatura de artigos referentes à utilização de jogos digitais no ensino de programação para alunos de graus de escolaridade fundamental, médio e superior.

Existe também o emprego de ambientes de desenvolvimento que já são utilizados muitas vezes como base para outras soluções, como é o caso do próprio Scratch, que foi desenvolvido para ser utilizados por crianças, porém não é limitado e pode ser utilizado por um público de qualquer faixa etária. Há, ainda, o Blockly, outra ferramenta bem semelhante ao Scratch, desenvolvido e disponibilizado sem custos (GOOGLE, 2018). Existe o Blockly Games, que é a utilização do ambiente de programação visual Blockly aplicado a jogos, os quais possuem níveis distintos a serem superados. Já uma solução voltada para o ambiente Android é o App Inventor, que é aplicado e comparado ao Scratch na metodologia de Papadakis et al. (2016), voltada para alunos do ensino médio. Todas essas soluções, independente de público alvo, têm em comum que nenhuma delas apresenta um componente

físico como ferramenta de apoio, o que vai de encontro com a proposta deste trabalho.

Outros estudos, no entanto, corroboram com este trabalho devido à grande semelhanças nos objetivos e metodologia, os quais visam tornar a aprendizagem de conceitos de programação mais concretos através da utilização de ferramentas lúdicas relacionadas à robótica educacional, com a utilização de kits de robótica, de eletrônica e robôs prontos. Entre esses, considerando kits de montagem, foram analisados os trabalhos apresentados por Kusuma, Utaminingrum e Kakeshita (2018) e Kurebayashi, Kamada e Kanemune (2006).

Na proposta de Kusuma, Utaminingrum e Kakeshita (2018), é utilizado o kit mBot, o qual disponibiliza peças para montagem de diferentes versões de um robô carro, com três rodas, sendo duas controladas por motores e outra um roda omnidirecional passiva. Esse kit contém também um LED RGB, Buzzer, receptor e transmissor IR, sensor de luminosidade, sensor ultrasônico, sensor seguidor de linha e módulo bluetooth. A programação é realizada através de um ambiente chamado mBlock, baseado em linguagem visual de blocos de Scratch.

O trabalho de Kurebayashi, Kamada e Kanemune (2006) também apresenta a montagem de um carro robô autônomo, construído a partir de partes de um kit de chassis de veículos para crianças. O protótipo conta apenas com dois motores para as rodas e um botão como sensor. A linguagem de programação utilizada é o Dolittle, a qual foi escolhida devido a disponibilidade de escrita de códigos em japonês e de uma respectiva interface de programação para essa linguagem, isso tudo devido a ser um software utilizado no Japão. Os comandos disponíveis para controlar o robô eram mover-se para frente, para trás e rotacionar. Aulas para desenvolvimento de um jogo qualquer utilizando apenas botões foram adicionadas à metodologia. Nesse estudo, 90% dos estudantes participantes afirmaram ser divertido aprender sobre robôs, embora mais do que 70% dos alunos tenham declarado que é difícil controlar robôs através de programas. Alguns alunos com desempenho alto consideraram a programação do robô chata e, justamente por isso, os autores concluíram a necessidade da presença de um número maior de sensores nos robôs educacionais. Alunos com desempenho mais baixo consideraram a linguagem Dolittle difícil, justificando que robôs são mais divertidos e que os fizeram se interessar em computadores. Contudo, o estudo convenceu os autores do aumento da motivação causada nos alunos com a introdução de robôs na aprendizagem de programação.

Há, ainda, uma abordagem que, além de um kit de robótica, utiliza também um site, o "Racha Cuca", o qual contém desafios de lógica, que são utilizados como primeira etapa de ensino, seguida de aulas teóricas, montagem de um robô com o kit LEGO MINDSTORM Education EV3, Scratch e, por fim, a programação dos robôs de LEGO montados (MARTINS et al., 2016). É interessante citar que, segundo os autores, os alunos que participaram das atividades propostas demonstraram grande empolgação, pois "foi visível a reação de felicidade das crianças a cada vez que o robô realizava uma tarefa proposta".

No entanto, de acordo com os relatos, os alunos apresentaram certa resistência à utilização correta da IDE oferecida pela empresa LEGO para programação dos robôs.

No quesito de robôs ferramentas já prontos, há os robôs carrinhos Infantes, o robô Atti, com uma identidade própria, o N-bot, outro robô carrinho, e o robô NAO respectivamente apresentados por Saleiro et al. (2013), Park et al. (2015), Aroca (2012) e Pot et al. (2009).

Em (SALEIRO et al., 2013), o intuito dos robôs denominados Infantes, é ser uma ferramenta de baixo custo para ser utilizada com crianças do ensino fundamental como forma de uma ferramenta motivacional para qualquer assunto. Ele é composto por dois servo motores modificados para funcionarem como rodas, um módulo bluetooth, um microcontrolador e quatro sensores infra-vermelhos. Os robôs recebem três tipos de comandos: ir para frente, girar 90º no sentido horário e girar 90º no sentido anti-horário, através de uma interface de programação baseada em Scratch.

A ferramenta Atti foi utilizada no estudo apresentado, para avaliar a criatividade e satisfação de estudantes do ensino fundamental ao expressar conhecimentos de disciplinas através da programação do robô. Para tanto, seis semanas foram utilizadas no ensino de programação básica e avançada para os alunos, através da linguagem de programação visual. Embora o foco tenha sido em disciplinas de coreano, matemática e música, os autores veem a necessidade de estudos semelhantes em outras áreas do conhecimentos para análise da efetividade de robôs pedagógicos.

O robô N-bot tem como objetivo ser uma opção de baixo custo para a robótica educacional. Possui aplicações em níveis de escolaridade técnica e universitária em diversas áreas de conhecimento, como a programação de computadores. Porém, devido à presença da opção de programação por blocos, torna a ferramenta adequada inclusive para crianças. A plataforma de desenvolvimento de código é na web, podendo ser realizada tanto por um computador quanto por um dispositivo móvel, telefone ou tablet. Possui suporte a inúmeras linguagens textuais, como Python, JavaScript, linguagem visual, entre outras. Possui até mesmo suporte à programação com linguagens híbridas, com a utilização de linguagens textuais e visuais misturadas. Em especial, essa solução corrobora fortemente com o trabalho atual, pois, conforme o autor afirma, a ferramenta tem "a intenção de oferecer uma plataforma simples e intuitiva para iniciantes, porém poderosa e flexível para usuários experientes"(AROCA, 2012).

Em Pot et al. (2009), a linguagem de programação visual Choregraphe é apresentada. Ela é uma das linguagens de programação presentes no robô humanoide NAO, o qual também tem suporte às linguagens C++ e Python. O robô NAO possui 25 graus de liberdade e inúmeros sensores a atuadores que podem ser controlados através de programação. Esses componentes são descritos na Tabela 2.3. O foco dessa publicação é a plataforma de programação disponibilizada pela ferramenta. Como os próprios autores explicam, robôs são dispositivos elaborados capazes de realizar tarefas complexas e para tal

é necessário uma plataforma de programação que dê acesso a todas as funcionalidades do robô de forma eficiente.

A seguir, três tabelas são apresentadas. Cada uma delas agrega trabalhos semelhantes, os quais foram separados de acordo com o tipo de soluções propostas: soluções em forma de *softwares*, kits ou de ferramentas prontas. Cada tabela apresenta características de *Software*, *Hardware* e linguagem de programação de cada uma das soluções propostas.

Ferramenta	**Descrição**	***Hardware***	**Linguagem de Programação**
Site code.org	Cursos online na plataforma do site	Nenhum	Diversas
GrubiBots	Jogos com um robô virtual	Nenhum	Visual
RoboCode	Jogo de batalha de robôs virtuais	Nenhum	Java
KidCoder	RPG com robô avatares	Nenhum	Visual
App Inventor e Scratch	Comparação entre Scratch, App Inventor para Android e ensino de Pascal	Nenhum	Visual

Tabela 2.1 – Ferramentas de *software* apresentadas

A Tabela 2.1 apresenta a comparação de todas as ferramentas citadas nessa seção mais a descrição do robô Beo. Com base nela, pode-se perceber a diferenciação do robô Beo em relação às demais ferramentas. Entre as principais diferenças temos, primeiramente, que o robô Beo se apresenta como uma solução física e não apenas virtual, quando comparado com as soluções apenas à nível de *software*.

Ferramenta	**Descrição**	***Hardware***	**Linguagem de Programação**
Kit Mbot	Carro robô	Kit com 1 LED RGB, 1 Buzzer, 1 Sensor de luz, 1 receptor e 1 emissor IR, 1 Sensor Ultrassônico, 1 Botão, 1 Sensor de linha, 2 Motores e 1 Módulo bluetooth	Visual

Kit para robôs (sem denominação)	Carro robô	kit com motores e botão	DoLittle (linguagem escrita com suporte à japonês)
Site Racha Cuca e LEGO	Atividades de lógica, teoria, Scratch, montagem de robô de LEGO e respectiva programação	Kit LEGO com 1 Sensor Ultrasônico, 2 Sensores de toque, 1 Sensor de cor, 1 Sensor de rotação, 3 Servo-motores e uma1 Bateria	Visual

Tabela 2.2 – Soluções apresentadas em forma de kits

No que diz respeito aos trabalhos que apresentam kits como soluções, resumidos na Tabela 2.2, nota-se que esse tipo de solução pode trazer um número considerável de sensores e atuadores, o que é interessante do ponto de vista das possibilidades de programas que podem ser criados pelos alunos com a utilização em conjunto de todos esses componentes eletrônicos. No entanto, nesse caso não há como fugir da montagem do objeto a ser controlado, o que pode ocasionar em limitações em relação ao que o artefato montado é capaz de fazer ou não.

Ferramenta	**Descrição**	***Hardware***	**Linguagem de Programação**
Infantes	robôs carrinhos	4 Sensores infravermelhos, 2 motores, 1 módulo bluetooth	Visual
Atti	robô com identidade própria	1 alto falante, 1 microfone, múltiplos sensores de toque e proximidade, LEDs, sensor de objetos, motores nos braços e cintura, permitindo movimentação omnidirecional	Visual
Nbot	robô carrinho	Decodificador DTMF, 2 sensores de toque e 2 servo-motores	Python, JavaScript, Visual, entre outras

Robô NAO	robô humanoide com identidade própria	2 giroscópios, 3 acelerômetros, 4 sensores de pressão sob cada um dos pés, 1 sonar, sensores de colisão nos pés, sensor de toque, 2 câmeras, 4 microfones, 2 autofalantes, LEDs nos olhos e 25 servo-motores	Choregraphe (linguagem visual), C++ e Python

Tabela 2.3 – Soluções apresentadas em forma de ferramentas prontas

Por último, a Tabela 2.3 mostra as soluções que vão ao encontro da proposta deste trabalho: a apresentação de uma ferramenta física pronta que permita o aluno aprofundar seus conhecimentos sem ser limitado pela própria ferramenta. Dentre as soluções apresentadas, o robô NAO é a que mais se assemelha ao robô Beo visualmente, exceto pela presença de rodas no lugar das pernas no caso do Beo. Os robôs Infantes e Nbot são as soluções prontas que apresentam o menor número de sensores e atuadores que podem ser controlados pelo usuário, o que pode diminuir a gama de aplicações possíveis da ferramenta. Em contraste com o robô Beo, que oferece mais de dez sensores e atuadores que podem ser controlados pelo usuário, o que aumenta a quantia de aplicações nas quais o robô pode ser utilizado.

No que diz respeito às linguagens de programação disponibilizadas nos trabalhos, nem todas disponibilizam a linguagem visual, a qual é utilizada no robô Beo e, de acordo com Maloney et al. (2010), é mais recomendada para usuários inexperientes em programação.

3 METODOLOGIA

O tipo de metodologia de pesquisa que mais se aproxima à utilizada neste trabalho é a pesquisa descritiva. Uma das principais características desse tipo de estudo é a utilização de técnicas padronizadas para coleta de dados, como, por exemplo, a observação sistemática (GIL, 2002). No caso, uma observação sistemática direta foi utilizada durante todo o processo de produção da ferramenta, com intuito de avaliar o correto funcionamento da ferramenta e suas respectivas funcionalidades.

Do ponto de vista de sua natureza, este trabalho apresenta uma pesquisa aplicada, a qual, segundo a definição de Silva e Menezes (2005), tem como objetivo a sua aplicação prática, gerando conhecimentos dirigidos à solução de problemas específicos. No caso deste trabalho, é apresentada uma ferramenta para ser utilizada na prática em aulas de programação para crianças, de forma que essa ferramenta se apresente como uma solução física, lúdica e intuitiva para iniciantes ao mesmo tempo que proporciona possibilidade de aplicações complexas para usuários mais avançados.

O processo de desenvolvimento da ferramenta se iniciou com sua idealização, ou seja, qual seria seu visual físico e que funcionalidades possuiria. Em seguida, os sensores e atuadores necessários para atender às funcionalidades desejadas foram comprados. Posteriormente, um protótipo de teste foi montado para cada uma das funcionalidades, com seus respectivos sensores e atuadores necessários, assim como todo material de suporte necessário para montagem, como fios, Protoboard e fonte de alimentação. Por fim, cada uma das funcionalidades foi implementada no robô em forma de um módulo em *software*, descrito em Python.

Testes para verificar o funcionamento das funcionalidades do robô foram feitos dentro da empresa Qiron Robotics e também durante as aulas de robótica e programação ministradas pela mesma, em um espaço concedido por uma escola de idiomas da cidade de Santa Maria, na qual havia disponibilidade de mesas, cadeiras e projetores. Ao todo, foram montados quatro réplicas deste projeto, das quais, uma não chegou a ser finalizada. O período de prototipagem, montagem e programação dos robôs foi de aproximadamente seis meses. Durante esse período todas as funcionalidades foram desenvolvidas e replicadas para todos os robôs. Após esse período de prototipação, os robôs foram utilizados em sala de aula durante três semestres consecutivos, iniciando no primeiro semestre de 2017. As aulas ensinavam robótica e programação utilizando o robô e o *software* Snap! para o seu respectivo controle. Durante esse período, algumas atualizações e correções foram realizadas, especialmente a nível de *software*, corrigindo *bugs* e adicionando funcionalidades aos módulos programados de acordo com a necessidade vistas pelos instrutores da empresa que ministravam as aulas para as crianças. As aulas ministradas duravam 1h30min cada, sendo ao todo 16 aulas por semestre. Porém, o robô foi utilizado apenas

em quatro ou cinco das aulas de cada um dos semestres, dependendo da turma, as quais tinham de um a seis alunos. O robô foi utilizado por quatro instrutores diferentes, em mais de cinco turmas distintas.

As Tabelas 3.2 e 3.1 apresentam o status de funcionamento de *software* e *hardware* para cada uma das réplicas do robô montadas. Essas tabelas eram atualizadas diariamente durante o desenvolvimento e período de testes dos robôs em sala de aula, para que todos os envolvidos no desenvolvimento do projeto soubessem o que precisava ser arrumado e o que estava funcionando corretamente ou não.

Figura 3.1 – Tabela de controle de funcionamento de *Software* dos robôs

Beo	S.O.	Movements	Wheels	Audio	Eyes	Camera	Microfone	Touch Sensor	Ultrasonic	InfraRed	Snap
Sem núm.											
006											
aulas (A)											

Cor	Estado
	Ok - Módulo/Classe testado e funcionando
	Módulo não testado
	Módulo não está funcional
	Módulo ainda não implementado

Fonte: Elaborada pelo autor

Há apenas três linhas nessa tabela pois um dos robôs não chegou a ser montado de fato, existindo apenas três *softwares* portanto para serem monitorados. Isso é possível verificar também na Tabela 3.2, na qual é possível ver que um dos robôs apresenta todos seus módulos em azul, indicando que os módulos não haviam sido implementados no robô especificado.

Figura 3.2 – Tabelas de controle de funcionamento de *Hardware* das diferentes réplicas construídas do robô

Beo	Raspberry Pi	Servos	Rodas	Camera	Microfone	Auto Falante	Displays oLed	Touch Sensor	Ultrasonic Sensor	Infra Vermelho	Botões	Leds	Bateria	Carregador
005													Sem Bateria	LiFe config
006													Sem Bateria	LiPo config
007													Sem Bateria	LiPo config
008													Sem Bateria	LiFe config

Cor	Estado
	Ok - Módulo/Classe testado e funcionando
	Módulo não testado
	Módulo não está funcional
	Módulo ainda não implementado

Fonte: Elaborada pelo autor

Cada módulo montado e programado representa uma funcionalidade do robô, a qual funciona de acordo com o módulo de controle feito em Python. Para testar a funcionalidade de cada um dos módulos, foram criados *scripts* de teste, um para cada módulo.

Os testes eram executados um a um antes das tabelas de controle de funcionamento dos robôs serem atualizadas, com exceção da câmera do robô, que não possuía algoritmo de verificação pois imagem era testada a partir de um programa padrão do sistema operacional utilizado no robô. Os *scritps* de testes utilizados encontram-se na Sessão B.

A maior parte do projeto a qual o autor ficou responsável foi a implementação das rodas, e toda interface delas com o Rasperry Pi feita através do Arduino. Isso incluiu toda programação em C++, a qual abrangia o sensor de toque e controle PID das rodas, além das respectivas implementações em *Hardware* necessárias para o funcionamento de todos esses componentes eletrônicos. Além dessa atribuição, o autor desenvolveu por completo também, as interfaces de *hardware* e *software* para os módulos responsáveis denominados "PushButtons", "Leds"e "Movements", os quais eram responsáveis, respectivamente, pelos botões, LEDs e controle dos servo-motores utilizados nos braços e cabeça do robô. Ainda, o autor ajudou no desenvolvimento dos blocos de controle do robô, em linguagem visual, além de realizar teste de funcionalidade de todos os módulos constantemente ao longo do período de um ano e meio que esteve envolvido diretamente com o projeto do robô Beo.

Basicamente, os módulos foram feitos ou refeitos, testados e corrigidos/aprimorados, repetidamente, ou seja, esse ciclo se mantém durante todo o período do projeto até hoje. Em seguida, cada um dos módulos do robô Beo será apresentado. Cada um deles contém uma breve descrição do seu funcionamento, apresentação dos componentes eletrônicos utilizados em sua implementação e os métodos desenvolvidos em *software* para controle da funcionalidade em questão.

3.1 ROBÔ BEO

O robô humanoide Beo foi um projeto desenvolvido pela empresa Qiron Robotics, com participação do autor deste trabalho, e utilizado como tema deste TCC. O Beo foi criado com o objetivo de servir como ferramenta de ensino para robótica e programação. Todo o projeto, tanto de *hardware* quanto de *software*, foi totalmente idealizado em coautoria com demais membros da empresa, buscando proporcionar diversas funcionalidades intuitivas para os usuários da ferramenta. Assim como todo robô humanoide, o robô Beo possui atuadores, sensores e unidades de controle, incluindo todo *software* de suporte necessário para o funcionamento de seus componentes e para trazer intuitividade no controle do mesmo.

Em seu *hardware*, há grande quantidade de sensores e atuadores para possibilitar a interação do robô com o ambiente ao seu redor. Para controlar todos esses componentes, o robô conta com duas unidades físicas de controle: Raspberry Pi 3 modelo B e um Arduino Pro Mini. O Raspberry Pi é utilizado como principal unidade de controle, sendo responsável pelo controle de todos os dispositivos presentes no robô, inclusive da placa Arduino. Para

Figura 3.3 – Robô humanoide Beo

Fonte: (ROBOTICS, 2018a)

realizar a interface física entre o Raspberry Pi e demais componentes, foi projetada uma placa de circuito impresso específica para conter todo *hardware* e conectores necessários para essa tarefa. Sua estrutura física, corpo e membros, foi desenvolvida em *software* CAD e toda impressa em impressoras 3D, utilizando filamentos de PLA.

O sistema operacional sobre o qual toda programação do robô foi feita é o Raspbian, *software* livre baseado em Linux e otimizado para o *hardware* da placa Raspberry Pi. Com base nessa plataforma, o código de controle do robô e suas funcionalidades foram organizados em estrutura de módulos, sendo cada módulo responsável por alguma funcionalidade ou controle de algum componente físico de atuação ou sensoriamento do robô. Atualmente, o sistema conta com 17 módulos dos quais 11 são destinados ao controle de algum componente físico, os quais terão enfoque neste trabalho, enquanto que os demais têm como função o suporte dos demais módulos ou alguma funcionalidade extra do humanoide.

A programação do *hardware* presente no robô foi feita utilizando as linguagens de programação Python e C++. A linguagem Python foi utilizada na descrição dos módulos do robô, os quais são utilizados para controlar cada um dos sensores e atuadores existentes nele, assim como para controlar a comunicação entre Raspberry Pi e Arduino. Os módulos disponibilizados até então pelo projeto, os quais representam as interfaces de controle dos sensores e atuadores dos robôs em *software* são os seguintes: Audio, Camera, Eyes, IrSensor, LEDs, Microphone, Movements, PushButtons, SensorTouch, Ultrasonic e Wheels.

A linguagem C++ foi utilizada no Arduino com o objetivo de codificar nos motores das rodas, o protocolo e forma de comunicação presente nos servo-motores dos braços e cabeça do robô. Ainda há presença da linguagem de programação visual, que é disponibilizada como alternativa de controle dos sensores e atuadores do robô em relação aos

módulos descritos em Python.

Em geral, assim como os seres humanos, os robôs possuem meios de sentir o ambiente que os rodeia e agir sobre ele. Tanto a ação no mundo, através da fala ou movimento, quanto as informações recebidas dele, como sons e imagens, são processadas por uma unidade de controle no robô, a qual é constituída por um computador e é equivalente ao cérebro humano. Os componentes eletrônicos responsáveis por fazer os robôs sentirem o ambiente são denominados sensores, enquanto que os componentes responsáveis por permitir os robôs agirem no ambiente de alguma maneira são chamados de atuadores. Como comentado anteriormente, é necessário uma forma de controlar tanto atuadores quanto sensores, assim como toda informação recebida e enviada para ambos, tarefa que é exercida pelas unidades de controle colocadas no robô. Todos os componentes responsáveis pelo sensoriamento, atuação e processamento de informações presentes no robô são apresentados pelas sessões a seguir. Em especial, os motores do robô, tanto das rodas quanto do corpo, funcionam tanto como sensores quanto como atuadores, por isso, serão apresentados na Sessão 3.1.3.

3.1.1 Sensores

3.1.1.1 Câmera e microfone

As interfaces de captura de áudio e vídeo são feitos através da câmera modelo C270 produzida pela empresa Logitech a qual, além da lente, conta com um microfone embutido. Ela vem de fábrica conforme a imagem apresentada a seguir, porém, para instalação na cabeça do robô, a estrutura que recobre o *hardware* é retirada e, então é encaixada e parafusada internamente no local reservado a ela na parte relativa à testa do robô.

Figura 3.4 – Câmera Logitech C270 como é vendida e após ter sua estrutura de proteção aberta

Fonte: Autor desconhecido

A câmera é conectada ao Raspberry Pi através da entrada USB tendo, portanto, todo o suporte à transmissão de imagem ou áudio ao *hardware* central do robô já implementado, robusto e de fácil acesso.

A interface com o dispositivo de imagem foi feita através do módulo “Camera". Esse módulo em específico possui apenas um método que tem como aplicação a identificação de padrões de QR Code na imagem capturada. A imagem transmitida pela câmera permite ainda a aplicação de qualquer *software* de processamento de imagem e possibilita inúmeras outras aplicações além dessa codificada nesse módulo.

Nome do método	Descrição
read_qr_code(show=False)	De acordo com o parâmetro “show” define se a janela mostrando a imagem capturada pela câmera será ou não aberta. No caso de encontrar um QR Code na imagem capturada pela câmera, o método retorna o dado contido no QR Code identificado em forma de String

Tabela 3.1 – Métodos do módulo “Camera” e suas respectivas descrições

A detecção de sons possui apenas um método, o qual é apresentado na Tabela **??**, e é feita através do módulo “Microphone”.

Nome do método	Descrição
read_audio(duration, filename)	Método para captura de áudio, cuja duração é de “duration” e o nome do arquivo no qual o áudio captado é salvo é o parâmetro “filename”

Tabela 3.2 – Métodos do módulo “Microphone” e suas respectivas descrições

3.1.1.2 Botões

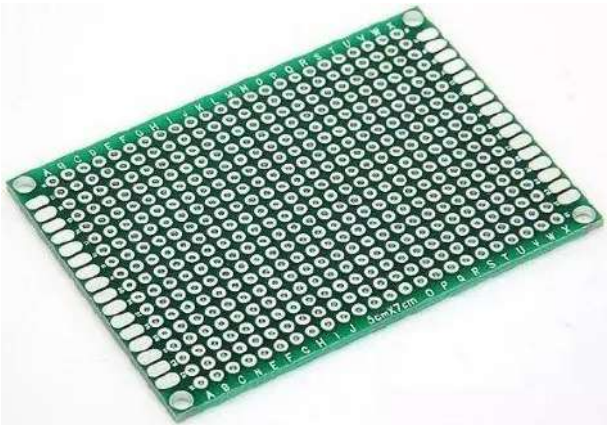
Juntamente à mochila do robô há três botões, utilizados conforme a necessidade da aplicação sendo desenvolvida no momento. O módulo Python que os controla é chamado de “PushButtons”. A seguir, seus métodos são apresentados.

Uma placa de circuito impresso ilhada foi recortada e utilizada para abrigar os botões e LEDs, a qual por sua vez foi fixada à lateral da mochila do robô no lado correspondente às aberturas para abrigar esses componentes.

Figura 3.5 – Mochila do robô com os botões

Fonte: Elaborada pelo autor

Figura 3.6 – Placa de circuito impresso ilhada



Fonte: Autor desconhecido

Nome do método	Descrição
listen_buttons()	Verifica se houve algum evento em um dos botões e retorna uma *String* identificando o botão caso algum deles tenha sido pressionado

Tabela 3.3 – Método do módulo “PushButtons” e sua respectiva descrição

3.1.1.3 Sensor capacitivo de toque e sensor ultrassônico

Dois sensores que possibilitam ao robô sentir o ambiente ao seu redor são os sensor capacitivo de toque e o sensor ultrasônico, permitindo-o identificar toques em sua cabeça e a que distância há objetos em sua direção, respectivamente. As interfaces em Python que descrevem o comportamento desses sensores são “SensorTouch” e “Ultrasonic” e suas funcionalidades são apresentadas em duas tabelas logo abaixo.

Figura 3.7 – Sensor capacitivo de toque presente na parte superior da cabeça do robô

Fonte: Elaborada pelo autor

Nome do método	Descrição
get_status()	Verifica o status atual do sensor de toque. Caso esteja sendo tocado, retorna o valor 1, caso contrário o valor 0

Tabela 3.4 – Métodos do módulo “SensorTouch” e suas respectivas descrições

Figura 3.8 – *Hardware* do sensor ultrassônico utilizado

Fonte: Autor desconhecido

Nome do método	Descrição
get_distance()	Retorna a distância ao objeto mais próximo identificada pelo sensor

Tabela 3.5 – Métodos do módulo “SensorTouch” e suas respectivas descrições

3.1.2 Atuadores

3.1.2.1 Áudio

A emissão de som é realizada através de um alto falante de 8 ohms com potência de 30 Watts, localizado no peito do robô, ligado a uma placa amplificadora de áudio digital Bluetooth de 50 Watts, modelo TDA7492P, produzida pela empresa Sanwu Electronics, através da qual qualquer som gerado pelo robô pode ser reproduzido em diferentes

dispositivos que tenham suporte à tecnologia Bluetooth.

Figura 3.9 – Alto falante e *hardware* amplificador de áudio com suporte Bluetooth utilizados para reprodução de áudio

Fonte: Elaborada pelo autor

O módulo em *software* responsável por todo processamento relacionado a sons é denominado “Audio”, ele gerencia tanto a criação quanto a execução de qualquer arquivo de som relacionado à “voz” do robô. Os sons reproduzidos podem ter origem de arquivos de áudio presentes no próprio sistema, ou de arquivos de áudio gerados pelo módulo através de um mecanismo online de conversão de arquivo de texto em arquivo áudio. Tal ferramenta é denominada *Text-To-Speech*, foi criada pela Google e é disponibilizada na nuvem pela mesma. O funcionamento de transcrição de texto em som ocorre por meio de algoritmos de inteligência artificial.

Em função da unidade de controle principal (descrita posteriormente) ter seus pinos relativos à interface Bluetooth compartilhados com os pinos de comunicação serial, um mini adaptador USB bluetooth 4.0 foi utilizado para possibilitar a comunicação entre o *hardware* principal do robô com qualquer dispositivo de áudio bluetooth.

A Tabela 3.6 apresenta os nomes dos métodos até então implementados pelo módulo e uma breve explicação da finalidade dos mesmos.

Nome do método	Descrição
adjust_mp3(source, destination)	Modifica os parâmetros de “echo”, “pitch” e “speed” de um arquivo de áudio “.mp3” e armazena o novo arquivo gerado sob o nome passado como parâmetro ao método. Os parâmetros de “echo”, “pitch” e “speed” são pré-definidos para produzirem, juntos, o timbre característico da voz do robô para a língua inglesa

ajusta_mp3(origem, destino)	Modifica os parâmetros de “echo”, “pitch” e “speed” de um arquivo de áudio “.mp3” e armazena o novo arquivo gerado sob o nome passado como parâmetro ao método. Os parâmetros de “echo”, “pitch” e “speed” são pré-definidos para produzirem, juntos, o timbre característico da voz do robô para a língua portuguesa
fala(frase, lang=’pt’)	A partir da frase passada como parâmetro, o método gera e posteriormente executa um arquivo de áudio temporário, editado de acordo com os parâmetros de timbre característicos do robô. Como padrão, o idioma de reprodução do áudio é o Português
generate_mp3(file, text)	A partir de um texto passado como parâmetro, o método gera um arquivo de áudio no formato “.mp3”, em inglês, e o armazena com o nome passado como parâmetro
gera_mp3(arquivo, texto)	A partir de um texto passado como parâmetro, o método gera um arquivo de áudio no formato “.mp3”, em português, e o armazena com o nome que é passado como parâmetro também
gera_mp3_de(datei, text)	A partir de um texto passado como parâmetro, o método gera um arquivo de áudio no formato “.mp3”, em alemão, e o armazena com o nome passado como parâmetro
gera_mp3_es(archivo, texto)	A partir de um texto passado como parâmetro, o método gera um arquivo de áudio no formato “.mp3”, em espanhol, e o armazena com o nome passado como parâmetro
gera_mp3_fr(fichier, texte)	A partir de um texto passado como parâmetro, o método gera um arquivo de áudio no formato “.mp3”, em francês, e o armazena com o nome passado como parâmetro
gera_mp3_ja(arquivo, texto)	A partir de um texto passado como parâmetro, o método gera um arquivo de áudio no formato “.mp3”, em japonês, e o armazena com o nome passado como parâmetro
play_mp3(file, volume$\bar{1}$)	Reproduz um arquivo no formato “.mp3”

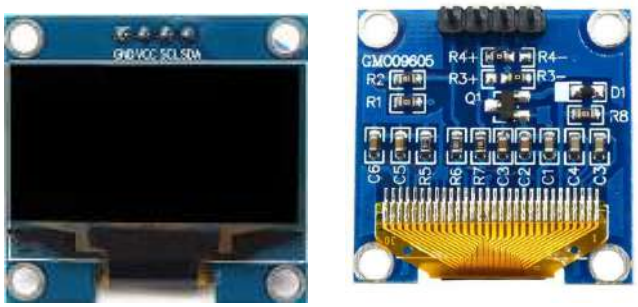
say_number(number)	Reproduz os arquivos de áudio ".mp3" presentes no sistema equivalentes aos números passados como parâmetro
speak(sentence)	A partir da sentença passada como parâmetro, o método gera e posteriormente executa um arquivo de áudio temporário, em inglês, editado de acordo com os parâmetros de timbre característicos do robô

Tabela 3.6 – Métodos do módulo "Audio" e suas respectivas descrições

3.1.2.2 Olhos

Com o intuito de demonstrar emoções, dois *displays* OLED, de 0,96 polegadas cada, foram utilizados para representar os olhos do robô. Tais interfaces podem expressar sentimentos como felicidade e tristeza, entre outros, além de permitir a implementação de novas expressões desejadas pelo usuário.

Figura 3.10 – *Display* OLED 0,96 polegadas

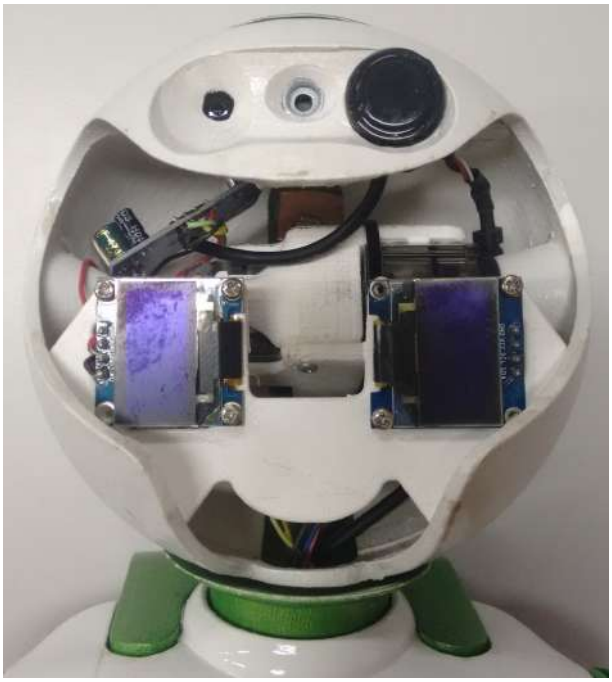


Fonte: Autor desconhecido

As expressões dos olhos são geradas a partir de arquivos com formato ".ppm". As expressões atualmente disponíveis são "default", "wink", "shut", "sad" e "mad", os quais representam, respectivamente, olhos abertos, piscando, fechados, tristes e por último maus.

O módulo "Eyes" é o responsável pelo controle desse componente eletrônico, assim como pela abstração do gerenciamento da comunicação entre Raspberry Pi e *displays*, a qual é feita através do protocolo de comunicação I2C.

Figura 3.11 – *Displays* fixados no robô



Fonte: Elaborada pelo autor

Nome do método	Descrição
disp_clear()	Remove as expressões de ambos *displays*
display_image(image, disp)	Exibe uma determinada expressão, no formato ".ppm", no display especificado
set_expression(expression)	Reproduz uma expressão visual utilizando ambos *displays*

Tabela 3.7 – Métodos do módulo "Eyes" e suas respectivas descrições

3.1.2.3 Emissor Infravermelho

Foram adicionados ao projeto dois LEDs infravermelhos na extremidade da mão esquerda do robô, com objetivo de permitir ao robô interagir com dispositivos controlados por infravermelho como é o caso de ar-condicionados e televisões. No caso, o robô possui emissores IR, o que permite o controle por parte do robô de dispositivos com suporte a esse tipo de comunicação sem fio.

O módulo de controle desse dispositivo por sua vez é denominado "IrSensor" e tem como métodos de controle os apresentados na Tabela 3.8. Esse módulo é baseado em um pacote denominado LIRC[1], o qual já implementa funcionalidades como decodificar e enviar sinais infravermelhos. Esse pacote também disponibiliza uma base de dados contendo o padrão de codificação do sinal de controle de diversos dispositivos comumente encontrados no mercado de eletrodoméstico, todos controlados por dispositivos infraver-

[1] Linux Infrared Remote Control (LIRC), é uma biblioteca aberta descrita em linguagem Python para manipulação de dispositivos infravermelhos.

Figura 3.12 – LEDs infravermelhos presentes na extremidade do braço esquerdo do robô

Fonte: Elaborada pelo autor

melhos. Portanto, o funcionamento dessa aplicação tem por trás arquivos de configuração e descrição de diversos modelos de controles remoto.

Nome do método	Descrição
send_once(remote_control_name, key)	Envia um comando através do emissor infravermelho equivalente à “key” passada como parâmetro relativa ao controle remoto selecionado por parâmetro também
list_buttons(remote_control_name)	Exibe uma listagem de todos os comandos conhecidos para um determinado controle remoto já cadastrado com seu respectivo arquivo de configuração
list_controls()	Lista todos os controles remotos já cadastrados no sistema
start()	Inicialização do serviço que controla as operações do infravermelho
stop()	Pausa do serviço que controla as operações do infravermelho
restart()	Reinicia o serviço que controla as operações do infravermelho
status()	Apresenta o status do serviço que controla as operações do infravermelho

Tabela 3.8 – Métodos do módulo “IrSensor” e suas respectivas explicações

3.1.2.4 LEDs

Juntamente à mochila do robô há dois LEDs azuis de 5mm cada, utilizados para propósitos gerais e conforme a necessidade de sinais visuais por parte dos usuários. O módulo Python que controla esse componentes é denominado “Leds” e seus métodos são apresentados na Tabela 3.9. Assim como os botões, os LEDs foram soldados junto à placa de circuito impresso apresentada na Fig. 3.6

Figura 3.13 – Mochila do robô com os LEDs

Fonte: Elaborada pelo autor

Nome do método	Descrição
on(id)	Aciona o LED cujo identificador foi passado como parâmetro
off(id)	Desativa o LED cujo identificador foi passado como parâmetro

Tabela 3.9 – Métodos do módulo “Leds” e suas respectivas descrições

3.1.3 Mistos

3.1.3.1 Motores dos braços e cabeça

Os motores responsáveis pelos movimentos dos braços e cabeça são servo-motores XL320, desenvolvidos pela empresa sul coreana Robotis. Ao todo são nove servo-motores, cada um deles é compostos por um motor DC, engrenagem de redução e um microcontrolador. Possuem controle de posição utilizando controle PID e LED de três cores para alertas visuais. São controlados por protocolo de comunicação serial assíncrono *half duplex* e possuem memória interna na qual armazenam dados do motor como posição, temperatura, tensão de alimentação, modo de operação, entre outros.

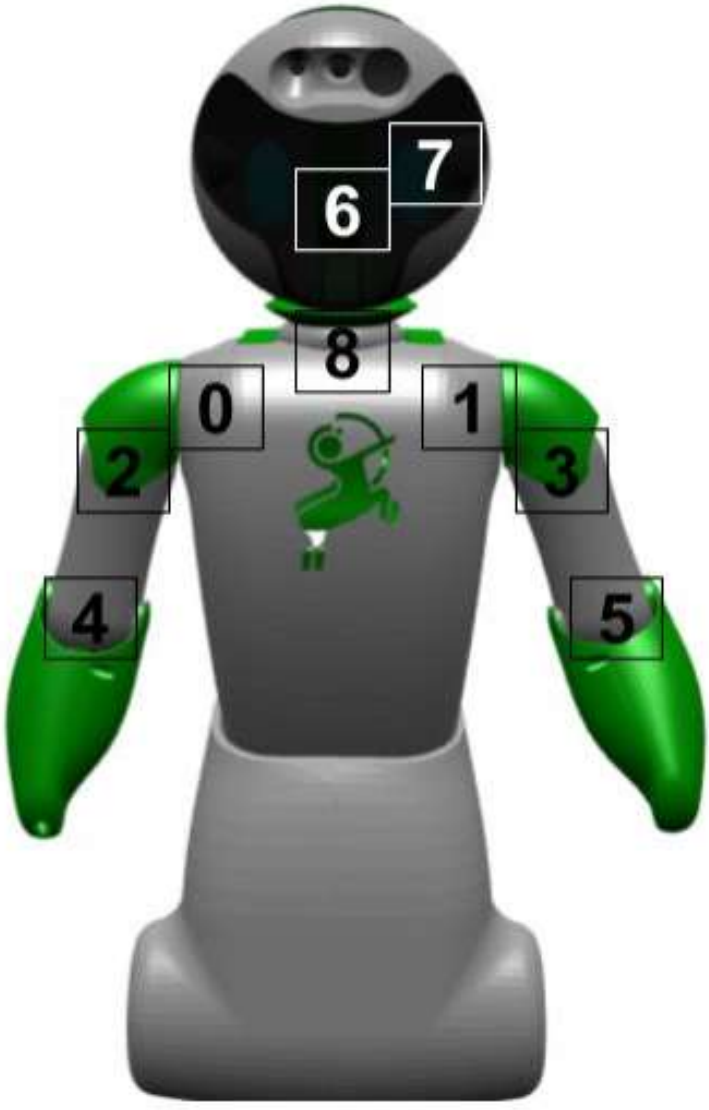
Cada motor possui um endereço de memória no qual fica armazenado seu identificador, o qual pode ser alterado modificando o conteúdo deste endereço. Essa característica existe pois permite que os motores sejam ligados em um mesmo barramento de dados e que comandos sejam enviados especificamente para um motor. A imagem a seguir mostra como são identificados os motores do robô e suas respectivas posições ao longo da estrutura física do mesmo.

“Movements” é o módulo responsável pelo controle desses motores e, também, por

Figura 3.14 – Servo-motor dynamixel XL320

Fonte: (Robotis INC, 2018)

Figura 3.15 – Posição e identificador de cada um dos servomotores no robô



Fonte: Elaborada pelo autor

toda abstração do protocolo de comunicação Dynamixel 2.0 e seu modo de comunicação com o *hardware* de processamento central do humanoide. Em seguida, a Tabela 3.10 apresenta as funcionalidades desse módulo.

Nome do método	Descrição
set_joint(index, value, respect_limits=True)	Move determinado motor para determinado POSIÇÃO, limitada ou não
set_angle(index, angle, respect_limits=True)	Move determinado motor para determinado ÂNGULO, limitado ou não
set_raw_angle(index, angle, respect_limits=True)	Move determinado motor para determinado ÂNGULO, limitado ou não. Diferentemente de "set_angle", esse método considera os ângulos dos motores equivalentes e não invertidos. Ou seja, o ângulo 0, por exemplo, representa a mesma posição para ambos braços e não complementares uma a outra
get_joint(index)	Retorna o valor da POSIÇÃO atual do motor passado como parâmetro
get_angle(index)	Retorna o valor do ÂNGULO atual do motor passado como parâmetro
get_raw_angle(index)	Retorna o valor do ÂNGULO atual do motor passado como parâmetro, considerando que os motores possuem posições angulares equivalentes
blink_led(index)	Faz com que o LED do servo, cujo índice foi passado como parâmetro pisque
enable_joint(index)	Habilita o torque do motor especificado
disable_joint(index)	Desabilita o torque do motor especificado
enable_all_joints()	Habilita o torque de todos os motores dos braços e cabeça do robô
disable_all_joints()	Desabilita o torque de todos os motores dos braços e cabeça do robô
set_pose(pose)	Determina a posição atual dos motores dos braços e da cabeça de acordo com um dicionário das posições dos motores
get_pose()	Retorna um dicionário contendo o valor da posição atual dos motores dos braços e da cabeça do robô
play_motion(motion)	Reproduz um movimento gravado usando o motionEditor
play_motion_file(filename)	Reproduz um movimento gravado em um arquivo

reboot_servo(index)	Reinicia o servo especificado nos parâmetros

Tabela 3.10 – Métodos do módulo “Movements” e suas respectivas descrições

3.1.3.2 Rodas

Os atuadores utilizados no controle das rodas são dois motores DC 6V de 100rpm, os quais contém engrenagem de redução e encoder de quadratura embutido, o qual permite a implementação de controle de distância percorrida.

Figura 3.16 – Motor DC com encoder embutido utilizado

Fonte: Fonte desconhecida

O módulo Python cuja tarefa é implementar o controle desses atuadores é denominado “Wheels”, seus métodos são descritos na Tabela 3.11.

3.1.4 Unidades de Controle

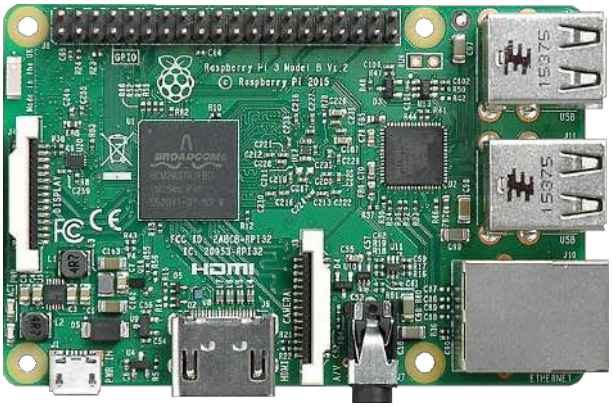
Como unidade central de controle o robô utiliza o computador Raspberry Pi B, mostrado na Fig. 3.17. As especificações desse *hardware* contam com 1GB de memória RAM, processador quad-core ARM Cortex-A7 com frequência de operação de 900MHz, quatro portas USB, espaço para cartão de memória uSD, tomada de áudio de 3,5mm, 40 pinos de propósito geral, além das interfaces físicas para Ethernet, HDMI, CSI e DSI. O sistema operacional utilizado, Raspbian, baseado em Linux, foi desenvolvido especificamente para essa plataforma de *hardware*. Essa unidade é responsável pelo gerenciamento de todos os dispositivos atuadores e de sensoriamento presentes no robô, exceto das rodas e do

Nome do método	Descrição
get_status()	Imprime no terminal informações sobre as rodas: “ID”, “goal_speed”, “avg_speed”, “pwm” e “present_position”
move(new_speed)	Move ambas rodas na velocidade passada como parâmetro
move_forward(new_speed)	Move ambas rodas para frente na velocidade passada como parâmetro
move_reverse(new_speed)	Move ambas rodas para trás na velocidade passada como parâmetro
move_stop()	Para o movimento de ambas rodas
move_wheel_left(speed)	Move apenas a roda esquerda, em ambas direções, de acordo com a velocidade passada como parâmetro
move_wheel_right(speed)	Move apenas a roda direita, em ambas direções, de acordo com a velocidade passada como parâmetro

Tabela 3.11 – Métodos do módulo “Wheels” e suas respectivas descrições

sensor capacitivo de toque, os quais estão sob administração de uma segunda unidade de controle, o Arduino Pro Mini.

Figura 3.17 – Raspberry Pi modelo B

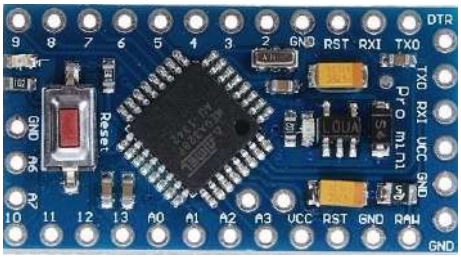


Fonte: Fonte desconhecida

Como unidade de controle secundária, o Arduino Pro Mini está constantemente verificando o estado do sensor de toque, se este foi ou está sendo tocado ou não. Além disso, a unidade é responsável pelo controle dos motores das rodas, o qual é feito através de um controle PID, o qual utiliza ganhos apenas porporcional e integral, sem uso do ganho derivativo. Ainda, o *hardware* é responsável também pela leitura dos encoders de quadratura presentes nos motores das rodas, os quais possibilitam a implementação do controle PI destes atuadores, assim como o cálculo da velocidade estimada da roda.

Além das funções citadas do Arduino, ele foi programado para desempenhar o mesmo papel que o microcontrolador presente nos motores XL320 dos braços e cabeça

Figura 3.18 – Arduino Pro Mini

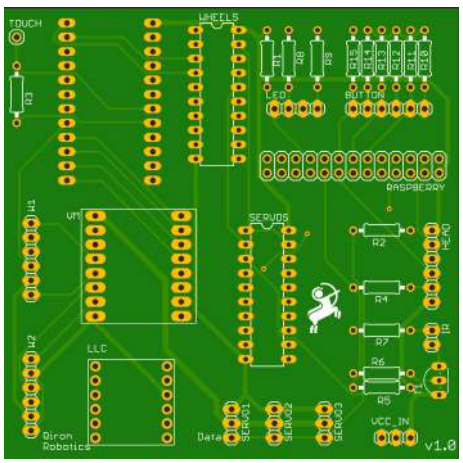


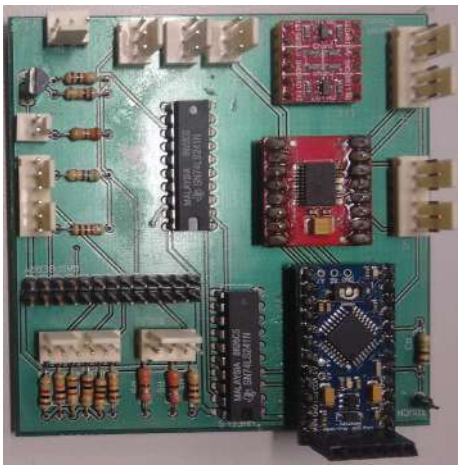
Fonte: Autor desconhecido

do robô. Ou seja, ele recebe comandos no mesmo protocolo que os motores Dynamixel, o protocolo Dynamixel 2.0. Isto foi feito justamente com intuito de que esse dispositivo pudesse ser ligado no mesmo barramento de dados utilizado para comunicação com os demais motores do robô e para que todos motores do robô passassem a ser controlados através de um único protocolo.

Para que as duas unidades de controle pudessem se comunicar entre si, utilizando o protocolo Dynamixel 2.0, foi desenvolvida uma placa de circuito impresso, a qual abriga todos os componentes necessários para realizar essa tarefa, além disso os demais componentes e conectores necessários para a utilização dos outros sensores e atuadores presentes no robô. Essa placa deixou o *hardware* organizado e fez com que diminuísse a quantidade de fios existentes no projeto.

Figura 3.19 – Placa de circuito impresso desenvolvida pela empresa em suas versões de projeto e pós fabricação já com os componentes soldados na mesma





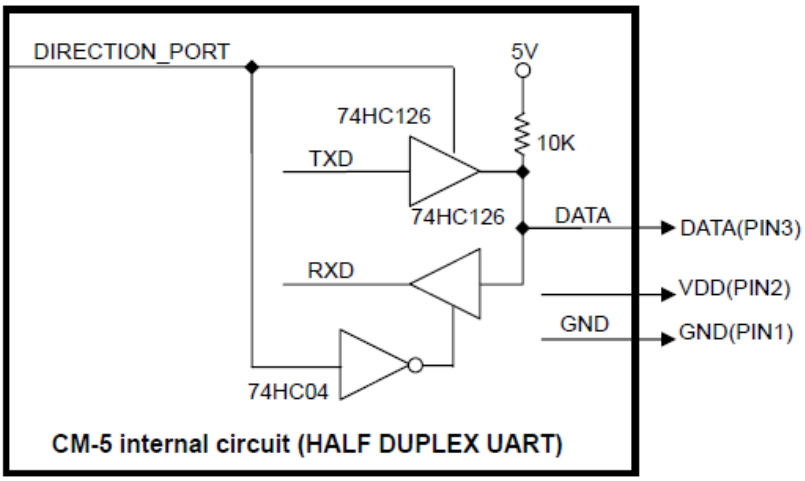
Fonte: Elaborada pelo autor

Um dos desafios de implementar a interface de comunicação entre as unidades de controle e os motores, como implementar em *hardware* a conversão de um padrão UART[2], que utiliza dois canais físicos para transmissão e recepção de dados para um tipo half-duplex, o qual utiliza apenas um canal físico para envio e recepção de informações.

[2](Universal Asynchronous Receiver/Transmitter (UART), é um padrão de comunicação serial que permite configurar características como taxa de comunicação e formato dos pacotes de dados transmitidos.

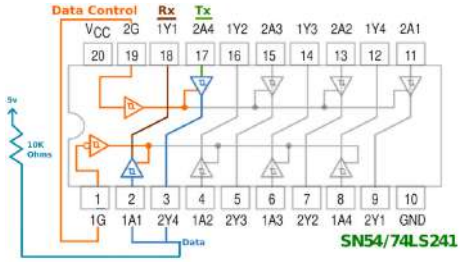
Como a própria empresa dos motores Dynamixel sugere, é necessário um circuito que utilize buffers para controlar a direção da transmissão de dados nessa transição de um tipo de comunicação para outra. Para solucionar isso, foram utilizados dois circuitos integrados SN74LS241N, os quais encapsulam buffers tri-state, conforme é possível verificar na Fig. 3.21. Essa interface física foi aplicada para ambas unidades de controle. Além da parte desenvolvida em *hardware*, houve também algumas adaptações feitas à mão nas bibliotecas Python relacionadas à temporização e sincronização das mensagens entre os dispositivos físicos, com a ajuda de um analisador de nível lógico USB.

Figura 3.20 – Esquemático da conversão Half-Duplex para UART



Fonte: (Robotis INC, 2018)

Figura 3.21 – Encapsulamento do circuito SN74LS241



Fonte: (SAVAGE, 2011)

3.1.5 Ferramentas disponíveis

3.1.5.1 Editor de movimento

Denominado MotionEditor, presente como ferramenta dentro do diretório chamado “Tools”, do inglês “ferramentas”, é um interface em *software* cujo objetivo é facilitar a criação de movimentos no robô e a reprodução desses pelo mesmo. Foi desenvolvido utilizando Pygame, uma biblioteca livre e de código aberto escrita em Python desenvolvida com intuito de servir como suporte para a produção de aplicações multimídia como jogos, por exemplo. Ela oferece interfaces simples de acesso a *hardwares* de áudio, teclado, mouse, entre outros. Além dela, Pickle foi utilizado, o qual é também uma biblioteca Python cuja funcionalidade é serializar e desserializar estruturas de objetos através da transformação desses objetos em uma série de bytes e vice-versa.

A ferramenta se baseia na configuração de poses, as quais são representadas pelo conjunto de valores de posição de cada um dos nove motores dos braços e cabeça do robô. A geração do movimento funciona através da transição de uma pose para outra, as quais são interpoladas entre si resultando no movimento desejado. A ferramenta possibilita ainda definir o tempo de interpolação entre cada pose, ou seja, o tempo que os motores demoram para sair da posição de uma pose para a seguinte. O editor permite a configuração de 19 poses por arquivo.

3.1.5.2 Software Snap!: Ambiente de programação para crianças

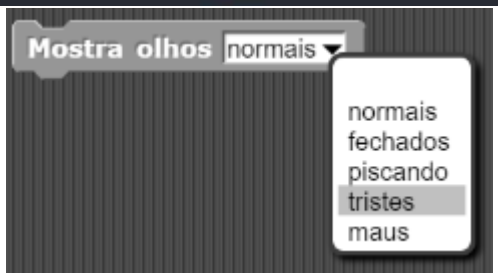
Blocos personalizados foram criados e adicionados à ferramenta Snap! para que fosse possível controlar o robô utilizando uma linguagem de blocos como a utilizada no programa. Isso é possível através da descrição em um arquivo de cada um dos blocos e de suas respectivas funcionalidades em relação ao robô. Esse arquivo é descrito em Python e funciona como um servidor, o qual fica sendo executado por trás do programa, fazendo a comunicação entre os blocos do programa Snap!, criados para controlar o robô, com os próprios módulos Python que controlam o robô. A imagem 3.22 apresenta uma função descrita dentro do arquivo "Server.py", o qual é o arquivo correspondente ao servidor responsável pela interface entre os blocos do Beo no Snap! com os módulos em Pyhton que o controlam.

As funcionalidades atualmente transcritas para blocos são as apresentadas na imagem abaixo, sendo cada uma delas relacionada a algum módulo Python presente no código-fonte do robô humanoide.

Esses blocos possuem seus formatos conforme o padrão já mencionado dos blocos

Figura 3.22 – Descrição da função responsável pela implementação do bloco que controla os *displays* OLED do robô e seu respectivo resultado em forma de bloco

```
@command("Mostra olhos %m.olhos", defaults=["normais"], is_blocking=True)
def set_expression(self, olhos):
    if olhos == "normais":
        self.eyes.set_expression("default")
    elif olhos == "fechados":
        self.eyes.set_expression("shut")
    elif olhos == "piscando":
        self.eyes.set_expression("wink")
    elif olhos == "tristes":
        self.eyes.set_expression("sad")
    elif olhos == "maus":
        self.eyes.set_expression("mad")
```



Fonte: Elaborada pelo autor

pertencentes à IDE. Sua coloração é cinza e em função de não serem nativos à ferramenta e a todo novo projeto iniciado, eles precisam ser importados para dentro da ferramenta, para então estarem disponíveis para serem usados nela.

Figura 3.23 – Blocos de controle desenvolvidos para o robô Beo

Fonte: Elaborada pelo autor

4 RESULTADOS

4.0.1 Componentes controlados pelo Raspberry Pi

4.0.1.1 Áudio

Em alguns dos robôs produzidos ocorreram problemas não identificados com a placa de áudio digital bluetooth utilizada. Após aproximadamente cinco meses de utilização com o áudio sendo gerado pelo módulo "Audio", houveram casos em dois robôs em que a placa do módulo bluetooth ficava se desligando ou reiniciando constantemente, tornando inviável a reprodução de áudio devido às seguidas interrupções ocasionadas. Em outros robôs, o problema não ocorreu até então e, além de ser possível reproduzir áudio no robô, foi possível conectar com sucesso em outros dispositivos de áudio com suporte a bluetooth.

Foi possível dar uma característica vocal pessoal para o robô, o qual, tendo sua identidade, possui um timbre específico de voz predeterminado em código. Porém, houveram dificuldades na reprodução da voz do robô com mesmo timbre para os diferentes idiomas implementados pela respectiva biblioteca. Uma mesma frase, utilizando os mesmos parâmetros de voz para diferentes idiomas, notavelmente gerava timbres de voz distintos. Portanto, para manter a identidade do robô, houve a necessidade de criar uma função de parametrização para cada um dos idiomas de forma a manter a voz do robô o mais semelhante possível, independente do idioma falado. A imagem da Fig. A.1 mostra os métodos de parametrização, descritos em Python, da voz do robô para algumas línguas.

Um problema enfrentado especificamente na língua portuguesa foi a impossibilidade de adicionar acentuação a qualquer palavra na ferramenta, o que fez com que a entonação do robô falando português fosse incorreta e estranha, causando estranheza ao usuário que o ouvia. Em uma tentativa de corrigir isso, as frases passadas como parâmetro nas funções de geração de áudio eram editadas acrescentando caracteres ou escrevendo as palavras de outra forma com intuito de alcançar uma pronúncia o mais próxima possível da correta.

4.0.2 Motores dos braços e cabeça

Os motores responsáveis pelos movimentos dos braços e pernas funcionaram corretamente quando o respectivo *script* de teste do módulo era executado, possibilitando

tanto o controle quanto um sensoriamento de poses do robô. A máxima taxa de comunicação alcançada com os motores foi de 1Mbps, com poucas perdas de pacotes. A conexão do braço com o tronco do robô, feita através de um parafuso, que prende o braço ao tronco sendo parafusado em um motor do próprio tronco, caiu em algumas ocasiões de uso do robô, mais especificamente quando o robô fazia movimentos com seus braços ou quando eles eram puxados por alguém. O parafuso que prendia o braço ao tronco acabava soltando do motor do tronco devido ao seu comprimento curto para conectar os dois membros.

Alguns movimentos do robô e suas amplitude dos motores Dynamixel são apresentados a seguir. A cabeça do robô possui três graus de liberdade, permitindo movimentação em torno dos três eixos *Roll*, *Pitch* e *Yaw*. Cada um dos braços possui três graus de liberdade que, somados aos da cabeça, totalizam nove graus de liberdade, fora as rodas. Os movimentos dos motores 0 e 1, mostrados na imagem 3.15, são limitados em software, não permitindo o robô colocar seus braços para trás do seu corpo.

Figura 4.1 – Movimentos ao entorno do eixo *Pitch* da cabeça

Fonte: Elaborada pelo autor

Figura 4.2 – Movimentos ao entorno do eixo *Yaw* da cabeça

Fonte: Elaborada pelo autor

Figura 4.3 – Alguns movimentos dos braços do robô

Fonte: Elaborada pelo autor

4.0.2.1 Demais sensores e atuadores

A câmera e microfone utilizados funcionaram relativamente bem. Foi possível a implementação de um algoritmo de detecção de rostos com a imagem da câmera, o qual apenas ficou com o desempenho comprometido devido à baixa capacidade de processamento do Raspberry Pi para esse tipo de aplicações utilizando processamento de imagem. Porém, em alguns momentos a câmera não era identificada pelo computador. O microfone funcionou sem nenhum *bug*, sendo possível captar áudios do ambiente com facilidade através da utilização tanto na execução do *script* de teste do módulo "Audio" quanto de exemplos feitos utilizando o módulo. Foi possível também, de acordo com as limitações de qualidade da captação de áudio, programar um código de reconhecimento de fala para uso em uma aplicação de conversação do robô.

Apesar de contar apenas com cinco expressões distintas, as quais podem ser expandidas em número, o robô cativou às pessoas com as quais interagia e expressava suas

"emoções" através de suas expressões em seus *displays* OLED. Em raras ocasiões os *displays* dos olhos apresentavam mal funcionamento, não sendo identificados como conectados ao robô pelo computador do mesmo.

O sensor infravermelho presente na extremidade do braço direito do robô mostrou-se de pouca utilidade e na prática não foi utilizado. O desenvolvimento de todos os *softwares* e *hardwares* necessários ao funcionamento dele foram feitos pelo autor. Sua implementação foi feita com base na biblioteca pronta para Linux, LIRC, porém nenhum dispositivo ao alcance do robô era suportado pela biblioteca e, portanto não estavam descritos na mesma. Logo, o módulo não pode ser testado.

O sensor Ultrassônico funcionou e possibilitou a medição de distância de acordo com sua limitação de alcance. Assim como para a câmera, houve a necessidade de cortar seu fio original e posterior soldagem do fio para que o sensor coubesse fisicamente na cabeça do robô. Seguidamente houve problema na identificação da distância medida, a qual permanecia como a distância máxima que pode ser lida, independente de haver um objeto a uma distância menor em frente ao sensor.

Os botões e LEDs se mostraram úteis e funcionais. Tanto a descrição do comportamento desses componentes através de bibliotecas descritas em Python no robô, quanto sua implementação física, ficaram sob responsabilidade de autor. Esses componentes funcionaram e serviram bem como interface para iniciar processos no robô, assim como forma de demonstrar o status dos mesmos, ou seja, se os programas sendo executados estavam prontos para serem iniciarem ou não. Houve um problema de projeto na placa de circuito impresso desenvolvida para abrigar todos os componentes e conexões do robô, nela faltou uma trilha que conectava o pino de GPIO do Raspberry Pi de controle ao seu respectivo LED controlado.

Outro problema relacionado aos LEDs e botões que ocorreu foi o encaixe deles nas aberturas destinadas a eles na mochila. Devido à placa de circuito impressa ilhada utilizada ter um padrão de distância entre seus furos, os componentes foram soldados de acordo com essas distâncias e, por isso, não ficaram alinhados com os buracos da mochila e, portanto, não se encaixam bem neles. Isso fez com que o terceiro botão de cima para baixo ficasse travado e fosse identificado como pressionado pelo algoritmo que recebia o sinal dos botões.

4.0.3 Componentes controlados pelo Arduino

4.0.3.1 Rodas e Sensor Capacitivo de Toque

Houveram e ainda ocorrem vários problemas com a utilização das rodas, os quais ainda não foram bem diagnosticados e resolvidos. Alguns deles foram resolvidos, como a diferente polaridade existentes em modelo de motores iguais, o que fazia com que, aplicados a mesma polaridade de alimentação, motores distintos poderiam rotacionar em sentidos opostos. Isso foi facilmente resolvido acrescentando-se uma contante no código do Arduino indicando a polaridade de cada um dos motores conectados ao robô.

Há um grande problema sendo enfrentado no âmbito das rodas, que é a perda de controle das mesmas devido à perda de comunicação entre as unidades de controle ou devido a uma aceleração descontrolada da roda. No primeiro caso, a comunicação entre o Arduino e o Raspberry Pi é perdida, impedindo a troca de mensagens entre os dois dispositivos. No caso do segundo problema, esporadicamente as rodas começam a acelerar sem parar e a partir desse momento não obedecem a nenhum comando dado posteriormente.

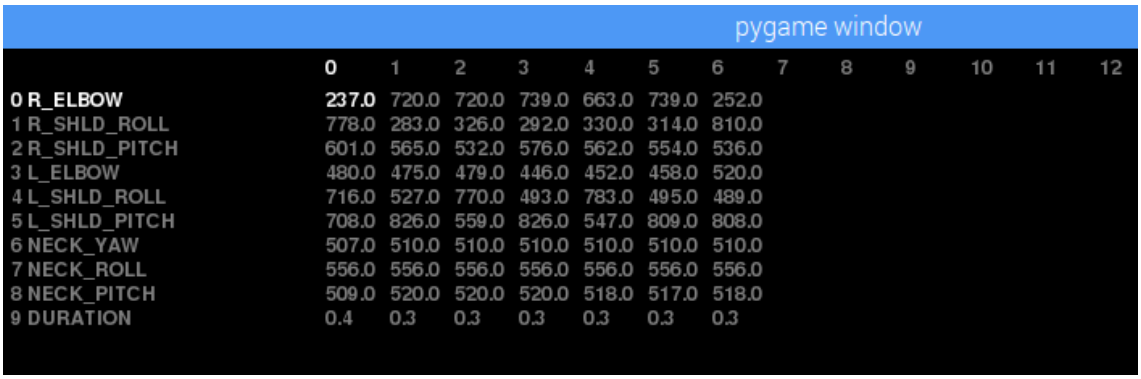
Houve certa dificuldade na implementação do sensor capacitivo de toque devido à variação das leituras feitas do sensor pelo Arduino devido a mudanças externas, como por exemplo, o fato do robô estar ou não conectado à fonte de energia. Para resolver tal questão foi feito um algoritmo que fica constantemente lendo os valores recebidos e fazendo uma média aritmética deles, a qual muda drasticamente quando há uma pessoa tocando o sensor, possibilitando a identificação do toque independente do robô de estar ou não conectado à energia elétrica. No controle PID foram utilizados os valores de 0,275 para o ganho proporcional, 0,125 para o ganho integral e um ganho nulo para a componente derivativa. Além disso, o cálculo da velocidade presente nas rodas considerava uma média ponderada, considerando 20% da velocidade medida no momento e 80% da velocidade já calculada anteriormente, para atualizar o valor de velocidade.

4.0.4 Ferramentas Disponíveis: Editor de Movimento e Snap!

Em relação as ferramentas disponíveis pelo robô, o editor de movimentos possibilita a produção de qualquer movimento dos membros do robô, exceto as rodas. Muitas apresentações em feiras e eventos utilizaram essa ferramenta para gerar os movimentos do robô. Um exemplo utilizando essa funcionalidades do robô pode ser visto em Robotics (2018b), onde o robô se apresenta e fala de suas capacidades. Nesse caso, o editor de movimentos foi utilizado para gerar a movimentação dos braços e cabeça do robô de forma

sincronizada a sua fala, cujas frases por sua vez foram geradas através do módulo "Audio". Um exemplo de movimento produzido utilizando a ferramenta de editor de movimentos é apresentado na Fig. 4.4.

Figura 4.4 – Exemplo de movimento criado utilizando o editor de movimentos



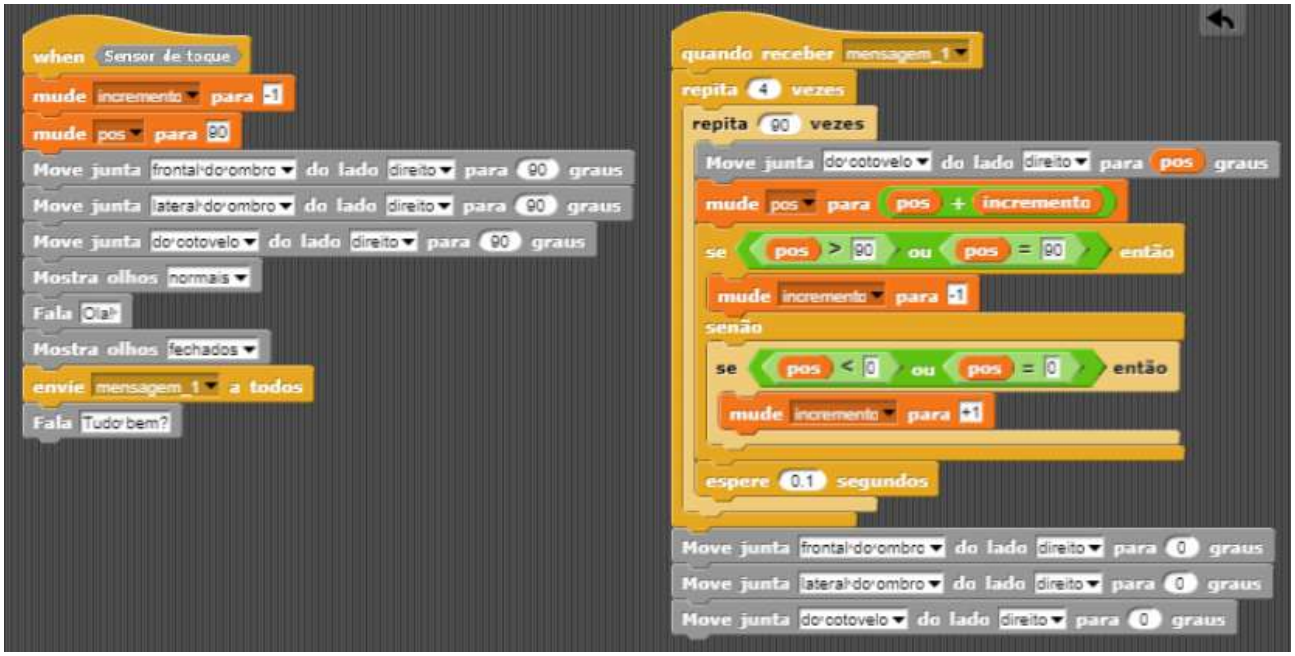
Fonte: Elaborada pelo autor

A integração do *software* Snap! ao robô ofereceu a vantagem de uma interface de programação fácil e intuitiva por meio da programação visual, uma alternativa à programação Python. O *software* permite criação de blocos personalizados para controle direto do robô no programa, os quais foram apresentados na Fig. 3.23. Ainda há limitação de utilização das capacidades do robô nessa interface, pois nem todas as funcionalidades foram implementadas em blocos, como é o caso das inúmeras funções que podem ser utilizadas com os motores Dynamixel, assim como as outros sensores que não aparecem implementados na Fig. 3.23, entre eles os botões e os LEDs.

Existe a dificuldade em gerar movimentos suaves, diferentemente da interpolação existente no editor de movimentos padrão, causando movimentos bruscos ou a má execução do movimento, dependendo da implementação de código feita. Isso pode ocorrer caso os blocos de controle dos motores dos braços e cabeça sejam utilizados em sequência sem utilização de blocos de intervalo de tempo, fazendo com que apenas a posição atribuída pelo último bloco da sequência seja atribuída ao robô. Os problemas com acentuação na língua portuguesa continuam aqui também, e há necessidade de ter internet para que o bloco "Fala" funcione.

A Fig. 4.5 apresenta um algoritmo desenvolvido na interface Snap!. Esse código espera até que o sensor de toque seja acionado para iniciar sua execução. Em seguida, o robô coloca seu braço direito em uma posição inicial com um ângulo de 90º para cada um dos motores presentes nele e em seguida fala "Ola" e muda o estado dos olhos para "fechados". Logo a seguir o robô executa o bloco de fala "Tudo bem" concomitantemente com o conjunto de blocos à direita da imagem, o qual é o conjunto de comandos necessários para o Beo mexer a mão direita simulando um aceno de olá. Por último, o braço direito retorna à posição reta junto ao tronco do robô.

Figura 4.5 – Exemplo de código criado utilizando o *software* Snap!



Fonte: Elaborada pelo autor

5 CONCLUSÃO

A proposta deste trabalho, que era o desenvolvimento de um robô humanoide para ser usado como ferramenta no ensino de programação à crianças, já alfabetizadas, foi feita. O robô e seu desenvolvimento foram apresentados, assim como uma breve contextualização em meio a outros estudos acadêmicos da área. A comparação com os trabalhos relacionados sugere que a ferramenta está no caminho correto. Ela utiliza programação visual, a qual é adequada para pessoas que nunca tiveram contato com programação antes, especialmente crianças e jovens, como é afirmado em Maloney et al. (2010). Segundo Andrade, Silva e Oliveira (2013), esse tipo de linguagem permite que a criança não tenha que se preocupar com erros de sintaxe. Esse tipo de linguagem permite fácil assimilação e torna a programação intuitiva, além de permitir que o foco do ensino seja nas estruturas de controle e lógica de programação, tirando toda preocupação de cumprimento de sintaxe existente nas linguagens de programação em texto (TOREZANI, 2014). Esse tipo de *software* de programação, utilizando blocos dá liberdade à criatividade da criança, permitindo a criação de inúmeros tipos de jogos e animações utilizando os blocos padrões da ferramenta e, no caso do Beo, a criança poderá utilizar a interface também para controlar todos os sensores a atuadores existentes no robô.

O Beo traz, ainda, como ferramenta física, conceitos de Robótica Educacional à sala de aula, assim como os benefícios descritos pela teoria construcionista de Papert. O robô ainda se apresenta ao aluno como um "alguém", e não um "algo", o que torna a relação da ferramenta com o aluno diferente, podendo tornar o processo de aprendizagem mais atrativo e encantador para ele.

A análise dos trabalhos apresentados no âmbito da utilização de kits de robótica para o ensino de programação, mostra que a maioria dos robôs montados nas atividades são tipo carros, ou semelhantes, contendo rodas e alguns sensores, com exceção do kit LEGO apresentado, o qual permite a construção de robôs mais diversos. Isso permite cogitar que as ferramentas de kit apresentam certa limitações em relação à quantidade de aplicações que podem ser produzidas a partir delas, tanto em relação à estrutura física quanto à quantidade de sensores e atuadores presentes neles. Há, também, a questão de que a dificuldade em montar uma estrutura física de robô mais complexo, por exemplo um humanoide, é maior. O tempo demandado por essa tarefa pode tornar inviável a construção, em sala de aula, de uma ferramenta mais complexa, com um número maior de aplicações possíveis, sensores e atuadores a serem controlados pelo aluno, devido ao curto período de tempo possivelmente disponível para tal tarefa.

Além dos resultados como ferramenta, a experiência do desenvolvimento do robô possibilitou além do enriquecimento de conhecimento na área de graduação, o amadurecimento profissional e pessoal do autor. O trabalho foi motivador e desafiador, possibilitando

a cada novo dia a aplicação prática dos conteúdos vistos em disciplinas teóricas da grade curricular do curso. O período de produção do robô promoveu a consolidação de conhecimentos na área de protocolos de comunicação, produção de *software* para robótica, e a incorporação de inúmeros outros novos por parte do autor, especialmente na área de projetos de *hardware*.

Embora existam vários estudos na área confirmando os benefícios da robótica educacional, um claro passo seguinte a este trabalho, é a realização de uma pesquisa com dados da utilização da ferramenta por alunos. Isso permitiria uma análise de resultados práticos gerados pela aplicação do robô no ambiente de sala de aula no ensino de programação a crianças, disponibilizando dados para servirem como base para verificar se a ferramenta cumpre de fato seu propósito de ser uma plataforma lúdica e intuitiva para usuários iniciantes, ao mesmo tempo que proporciona possibilidade de aplicações mais complexas para usuários mais avançados, ou não.

REFERÊNCIAS BIBLIOGRÁFICAS

ACKERMANN, E. Piaget's constructivism, papert's constructionism: What's the difference? In: . [S.l.: s.n.], 2001.

____. Constructing knowledge and transforming the world. In: ____. [S.l.: s.n.], 2004. cap. 2, p. 15–37.

ALBUQUERQUE, A. P. et al. Robótica pedagógica livre: Instrumento de criação, reflexão e inclusão sócio-digital. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. [S.l.: s.n.], 2007. v. 1, n. 1, p. 316–319.

ALIMISIS, D. Educational robotics: Open questions and new challenges. **Themes in Science & Technology Education**, v. 6, n. 1, p. 63–71, 2013.

AMARAL, L. R. do; SILVA, G. B. e; PANTALEAO, E. Plataforma robocode como ferramenta lúdica de ensino de programação de computadores - extensão universitária em escolas públicas de minas gerais. In: **XXVI Simpósio Brasileiro de Informática na Educação**. [S.l.: s.n.], 2015.

ANDRADE, M.; SILVA, C.; OLIVEIRA, T. Desenvolvendo games e aprendendo matemática utilizando o scratch. In: UNIVERSIDADE PRESBITERIANA MACKENZIE, 12., 2013, São Paulo, Brasil. **XII Simpósio Brasileiro de Jogos e Entretenimento Digital**. Brasil: Universidade Presbiteriana Mackenzie, 2013. Acesso em: 30/12/2018. Disponível em: $<http://www.sbgames.org/sbgames2013/>$.

AROCA, R. V. **Plataforma robótica de baixíssimo custo para robótica educacional**. Dezembro 2012. Dissertação — Universidade Federal do Rio Grande do Norte, Natal, RN, Dezembro 2012.

BELLIZIA, P. **Os empregos do futuro dependem da educação do presente**. 2017. Acesso em 27 dez. 2018. Disponível em: $<https://www.linkedin.com/pulse/os-empregos-do-futuro-dependem-da-educa\%C3\%A7\%C3\%A3o-presente-paula-bellizia/>$.

CARBAJAL, M. L.; BARANAUSKAS, M. C. C. Taprec: Desenvolvendo um ambiente de programação tangível de baixo custo para crianças. In: UNIVERSIDAD DE CHILE, FACULTAD DE CIENCIAS FíSICAS Y MATEMáTICAS, 20., 2015, Santiago, Chile. **Congresso Internacional de Informática Educativa**. Chile: Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas, 2015. Acesso em: 05/12/2018. Disponível em: <http://www.tise.cl/volumen11/TISE2015/TISE%202015.pdf>.

DANTAS, R. F.; COSTA, F. E. A. da. Code: O ensino de linguagens de programação educativas como ferramentas de ensino/aprendizagem. In: UNIVERSIDADE FEDERAL DE PERNAMBUCO, 5., 2013, Pernambuco. **Simpósio Hipertexto e Tecnologias na Educação**. Pernambuco, 2013. Acesso em: 05/12/2018. Disponível em: <http://nehte.com.br/simposio/anais/simposio2013.html>.

FORUM, W. E. **The Future of Jobs**. 2016. Acesso em 27 dez. 2018. Disponível em: $<http://reports.weforum.org/future-of-jobs-2016/>$.

FREIRE, P.; SHOR, I. **Medo e Ousadia: O cotidiano do professor**. Rio de Janeiro: Paz e Terra, 2008.

GALDINO, C. B. T.; NETO, S. R. da S.; COSTA, E. de B. Kidcoder: Uma proposta de ensino de programação de forma lúdica. In: **XXVI Simpósio Brasileiro de Informática na Educação**. [S.l.: s.n.], 2015.

GIL, A. C. **Como Elaborar Projetos de Pesquisa**. 4. ed. São Paulo: Atlas, 2002.

GOOGLE. **Blockly**. 2018. Acesso em 23 dez. 2018. Disponível em: $<https : //developers.google.com/blockly/>$.

HARVEY, B.; MöNIG, J. **Snap! Reference Manual**. 4.1. ed. [S.l.]. 109 p. Acesso em 17 dez. 2018. Disponível em: $<https : //snap.berkeley.edu/SnapManual.pdf>$.

KUREBAYASHI, S.; KAMADA, T.; KANEMUNE, S. Learning computer programming with autonomous robots. In: **International Conference on Informatics in Secondary Schools**. [S.l.: s.n.], 2006.

KUSUMA, I. D.; UTAMININGRUM, F.; KAKESHITA, T. A toolkit to learn algorithmic thinking using mbot robot. **IPSJ SIG Technical Reports**, 2018.

LINDBLOM, J. **Serial Communication**. Dezembro 2012. Acesso em 8 dez. 2018. Disponível em: $<https : //learn.sparkfun.com/tutorials/serial - communication>$.

MALONEY, J. et al. The scratch programming language and environment. **Journal on Educational Resources in Computing**, ACM Transactions on Computing Education, v. 10, n. 4, p. Article No. 16, 2010.

MARTINS, L. A. da S. et al. Ensinando lógica de programação aplicada à robótica para alunos do ensino fundamental. In: **XXVII Simpósio Brasileiro de Informática na Educação**. [S.l.: s.n.], 2016.

MEDEIROS, T. J.; SILVA, T. R. da; ARANHA, E. H. da S. Ensino de programação utilizando jogos digitais: uma revisão sistemática da literatura. **Revista Novas Tecnologias na Educação**, 2013.

OLIVEIRA, G. A. A. de et al. Grubibots educacional: jogo para o ensino de algoritmos na educação básica. In: **III Congresso Brasileiro de Informática na Educação**. [S.l.: s.n.], 2014.

PAPADAKIS, S. et al. Using scratch and app inventor for teaching introductory programming in secondary education. a case study. **International Journal of Technology Enhanced Learning**, 2016.

PAPERT, S. **Mindstorms: children, computers, and powerful ideas**. New York: Basic Books, 1980.

PARK, I. et al. Learning effects of pedagogical robots with programming in elementary school environments in korea. **Indian Journal of Science and Technology**, v. 8, n. 26, October 2015.

POT, E. et al. Choregraphe: a graphical tool for humanoid robot programming. In: **Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on Robot and Human Interactive Communication**. Toyama, Japan: IEEE, 2009. p. 46–51.

PWC. **UK Economic Outlook**. 2017. Acesso em 26 dez. 2018. Disponível em: $<https : //www.pwc.co.uk/economic - services/ukeo/pwc - uk - economic - outlook - full - report - march - 2017 - v2.pdf>$.

RIBEIRO, C. R. **RobôCarochinha: Um Estudo Qualitativo sobre a Robótica Educativa no 1º ciclo do Ensino Básico**. Outubro 2006.

ROBOTICS, Q. **Beo**. 2018. Acesso em 7 dez. 2018. Disponível em: $<http : //qironrobotics.com/robos/>$.

____. **Conheça o Beo**. 2018. Acesso em 15 dez. 2018. Disponível em: <https://www.youtube.com/watch?v=TeRZ2rDMFoc>.

Robotis INC. **ROBOTIS e-Manual**. 2018. Acesso em 7 dez. 2018. Disponível em: $<http : //emanual.robotis.com/>$.

SALEIRO, M. et al. A low-cost classroom-oriented educational robotics system. In: SPRINGER (Ed.). **Social Robotics - 5th International Conference**. Bristol, UK: [s.n.], 2013. p. pp. 74–83.

SAVAGE, J. A. **Arduino y Dynamixel AX-12**. janeiro 2011. Acesso em 7 dez. 2018. Disponível em: $<http : //savageelectronics.blogspot.com/2011/01/arduino - y - dynamixel - ax - 12.html>$.

SILVA, E. L. da; MENEZES, E. M. **Metodologia da Pesquisa e Elaboração de Dissertação**. 4ª edição revisada e atualizada. ed. Florianópolis: universidade Federal de Santa Catarina - UFSC, 2005. 138 p.

STALLINGS, W. **Data and Computer Communications**. 8. ed. Upper Saddle River, NJ: PEARSON Prentice Hall, 2007.

TOREZANI, C. **NewProg - Um ambiente online para crianças aprenderem programação de computadores**. 2014. Dissertação — Universidade Federal do Espírito Santo, Vitória, Espírito Santo, 2014.

APÊNDICE A – FUNÇÕES DE PARAMETRIZAÇÃO DA VOZ DO ROBÔ

Figura A.1 – Funções de parametrização da voz do robô para Português, Inglês e Espanhol, respectivamente

```
def ajusta_mp3(self, origem, destino):
    """
    Description: Modifies echo, pitch and speed of an mp3 audio to portuguese language
    Utilization: ajusta_mp3(origem, destino)
    Parameters:
     -origem: filename of the mp3 file to be modified
     -destino: filename for the modified mp3 file
    """
    print('Ajustando')
    print(destino)
    call(['sox', origem, destino,\
        'pitch', '150', 'speed', '1.25',\
        'echo', '0.8', '0.88', '60', '0.4'])
def adjust_mp3(self, origem, destino):
    """
    Description: Modifies echo, pitch and speed of an mp3 audio to english language
    Utilization: adjust_mp3(origem, destino)
    Parameters:
     -origem: filename of the mp3 file to be modified
     -destino: filename for the modified mp3 file
    """
    print('Adjusting')
    call(['sox', origem, destino,\
        'pitch', '+175', 'speed', '1.2',\
        'echo', '0.8', '0.88', '60', '0.4'])
def ajusta_mp3_es(self, origem, destino):
    """
    Description: Modifies echo, pitch and speed of an mp3 audio to Spanish language
    Utilization: ajusta_mp3_de(origem, destino)
    Parameters:
     -origem: filename of the mp3 file to be modified
     -destino: filename for the modified mp3 file
    """
    print('Ajustando')
    call(['sox', origem, destino,\
        'pitch', '500', 'speed', '1.1',\
        'echo', '0.8', '0.88', '60', '0.4'])
```

Fonte: Elaborada pelo autor

APÊNDICE B – *SCRIPTS* DE TESTES DOS MÓDULOS DO ROBÔ

Figura B.1 – *Script* para teste do módulo "Microfone"

```
from Microphone import Microphone

phone = Microphone()
file_name = "testAudio.mp3"
duracao = 2

try:
    print("Fale durante " + str(duracao) + " segundos")
    phone.read_audio(duracao,file_name)
    print("Audio capturado")
finally:
    print("\n\nEnd of Microphone example")
```

Fonte: Elaborada pelo autor

Figura B.2 – *Script* para teste do módulo "PushButtons"

```
from PushButtons import PushButtons

pb = PushButtons()

try:
  while True:
    b = pb.listen_buttons()
    if b:
        print(b)

finally:
    print("\n\nEnd of Push Buttons example\n\n")
```

Fonte: Elaborada pelo autor

Figura B.3 – *Script* para teste do módulo "SensorTouch"

```
'''Touch sensor interface test'''

from time import clock
from SensorTouch import SensorTouch

S_TOUCH = SensorTouch()
LAST_TIME = clock()

while True:
    if(clock() - LAST_TIME) >= 0.05:
        LAST_TIME = clock()
        print(S_TOUCH.get_status())
```

Fonte: Elaborada pelo autor

Figura B.4 – *Script* para teste do módulo "Ultrasonic"

```
import time
from Ultrasonic import Ultrasonic
import RPi.GPIO as GPIO

try:
    u = Ultrasonic()
    while True:
        print ("Measured Distance = %.1f cm" %  u.get_distance())
        time.sleep(0.05)

    # Reset by pressing CTRL + C
except KeyboardInterrupt:
    print("Measurement stopped by User")
    GPIO.cleanup()
```

Fonte: Elaborada pelo autor

Figura B.5 – *Script* para teste do módulo "Audio"

```
'''Audio test'''
from Audio import Audio
from time import sleep

a = Audio()

#Audio pre-gravado
a.play_mp3("beo_audios/ola.mp3")
sleep(5)
a.say_number("9")
a.say_number("8")

#Audio gerado com conexao na internet
a.fala("Audio gerado instantaneamente!")
#sleep(3)
a.speak("Instantly generated audio!")
```

Fonte: Elaborada pelo autor

Figura B.6 – *Script* para teste do módulo "Eyes"

```
'''Test of Eyes module'''
from Eyes import Eyes
import time

EYES = Eyes()
EYES.set_expression('default')
time.sleep(0.5)
EYES.set_expression('wink')
time.sleep(0.5)
EYES.set_expression('mad')
time.sleep(0.5)
EYES.set_expression('sad')
time.sleep(0.5)
EYES.set_expression('shut')
time.sleep(0.5)
EYES.disp_clear()
print("End of example")
```

Fonte: Elaborada pelo autor

Figura B.7 – *Script* para teste do módulo "IrSensor"

```
'''infrared sensor example'''
from IrSensor import IrSensor

REMOTE_CONTROL_NAME = "philips_tv"
KEY_NAME = "key_power"

ir = IrSensor()

ir.status()

ir.list_controls()

ir.list_buttons(REMOTE_CONTROL_NAME)

ir.send_once(REMOTE_CONTROL_NAME, KEY_NAME)

print("End of IrSensor example")
```

Fonte: Elaborada pelo autor

Figura B.8 – *Script* para teste do módulo "Leds"

```
from Leds import Leds
import time

led = Leds()

#with Leds() as led:
try:
  while True:
    time.sleep(1)
    led.on('TOP')
    time.sleep(1)
    led.off('TOP')

    time.sleep(1)
    led.on('BOT')
    time.sleep(1)
    led.off('BOT')
except Exception as e:
    print(e)
finally:
    print("\n\nEnd of Leds example\n\n")
```

Fonte: Elaborada pelo autor

Figura B.9 – *Script* para teste do módulo "Movements"

```
from time import sleep
from random import randint, random, choice
from Eyes import Eyes
from Movements import Movements

eyes = Eyes()
movements = Movements()

eyes.set_expression('default')
movements.disable_all_joints()

goal = [0 for i in range(9)]
wink = 0
count = 0
while(1):
    count = (count + 1) % 1000
    if count == 999:
        movements.disable_all_joints()
    for i in range(3):
        angle = movements.get_raw_angle(i*2)
        movements.set_raw_angle(i*2 +1,angle)

    for i in range(6,9):
        error = (movements.joints_center[i]-goal[i]) - movements.goal_pos[i]
        value = movements.goal_pos[i] + int(error/20)
        movements.set_joint(i, value)

    if randint(0,100) == 0:
        goal[6] = randint(-1,1)*100
    if randint(0,100) == 0:
        goal[6] = 0
        goal[7] = randint(-1,1)*100
    if randint(0,100) == 0:
        goal[8] = randint(-1,1)*100
    if randint(0,75) == 0:
        if randint(0,15) == 0 and wink == 0:
            eyes.set_expression('shut')
            wink = 1
        elif wink == 1:
            eyes.set_expression('default')
            wink = 0
```

Fonte: Elaborada pelo autor

Figura B.10 – *Script* para teste do módulo "Wheels"

```
import time
from Wheels import Wheels

w = Wheels()

TIME_START = time.time()

w.move_forward(2)
#w.move_reverse(2)

while (time.time() - TIME_START) < 4.0:
        time.sleep(0.005)
        print("Wheel Left: " + str(w.wheel_l.get_present_position())  + "\t" + \
              "Wheel Right: " + str(w.wheel_r.get_present_position()))
w.move_stop()
```

Fonte: Elaborada pelo autor