PUCRS

FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

JULIANA DAMASIO OLIVEIRA

GODONNIE: DEFINIÇÃO E AVALIAÇÃO DE UMA LINGUAGEM DE PROGRAMAÇÃO PARA COMANDAR ROBÔ POR PROGRAMADORES INICIANTES COM DEFICIÊNCIA VISUAL

Porto Alegre
2017

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GODONNIE: DEFINIÇÃO E AVALIAÇÃO DE UMA LINGUAGEM DE PROGRAMAÇÃO PARA COMANDAR ROBÔ POR PROGRAMADORES INICIANTES COM DEFICIÊNCIA VISUAL

JULIANA DAMASIO OLIVEIRA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientadora: Prof. Dra. Márcia de Borba Campos
Coorientador: Prof. Dr. Alexandre de Moraes Amory

Porto Alegre
2017

Ficha Catalográfica

O48 Oliveira, Juliana Damasio

GoDonnie : definição e avaliação de uma linguagem de programação para comandar robô por programadores iniciantes com deficiência visual / Juliana Damasio Oliveira . – 2017.

189 f.

Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientadora: Profa. Dra. Márcia de Borba Campos.

Co-orientador: Prof. Dr. Alexandre de Moraes Amory.

1. pessoa com deficiência visual. 2. linguagem de programação. 3. orientação e mobilidade. 4. robótica educacional. 5. ensino de programação. I. Campos, Márcia de Borba. II. Amory, Alexandre de Moraes. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS com os dados fornecidos pelo(a) autor(a).

Juliana Damasio Oliveira

GoDonnie: definição e avaliação de uma linguagem de programação para comandar robô por programadores iniciantes com deficiência visual

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciência da Computação do Programa de Pós-Graduação em Ciencia da Computação, Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado em 04 de agosto de 2017.

BANCA EXAMINADORA:

Profa. Dr. Milene Selbach Silveira (PPGCC/PUCRS)

Profa. Dra. Lucia Maria Martins Giraffa (PPGEDU/PUCRS)

Profa. Dra. Márcia de Borba Campos (PPGCC/PUCRS - Orientador)

AGRADECIMENTOS

Em primeiro lugar, gostaria de agradecer imensamente a orientadora deste trabalho profa. Márcia de Borba Campos, por ter acreditado em mim desde a graduação quando orientou meu Trabalho de Conclusão. Além disso, agradeço por toda dedicação, incentivo, paciência e amizade. Foi um caminho com muitos ensinamentos, que vou levar para sempre comigo. Também agradeço ao Prof. Alexandre Amory, por ter aceitado coorientar este trabalho e por todo auxílio prestado.

Agradeço ao meu marido Paulo Caliendo, pelo apoio e amor. Sem essa fortaleza em casa certamente não seria possível a conclusão deste trabalho. À minha família, muito obrigada por torcer sempre por mim.

Aos meus amigos – não colocarei nomes para não cometer o equívoco de esquecer alguém –, meu muito obrigada pelas horas de descompressão. Com certeza vocês aliviaram a grande pressão que é realizar um trabalho com essa seriedade.

Ao grupo do laboratório LSA, agradeço por terem dividido comigo esse projeto e por toda a ajuda prestada, implementando o Ambiente de programação *Donnie*, que é o foco deste trabalho. Um agradecimento especial, a Camila Kolling, Henry Nunes, Lucas Chaves, Alisson Kissel, Joice Marek, Augusto Bergamin, Daniel Einloft e Guilherme Marques.

Não existem palavras que consigam expressar o meu agradecimento aos participantes dessa pesquisa. Muito obrigada por todas as sugestões e críticas. Sem vocês este trabalho não teria se concretizado.

Ao Raí Cabeleira e Diego Campelo, meus profundos agradecimentos. Vocês me ensinaram mais do que possam imaginar. Muito obrigada pela dedicação e comprometimento.

Ao Programa de Pós-graduação em Ciência da Computação (PPGCC), agradeço por ter me aceitado nesse programa e pelo fornecimento de bolsa-taxas.

Aos colegas de pós, que estiveram esse período ao meu lado, obrigada pelo companheirismo, amizade e auxílio prestado.

À CAPES pela concessão de bolsa integral durante o período do mestrado.

Agradeço à profa. Milene Silveira e à profa. Lúcia Giraffa, por terem aceitado ser avaliadoras deste trabalho e por toda contribuição prestada.

Por fim, agradeço a todos que contribuíram de forma direta ou indireta para a realização deste trabalho. Muito obrigada a todos.

GODONNIE: DEFINIÇÃO E AVALIAÇÃO DE UMA LINGUAGEM DE PROGRAMAÇÃO PARA COMANDAR ROBÔ POR PROGRAMADORES INICIANTES COM DEFICIÊNCIA VISUAL

RESUMO

O objetivo geral desse trabalho foi definir e avaliar uma linguagem de programação para comandar robô para estimular as habilidades de orientação e mobilidade (O&M) por pessoas com deficiência visual (DV). Acredita-se que ao utilizar uma linguagem de programação para guiar um robô em um cenário virtual, a pessoa com deficiência visual possa melhor compreender habilidades de O&M. Neste sentido, foi definida a linguagem de programação, que foi denominada *GoDonnie*, e foi baseada na linguagem Logo. A *GoDonnie* é executada em um ambiente de programação chamado *Donnie*, que fornece *feedbacks* sonoros sobre a execução do robô. Além disso, possui um simulador gráfico 2D com um robô virtual, no qual se pode visualizar a execução dos comandos da linguagem, ou seja, o robô virtual se deslocando no cenário. Esse simulador permite o uso por pessoas com baixa visão, além de servir como recurso para melhor visualizar como a pessoa que é cega está interagindo com o robô no ambiente. A *GoDonnie* foi avaliada para verificar a sua usabilidade. Foram realizadas avaliações que incluíram pessoas com deficiência visual e professores de programação, sem deficiência visual. Os resultados apontaram que a *GoDonnie* auxilia o desenvolvimento de O&M em pessoas com DV e atende ao esperado no que tange ao ambiente de programação.

Palavras-Chave: pessoa com deficiência visual, linguagem de programação, orientação e mobilidade, robótica educacional, ensino de programação.

GODONNIE: DEFINITION AND EVALUATION OF A PROGRAMMING LANGUAGE TO COMMAND ROBOT BY BEGINNER PROGRAMMERS WITH VISUAL IMPAIRMENT

ABSTRACT

The general objective of this work was to define and evaluate a programming language to command robot to stimulate orientation and mobility (O&M) skills by people with visual impairment (VI). It is believed that by using a programming language to guide a robot in a virtual setting, the person with VI can better understand O&M skills. In this sense, it was defined the programming language, which was called GoDonnie, and was based on the Logo language. A GoDonnie is run in a programming environment called Donnie, which provides soundbacks about running the robot. In addition, it has a 2D graphic simulator with a virtual robot, in which one can visualize the execution of the language commands, that is, the virtual robot sedeslocando in the scenario. This simulator allows the use by people with low vision, as well as serve as a resource to better visualize how the person who is blind is interacting with the robot in the environment. A GoDonnie has been evaluated to verify its usability. Evaluations were carried out which included visually impaired people and teachers of programming, without visual impairment. The results pointed out that *GoDonnie* supports the development of O&M in people with VI and meets the expectations regarding the programming environment.

Keywords: person with visual impairment, orientation and mobility, programming language, educational robotics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

APD – Ambiente de Programação *Donnie*

DV – Deficiência Visual

EVA – *Etil Vinil Acetato*

IC – Iniciação Científica

IHC – Interação Humano-Computador

O&M – Orientação e Mobilidade

RSL – Revisão Sistemática da Literatura

TA – Tecnologia Assistiva

TCLE – Termo de Consentimento Livre e Esclarecido

SUMÁRIO

1. INTRODUÇÃO

O ensino de programação com uso de robôs vem sendo empregado como facilitador e incentivador do aprendizado [7, 49, 69]. Neste sentido, vários ambientes de robótica estão sendo propostos para facilitar o ensino de programação, porém, conforme Barros et al. [6], essas iniciativas não contemplam todos os alunos, especialmente os com deficiência visual (DV)[1], que são foco deste trabalho. Isso ocorre devido a muitos ambientes de programação serem baseados em interface gráfica, o que os tornam inacessíveis para esse grupo de usuários [6, 26, 32, 41].

Uma pessoa que possui DV necessita de um apoio complementar de estímulos não visuais para perceber o ambiente e conseguir construir mapas mentais do ambiente. Receber informações do espaço por meio de outros sentidos, como audição e tato, colabora com a criação de mapas mentais como representação do ambiente [38]. Mapa mental é uma estrutura utilizada para a formação e organização de ideias. De acordo com Marques [45], a estrutura de mapas mentais é apresentada em forma de diagrama, em que, por meio de um conceito chave, outras informações são ligadas a esse conceito na forma de ramos, formando uma rede de ideias. As pessoas que possuem DV podem fazer uso de outras estratégias para externalizar um mapa mental, como por exemplo, por meio de maquetes. A intenção dessa estrutura é favorecer a memorização, facilitando assim a compreensão de assuntos de forma resumida.

Tendo domínio dos conceitos citados, os mapas mentais podem ser utilizados para vários propósitos, entre eles, o ensino de orientação e mobilidade (O&M) para pessoas com DV. Para essas pessoas, as habilidades de O&M são indispensáveis para a conquista da autonomia e, consequentemente, para sua independência e inclusão na escola e na sociedade [17]. Nesse sentido, pesquisas realizadas por Sánchez et al. [61, 62, 63] descrevem o uso de mapas mentais e mapas conceituais para formação da compreensão e aquisição das habilidades de O&M por pessoas com deficiência visual. O foco dos estudos foi o desenvolvimento e avaliação de sistemas interativos para deficientes visuais como apoio ao (re)conhecimento de ambientes por meio da construção de mapas mentais. Em Campos et al. [12], foi realizada uma avaliação da aplicação de áudio chamada mAbES, que mapeava experimentos do Museu de Ciência e Tecnologia PUCRS (MCT-PUCRS). Neste estudo foi verificado que os usuários conseguiram representar o espaço real do museu por meio de maquetes táteis, a partir do uso do sistema.

Ao contrário de pesquisas que tem por objetivo o desenvolvimento de Tecnologia Assistiva (TA) para guiar uma pessoa que é DV em um espaço, o propósito desse trabalho

[1] Conforme o Decreto no 5296/2004, a deficiência visual inclui "cegueira, na qual a acuidade visual é igual ou menor do que 0,05 no melhor olho, com a melhor correção óptica; a baixa visão, que significa acuidade visual entre 0,3 e 0,05 no melhor olho, com a melhor correção óptica." Link: http://www.planalto.gov.br/ccivil_03/_ato2004−2006/2004/decreto/d5296.htm

foi fazer com que uma pessoa com DV pudesse criar mapas mentais ao utilizar uma linguagem de programação para guiar um robô por um espaço determinado, e, assim, melhor compreender habilidades de O&M. Neste sentido, foi elaborada uma linguagem de programação chamada *GoDonnie*, baseada na linguagem Logo[2]. A *GoDonnie* foi criada para ser utilizada como uma ferramenta no auxílio de resolução de problemas espaciais que envolvam O&M. Foi concebida sobre critérios de acessibilidade e usabilidade visando seu uso por pessoas com deficiência visual. Existe também um cenário virtual que contém um robô chamado *Donnie* que deve obedecer aos comandos da *GoDonnie* e emitir um *feedback* sonoro ao usuário. O pressuposto é que, usando a *GoDonnie* e o cenário virtual, o usuário com deficiência visual possa melhorar sua análise, programação e suas habilidades de O&M. Além disso, o aspecto lúdico da robótica pode encorajar, especialmente jovens estudantes, a usarem o ambiente proposto e adquirirem mais habilidade em programação e em O&M, a partir de seu uso.

Dentre as contribuições desta dissertação, destacam-se: (I) definição da linguagem de programação *GoDonnie* para pessoas com DV; (II) metodologia de avaliação para linguagens de programação baseadas em robótica quanto à usabilidade e O&M; (III) metodologia de avaliação para ambientes de programação com uso de robótica para desenvolvimento de programação e O&M; e (IV) protocolo e resultado da Revisão Sistemática da Literatura (RSL) sobre o ensino de programação com uso de robótica.

1.1 Objetivos

O objetivo geral dessa pesquisa, é definir e avaliar uma linguagem de programação para estimular as habilidades de orientação e mobilidade por pessoas com deficiência visual, a partir do uso de programação de robô.

Os objetivos específicos são:

- Elaborar uma linguagem de programação para comandar um robô que possa ser utilizada por usuários que possuem DV na resolução de problemas de lógica e de O&M.
- Avaliar a linguagem de programação que comanda o robô, junto a pessoas que possuem deficiência visual, no que se refere:
  - à usabilidade
  - ao apoio do desenvolvimento de habilidades de orientação e mobilidade
- Verificar como a programação de robôs pode ser utilizada como um recurso para apoiar a solução de problemas relacionados às habilidades de orientação e mobilidade de pessoas que possuem deficiência visual.

[2]Durante o trabalho quando for falado Logo, refere-se ao Logo e as suas variações.

1.2 Justificativa e relevância da pesquisa

Escolher a linguagem de programação inicial para estudantes com deficiência visual ainda é um problema em aberto. Trabalhos relatam que algumas linguagens são de difícil uso para esse grupo de usuários [32, 48]. Entre linguagens de programação, que apresentam dificuldades, cita-se o *Python*, que usa espaços em branco para delimitar blocos de código, que é de difícil navegação com leitores de tela [32, 48]; *C* e *Java*, que usam chaves para delimitar blocos de código, neste caso o obstáculo é quando há um bloco grande com sub-blocos porque os alunos com deficiência visual tendem a ter dificuldade para encontrar chaves relacionadas. Outra questão é que os ambientes de programação são normalmente gráficos e sem retorno de áudio ao usuário, tornando-os inacessíveis às pessoas com DV [32].

Aliado a isso, outro aspecto importante, é que as técnicas tradicionais de ensino baseiam-se principalmente em modelos visuais para ajudar na compreensão de informações complexas, tais como diagramas, fluxogramas, tabelas e imagens. Infelizmente, esse tipo de recurso não é útil a alunos com DV, uma vez que não permitam o uso de recursos concretos ou de tecnologia assistiva [2].

De acordo com Garcia e Galvão [21], os estudos e análises referentes aos processos de pesquisa e desenvolvimento na área da Tecnologia Assistiva no Brasil ainda são bastante escassos. Ainda existe uma volumosa necessidade de recursos de TA, relacionados ao uso do computador, para que possa ocorrer uma verdadeira inclusão das pessoas com deficiência em seus espaços [21].

Além disso, a Sociedade Brasileira de Computação (SBC) [13] pontuou os grandes desafios para a computação do Brasil para o período de 2006-2016, e entre eles está o “Acesso Participativo e Universal do cidadão brasileiro ao conhecimento”. A multidisciplinaridade é citada pela SBC como uma ação associada aos grandes desafios. Essa ação multidisciplinar deve ocorrer não apenas entre a Computação e outros domínios científicos, mas também dentro da própria Computação. Por exemplo, especialistas em Hardware precisam cooperar com especialistas em Redes, em Banco de dados, em Interação Humano-Computador. Neste sentido, essa dissertação demonstra a multidisciplinaridade que a SBC sugere, já que existe a integração de Interação Humano-Computador (IHC), Robótica, Educação e Programação.

Essa dissertação possui relevância social e relevância científica/tecnológica. O trabalho é relevante socialmente, pois contribui para atender as necessidades educacionais de alunos com deficiência visual, e no que se refere à orientação e mobilidade, que são necessárias à autonomia e melhor inclusão da pessoa com DV na sociedade. Além disso, ao próprio ensino de programação, ao ser disponibilizada uma linguagem de programação, e um ambiente para executá-la, acessíveis. Desta forma, se considera que, ao programar

um robô, a pessoa que é cega possa melhor compreender o espaço e os objetos que estão ao redor do mesmo. No que se refere à relevância científica e tecnológica, o trabalho traz contribuições para as áreas de Interação Humano-Computador, de Robótica Educacional e de Ensino de Programação.

1.3 Estrutura do Trabalho

Este documento foi organizado da seguinte forma: o Capítulo 2 apresenta a fundamentação teórica relacionada a este trabalho, o Capítulo 3 descreve a metodologia de pesquisa utilizada, o Capítulo 4 mostra os resultados obtidos na Revisão Sistemática da Literatura, o Capítulo 5 apresenta a linguagem de programação *GoDonnie*, o Capítulo 6 mostra as avaliações realizadas na linguagem *GoDonnie*, o Capítulo 7 descreve o ambiente de programação *Donnie*, o Capítulo 8 descreve a avaliação da *GoDonnie* junto ao ambiente de programação e o Capítulo 9 apresenta as considerações finais e os trabalhos futuros. Por fim, são apresentadas as referência utilizadas no texto e os apêndices, que complementam esse documento.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo encontra-se o referencial teórico relacionado a esta dissertação. Discute-se sobre os seguintes temas: orientação e mobilidade (O&M) (Seção 2.1), robótica educacional (Seção 2.2), pensamento computacional e ensino de programação (Seção 2.3) e usabilidade em linguagem de programação (Seção 2.4).

2.1 Orientação e mobilidade

O público-alvo deste trabalho são pessoas que possuem deficiência visual. A fim de compreender como as habilidades de orientação e mobilidade são desenvolvidas, faz-se necessário conhecer este fenômeno. Jacobson [27] define orientação como a capacidade de utilizar os sentidos remanescentes (a audição, o tato, o olfato, a cinestesia, etc.) para compreender sua localização no ambiente, enquanto que mobilidade é definida como a habilidade de se mover com facilidade em um ambiente. Programas de treinamento em orientação e mobilidade se referem a ensinar os conceitos e técnicas necessárias para que as pessoas com deficiência visual possam se locomover com mais segurança, mais autonomia, e de forma mais eficiente em diferentes ambientes e situações.

No estudo de Mazzaro et al. [47] são citados programas no qual as pessoas que são cegas aprendem a conhecer a si mesmas e, após, aprendem sobre a concepção do espaço relacionado a elas e aos objetos do ambiente, por meio de conceitos espaciais. Existem locais com programas específicos para desenvolver e treinar O&M, tais como: a *American Foundation for the Blind*[1], *Association for Education and Rehabilitation of the blind and visually impaired*[2], *National Center on Deaf-Blindness*[3], União de Cegos do Rio Grande do Sul (UCERGS)[4] e Escola de Cegos Santa Luzia[5]. Para as pessoas que possuem retorno visual, essas habilidades são desenvolvidas desde a infância por meio da visão, na qual são aprendidos conceitos de distância, espaço, tamanhos, formas de objetos e localização de objetos. Já para as pessoas que possuem DV, é necessário utilizar outras formas de estímulos. Por isso, os programas já mencionados se tornam importantes na formação dos conceitos de relação espacial.

De acordo com Lahav e Mioduser [37], criar mapas mentais de ambientes é algo essencial para o desenvolvimento eficiente de habilidades de O&M. A visão desempenha um importante papel captando a maior parte das informações sobre o ambiente [37]. As

[1] http://www.afb.org/
[2] http://aerbvi.org/
[3] https://nationaldb.org/
[4] http://www.ucergs.org.br/
[5] http://www.escoladecegositu.com.br/cursos_atividade.asp#orientacao

pessoas com deficiência visual conseguem captar as informações sobre o ambiente (estrutura geral, componentes espaciais, marcos, dimensões e posições relativas) por meio de outros sentidos como o tato, o olfato e a audição. Tendo posse do mapa mental do ambiente é possível realizar com facilidade a orientação e mobilidade em um espaço [38].

As pessoas com DV precisam ter conhecimento sobre conceitos espaciais para a criação de mapas mentais e melhor O&M, de acordo com Massi [46]. Normalmente, as pessoas com DV têm dificuldade em entender o mundo a sua volta e é importante realizar atividades que facilitem sua compreensão e interiorização desses conceitos.

Machado et al. [44] descrevem que, junto às crianças, são realizados jogos do tipo "faz-de-conta" para desenvolver os conceitos de objetos e espaço. Assim, supondo que a brincadeira seja de "casinha", no qual a criança precisa assumir o papel de dona de casa e de mãe/pai, existe uma boneca e itens que compõem a casa. Interagindo com esses objetos, a criança cega explora e compreende o cenário, e, assim, relaciona esse e todos os objetos, com seu próprio corpo. Esta proposta de explorar as habilidades de O&M torna-se interessante à pesquisa apresentada nessa dissertação porque o usuário deverá comandar um robô, que interage com o cenário. Para tal, deverá se colocar no lugar do robô e, por meio de linguagem de programação, comandá-lo. Deseja-se, assim, que a criança possa relacionar o mundo do robô com o seu próprio mundo. Por isso, é importante que, como no jogo de "faz-de-conta", possam ser utilizados cenários que representem espaços reais.

Além disso, conforme Mazzaro et al. [47], o processo de orientação tem três princípios básicos: Onde estou? Para onde quero ir? Como vou chegar ao local desejado? Para que esse processo possa ocorrer, o autor descreve as seguintes fases:

- Percepção: por meio dos canais sensoriais, captar as informações presentes no meio ambiente;
- Análise: organizar os dados percebidos em graus variados de confiança, familiaridade, sensações e outros;
- Seleção: escolher os elementos mais importantes que satisfaçam as necessidades imediatas de orientação;
- Planejamento: traçar um plano de ação para chegar ao objetivo com base nas fases anteriores;
- Execução: realizar o plano de ação por meio da prática, a mobilidade propriamente dita.

Baseado nessas fases, acreditamos que, para proporcionar às pessoas cegas o desenvolvimento das habilidades de O&M, por exemplo, essas fases podem ser implementadas no desenvolvimento de um programa:

- Percepção: relacionar às estratégias utilizadas pelo usuário, para que possa capturar as informações sobre o robô e o cenário onde ele está;
- Análise: verificar se o usuário conseguiu perceber o ambiente e onde está o robô;
- Seleção: verificar os comandos que seriam necessários para o robô realizar a tarefa;
- Planejamento: elaborar o programa que reunirá os comandos para que o robô possa se deslocar;
- Execução: executar o programa e ver se o robô alcançou o objetivo previamente definido.

2.2 Robótica educacional

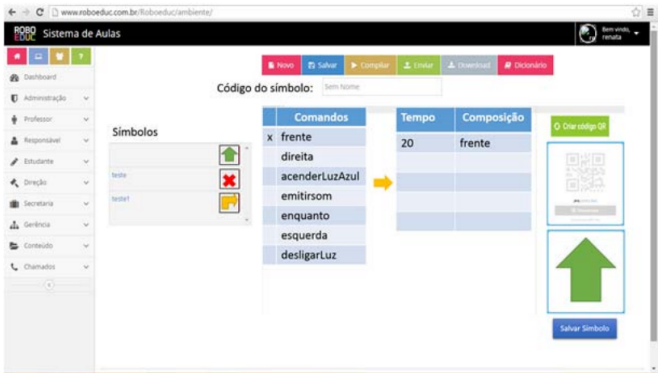
A robótica surge na educação como uma forma de tornar o aprendizado mais significativo, promovendo, por meio do seu uso pedagógico, diferentes tipos de conhecimento e competências [15]. Por exemplo, a robótica educacional pode promover, dentre outros, a pesquisa, o desenvolvimento do raciocínio lógico, o trabalho em grupos, o diálogo entre campos do saber [29]. Os primeiros avanços nessa área surgiram com Seymour Papert[6] nos anos 60 [55]. Papert desenvolveu a linguagem de programação Logo, que, por meio dos seus comandos, controlava uma tartaruga gráfica. Este ambiente de programação era de fácil assimilação e incentivava as crianças a aprenderem de forma não convencional, utilizando-se de meios lúdicos [64]. Ainda, conforme Valente [72], as características que tornam a linguagem Logo de fácil assimilação são: exploração de atividades espaciais, fácil terminologia e capacidade de criar novos termos ou procedimentos.

Várias iniciativas estão sendo desenvolvidas para inclusão de robótica nas escolas. Entre elas, pode-se citar Benitti et al. [7], que permitiram aos alunos de ensino médio realizar atividades de robótica visando aplicar conceitos relacionados à matemática, geografia e programação de computadores. Neste projeto foi utilizado o *kit* de robótica LEGO e a linguagem Logo. No trabalho de Vahldick et al. [69], usaram a robótica pedagógica para o ensino de algoritmos e programação, utilizando a IDE RoboMind que possui uma linguagem de programação simples para movimentação do robô, e o robô da LEGO, e, ainda, criaram um tablado para o deslocamento do robô. No estudo de Barros et al. [5], foi criado o CardBot 2.0 para pessoas com deficiência visual, que permite que os professores criem linguagens de programação tangíveis a partir do nível de cognição de cada turma. O CardBot possui um ambiente de criação de linguagens, linguagens tangíveis assistivas que utilizam cartões em formas geométricas e um aplicativo para dispositivos móveis (Figura 2.1a). O professor cria o alfabeto da linguagem (Figura 2.1b) que será utilizada pelos alunos. A partir

[6]Matemático e cientista pesquisador em estudos cognitivos do MIT (*Massachusetts Institute of Technology*)

disso, o aluno desenvolve um programa organizando os cartões geométricos. Esses cartões possuem marcações com as ações que o robô deve executar. Essas marcações são lidas por meio de um aplicativo de celular, visando computar a localização de cada cartão. Estes estudos demonstram que a robótica pedagógica pode ser utilizada para o ensino de diferentes áreas de conhecimento, para o trabalho aqui proposto, o foco é desenvolvimento de habilidades de O&M e de pensamento computacional.

(a) arquitetura do CardBot 2.0



(b) ambiente para criação do alfabeto da linguagem

Figura 2.1: CardBot 2.0
Fonte: Barros et al. [5]

No estudo de Cambruzzi e Souza [11], dois métodos foram aplicados para o ensino de lógica de programação: ensino tradicional e robótica educacional. No ensino tradicional foram apresentados conceitos teóricos e, em seguida, os problemas eram resolvidos com uso da linguagem de programação. Já o ensino com robótica educacional foi baseado no processo construtivista, neste caso, os alunos são os protagonistas na ação de aprendizado, envolvendo-se nas tarefas e resolução de problemas. Para isso, foi utilizado o *kit* de robótica da *Lego*. Como resultado, verificaram que utilizar metodologia mais lúdica, como a robótica educativa, colabora para o aluno desenvolver níveis de abstração. Os alunos que

utilizaram robótica educativa tiveram um desempenho melhor nas atividades e se sentiram mais motivados na execução das atividades. Este estudo mostrou que utilizar a robótica educacional incentiva os alunos durante a aprendizagem.

2.3 Pensamento computacional e ensino de programação

Para entender sobre o ensino de programação, é importante entender-se como o pensamento computacional se estabelece. Wing [73] define pensamento computacional como um conjunto de habilidades intelectuais e de raciocínio que indicam como as pessoas interagem e aprendem a pensar por meio da linguagem computacional. Para Aho [1], é o processo de pensamento envolvido na formulação de problemas, de modo que suas soluções possam ser representadas como passos de algoritmos. Por mais que o pensamento computacional esteja relacionado a compreensão de algoritmos e a aprender como pensar por meio de uma linguagem computacional, esta habilidade não se restringe apenas a cientistas da computação, é uma competência fundamental para todos [73].

Pesquisadores [9, 33, 34, 73] descrevem um conjunto de 5 principais habilidades do pensamento computacional:

- Lógica condicional: envolve a resolução de problemas através da utilização de vários modelos computacionais, por exemplo, *if-then-else*. Os alunos avaliam um problema e especificam critérios adequados para desenvolver uma abstração adequada. Nesta fase, os alunos distinguem os problemas e compreendem em um nível abstrato.
- Construção de algoritmos: construção passo-a-passo de procedimentos para resolver um problema particular e desenvolver abstrações robustas o suficiente para serem reutilizadas para resolver problemas semelhantes.
- Depuração: é analisar os problemas e erros lógicos. Os estudantes recebem *feedback* sobre os códigos criados e os analisam.
- Simulação: é a modelagem ou teste de algoritmos e da lógica do algoritmo. Nesta fase, os estudantes projetam ou executam modelos como um teste, para tomar decisões sobre quais circunstâncias consideram que sua abstração está completa.
- Socialização: refere-se à coordenação, cooperação e/ou competição durante as etapas de resolução de problemas, construção de algoritmo, depuração e simulação. É relatado que a socialização é uma característica que distingue pensamento computacional da programação de computador tradicional. Esta característica permite a troca de ideias entre os alunos, para avaliação de problemas e desenvolvimento de estratégias.

No nosso trabalho essas questões são importantes para a elaboração da metodologia de uso do ambiente de programação com robô. Inicialmente, prevemos a utilização da seguinte forma:

- Lógica condicional: os alunos identificam o problema, por exemplo, como o robô pode se deslocar para frente N passos sem bater em objetos. E verificam quais comandos podem utilizar para resolver o problema.
- Construção de algoritmos: os alunos constroem o código para resolver o problema e esse código pode ser utilizado mais tarde para problemas semelhantes.
- Depuração: os alunos recebem *feedback* sonoro. Caso o robô encontre algum obstáculo não previsto pelo aluno no código criado, esse *feedback* será dado ao aluno por áudio. Assim, será possível proceder com a correção. A ideia também é o aluno, por meio do tato, identificar o espaço.
- Simulação: os alunos podem testar os códigos criados executando-os com outras variáveis, para verificar as outras situações em que a sua solução funciona.
- Socialização: estão previstas atividades em grupos, de pessoas com deficiência visual, para resolução de problemas e desenvolvimento de estratégias.

Algumas iniciativas para o ensino de programação, a partir de meios lúdicos, vêm sendo desenvolvidas. Existem diversas linguagens de programação para fins educacionais, por exemplo, o Logo e suas variações, tais como NetLogo[7], Kturtle[8], StarLogo[9], Etoys[10] (Figura 2.2), dentre outros. Também há ferramentas de programação em blocos de comandos, tais como Scratch[11](ver Figura 2.3), Gameblox[12] (Figura 2.4), por exemplo. De forma geral, essas linguagens e ferramentas de programação apoiam uma aprendizagem mais lúdica, utilizando uma linguagem de *scripts*. Os trabalhos mostrados a seguir representam categorias de jogos que fazem uso de ambiente virtual com personagem ou robô, em que o personagem/robô virtual é guiado por comandos de programação. No trabalho aqui proposto, a linguagem *GoDonnie* é apoiada por um ambiente de programação que possui um robô virtual e um robô físico. Por este motivo, faz-se importante conhecer os ambientes virtuais que estão sendo trabalhados.

[7] http://ccl.northwestern.edu/papers/agent2004.pdf
[8] https://edu.kde.org/kturtle/
[9] http://education.mit.edu/starlogo-tng/
[10] http://www.ufrgs.br/soft-livre-edu/software-educacional-livre-na-wikipedia/etoys-linguagem-de-programacao/
[11] http://scratch.mit.edu/
[12] http://education.mit.edu/portfolio_page/gameblox/

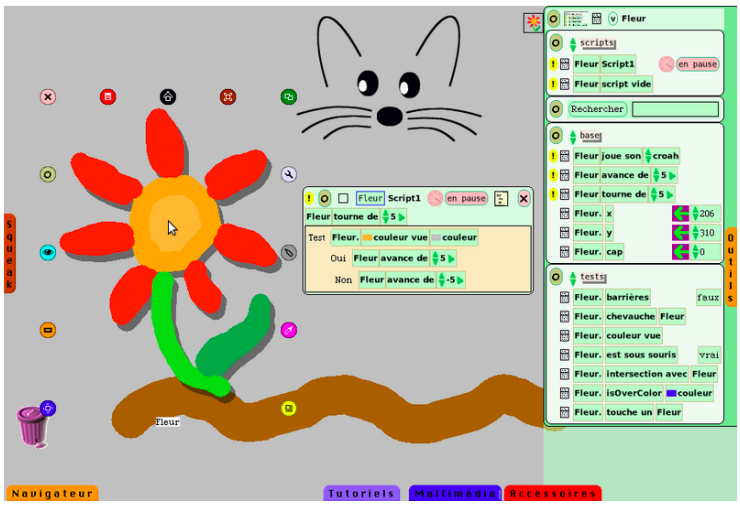


Figura 2.2: Etoys

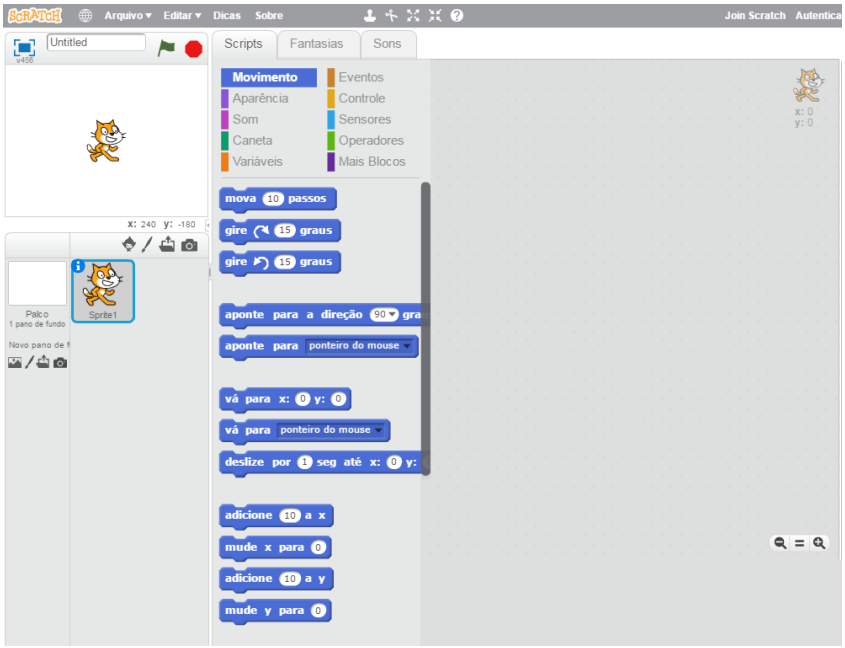


Figura 2.3: Scratch

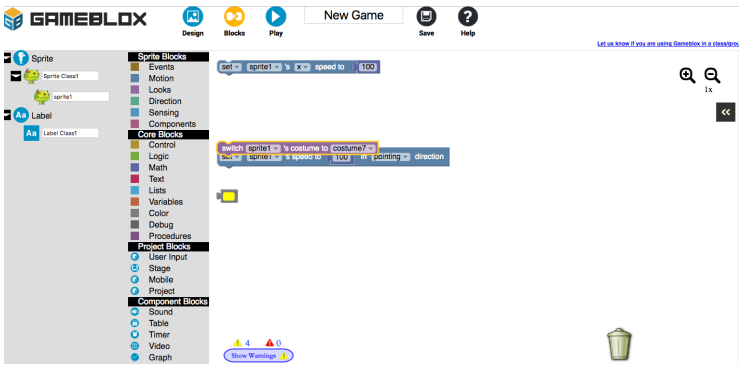


Figura 2.4: Gameblox

Meira et al. [49] apresentam competições e torneios realizados durante 5 anos (2010 à 2015), baseados no jogo educacional *Robocode*, um jogo de programação cujo objetivo é simular batalhas de robôs virtualmente. O jogador é o programador do robô virtual, cuja função é desenvolver a lógica com informações de comportamento e reação a eventos que ocorrem na arena. O jogo *Robocode* permite o ensino de linguagem de programação *Java* ou C# (plataforma .NET) (ver Figura 2.5).

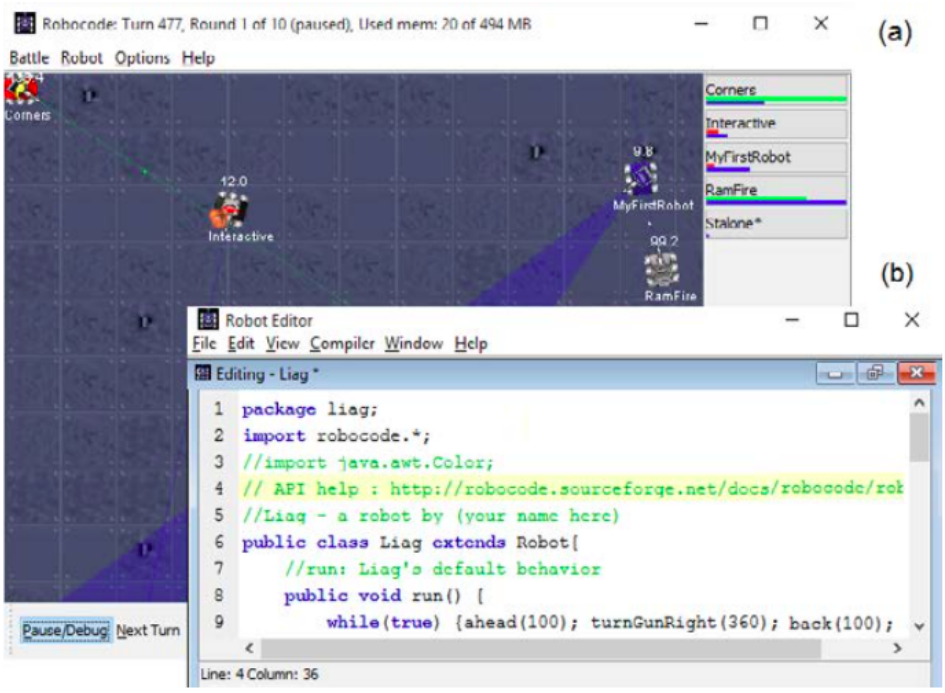


Figura 2.5: Robocode
Fonte: Meira et al. [49]

Por último, em Vahldick et al. [70] criaram um jogo casual sério para o aprendizado e prática das técnicas de pensamento computacional. O jogo chama-se *NoBug's Snack Bar*[13]. O cenário do jogo é uma lanchonete, em que o jogador controla o funcionário e o jogo controla os clientes que efetuam pedidos. As ações do funcionário são programadas em blocos (Ver Figura 2.6).

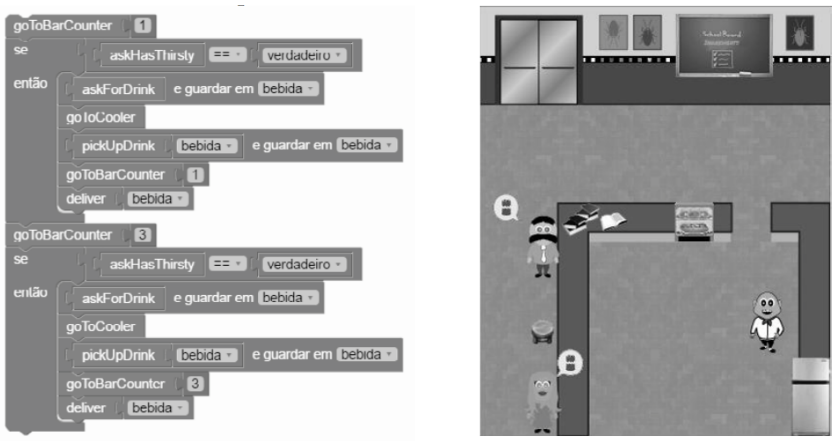


Figura 2.6: NoBug's Bar
Fonte: Vahldick et al. [70]

Tran et al. [68] propuseram uma ferramenta de programação com áudio para ajudar as pessoas com deficiência visual a aprender a linguagem C#, baseado em tecnologia *text−to−speech* (TTS). Assim, todos os elementos como o texto, recursos gráficos, localização e cor, são pronunciados por um sintetizador de fala. O ambiente não foi projetado para crianças ou adolescentes.

Jašková e Kaliaková [28] analisaram dois ambientes de programação para alunos com deficiência visual de escolas primárias. Uma vez que eles não encontraram um ambiente de programação baseado em texto, e também amigável para iniciantes em programação cegos, eles ensinaram algoritmos básicos usando um objeto tangível, um brinquedo chamado *Bee-Bot*. Eles criaram e testaram um conjunto de atividades educacionais para o desenvolvimento de competências básicas de programação. Seu principal objetivo foi desenvolver um ambiente de programação universal para ensinar habilidades de programação para todas as crianças, mas não encontraram uma maneira adequada de introduzir alguns conceitos algorítmicos, como um comando condicional e procedimento.

[13] http://nobugssnackbar.dei.uc.pt/

2.4 Usabilidade em linguagem de programação

Os autores Hoc e Xuan [25] definem programação como o processo de transformar um plano mental em um que seja compatível com o computador. A linguagem de programação é a forma como essa transformação é expressa, e quanto menor for a transformação, mais fácil será a tarefa de programar [24]. Neste sentido, a usabilidade é um recurso importante para linguagens de programação, e as avaliações com usuários podem fornecer um *feedback* valioso sobre o design da linguagem [59]. Assim, a linguagem de programação que possui uma boa usabilidade pode tornar-se mais fácil para os usuários programarem.

Pesquisadores têm adaptado os métodos convencionais de Interação Humano-Computador (IHC) para o estudo da usabilidade em linguagem de programação. Pane [54] acredita que os princípios gerais e heurísticas de IHC podem ser aplicadas ao projeto de sistema de programação [52] (Tabela 2.1). Para o autor, ao projetar e avaliar sistemas de programação, também é útil considerar os critérios de avaliação mais específicos na estrutura de Dimensões Cognitivas das Notações [23] (Tabela 2.2). Considerando esses princípios e dimensões, e fazendo uso de design centrado no usuário, foi construído o sistema de programação *HANDS*, que tem as crianças como público-alvo. Para examinar a eficácia de três características-chave de *HANDS*, foi realizado um estudo comparando o sistema com uma versão limitada que não possui as características-chave. Na versão limitada, os programadores poderiam obter os mesmos resultados que a versão completa, mas tiveram que usar técnicas de programação mais tradicionais. *HANDS* foi avaliado por 23 usuários de 9 à 11 anos. Um tutorial de 13 páginas foi usado para ensinar os conceitos básicos do sistema *HANDS* aos participantes. Após a conclusão do tutorial, os participantes receberam um conjunto de duas páginas com cinco tarefas, além de um problema bônus opcional. Os participantes trabalharam individualmente, em seu próprio ritmo. No final da sessão, os participantes preencheram um breve questionário, fornecendo informações sobre sua experiência anterior com a computação e indicando o quanto eles gostaram da atividade. Os pesquisadores tomaram nota enquanto observaram os participantes.

Tabela 2.1: Heurísticas de Nielsen

Heurística	**Descrição**
Diálogo simples e natural	As interfaces de usuário devem ser simplificadas e devem corresponder à tarefa do usuário da maneira mais natural possível, de modo que o mapeamento entre conceitos de computador e conceitos de usuário se torne direto.

Falar o idioma do usuário	A terminologia nas interfaces de usuário deve ser baseada no idioma do usuário, em vez de usar termos orientados ao sistema ou anexar significados não-padrão a palavras familiares.
Minimizar a carga de memória do usuário	O sistema deve assumir o peso da memória do usuário.
Consistência	O mesmo comando ou ação deve sempre ter o mesmo efeito.
Feedback	O sistema deve informar continuamente o usuário sobre o que está fazendo e como ele está interpretando a entrada do usuário.
Saídas claramente marcadas	O sistema deve oferecer ao usuário uma saída fácil de tantas situações quanto possível, incluindo maneiras de desfazer.
Atalhos	O sistema deve possibilitar que usuários experientes executem rapidamente operações frequentes.
Boas mensagens de erro	O sistema deve relatar os erros adequadamente em linguagem clara, evitar códigos obscuros, usar explicações precisas e não vagas ou gerais e incluir ajuda construtiva para resolver o problema.
Evitar erros	Sempre que possível, a interface do usuário deve ser estruturada para evitar situações de erro.
Ajuda e documentação	O sistema de ajuda e a documentação devem fornecer uma maneira rápida para que os usuários encontrem informações específicas da tarefa quando estão tendo um problema.

Fonte: adaptado de [54]

Tabela 2.2: Dimensões Cognitivas das Notações

Dimensão	**Descrição**
Viscosidade	O sistema não deve resistir à mudança; ele não deve exigir muitas ações do usuário para realizar um pequeno objetivo.
Visibilidade	As informações necessárias ao programador em qualquer momento devem ser visíveis ou de fácil acesso.

Compromisso prematuro	O sistema não deve forçar o usuário a fazer o trabalho em uma ordem específica ou tomar uma decisão antes que as informações necessárias estejam disponíveis.
Dependências ocultas	Links importantes entre entidades devem ser visíveis. Um sistema de programação para crianças que é projetado para usabilidade.
Expressividade do papel	A finalidade de uma entidade deve ser prontamente aparente.
Propensão a erros	A notação deve proteger contra deslizamentos e erros.
Proximidade do mapeamento	As operações do sistema devem se aproximar da maneira como os usuários pensam sobre soluções de problemas.
Notação secundária	O sistema deve permitir ao programador comunicar informações adicionais com comentários, tipografia, layout, etc.
Avaliação progressiva	O sistema deve permitir aos usuários testar programas parciais.
Difusibilidade	Pequenos objetivos não devem exigir soluções extraordinariamente longas ou grandes quantidades de espaço na tela.
Provisão	O sistema deve permitir ao usuário esboçar partes incertas de sua solução.
Operações mentais difíceis	Nenhuma das operações do sistema deve exigir grande esforço mental para usar.
Consistência	Notações semelhantes devem significar coisas semelhantes, e vice-versa.
Escala de abstração	O sistema deve fornecer uma maneira de definir novas instalações ou termos que permitam ao usuário expressar ideias de forma mais clara ou sucinta, mas não deve forçar os usuários a usarem esse recurso desde o início.

Fonte: adaptado de [54]

Sadowski e Kurniawan [59] adaptaram as 10 heurísticas de Nielsen e as 13 Dimensões Cognitivas de Notações [23] para criar uma seleção de heurísticas para avaliar as características de linguagem de programação. As heurísticas que não faziam sentido no contexto foram atualizadas, mescladas ou excluídas, restando 11. Estas podem ser verificadas na Tabela 2.3.

Tabela 2.3: 11 heurísticas para avaliar linguagens de programação

Id	Heurística	Descrição
1	Escala de abstração	Esse recurso cobre adequadamente a escala entre os níveis mínimo e máximo de abstração em um programa? (Por exemplo, a escala de detalhes de código de baixo nível para detalhes de nível de interface)
2	Consistência	Esse recurso tem um significado consistente?
3	Prontidão de erro	A notação do recurso de linguagem induz a "erros sem cuidado"? Existe uma correspondência entre a notação do sistema e como uma linguagem similar é usada no mundo real?
4	Dependências ocultas	Todas as dependências estão claras com este recurso? As mudanças locais poderiam ter efeitos globais confusos?
5	Compromisso prematuro	Usando esse recurso de linguagem, os programadores precisam tomar decisões antes de terem a informação de que precisam?
6	Avaliação progressiva	Uma aplicação parcialmente concluída deste recurso pode fornecer comentários sobre "Como estou fazendo"?
7	Viscosidade	Quanto esforço é necessário para realizar uma única alteração envolvendo esse recurso?
8	Flexibilidade e Eficiência	Este recurso é eficaz em toda a escala entre programadores novatos e especialistas?
9	Design estético	A notação do recurso é simples e concisa?
10	Recuperação de erros	Se houver um erro no uso do recurso de linguagem, as mensagens de erro são apresentadas de forma precisa, construtiva e simples?
11	Documentação	A documentação que descreve o recurso é concisa, concreta e relevante?

Fonte: Sadowski e Kurniawan [59]

Kurtev et al. [36] apresentam um método para avaliação de linguagem de programação baseado no método de avaliação de usabilidade proposto por Benyon [8] e no método de análise de dados instantâneos proposto por Kjeldskov et al. [35]. O método foi projetado para avaliar linguagens sem a necessidade de um compilador ou um *Integrated Development Environment* (IDE). Foi baseado principalmente na resolução de um conjunto de tarefas de programação com a ajuda de uma folha de exemplo, que demonstra o uso de construções relevantes a partir da linguagem que está sendo testada. Além disso, uma entrevista, escrita ou falada, serve para o propósito de aprofundar os problemas mais interessantes observados pelo pesquisador durante o processo. O procedimento do método para avaliar linguagens sem um compilador é descrito a seguir:

1. **Criar tarefas**: devem ser criadas tarefas que explorem as principais características da linguagem.

2. **Criar uma folha de exemplo**: deve ser criada uma folha com exemplos de códigos que auxiliem os participantes a realizarem as tarefas propostas.

3. **Estimar a duração da tarefa**: o pesquisador deve realizar as tarefas para ter uma ideia de quanto tempo a experiência terá por participante.

4. **Preparar a configuração**: deve ser preparado o laboratório previamente ao estudo. Além disso, recomenda-se registrar a experiência para melhor analisar o processo de resolução das tarefas.

    - (opcional) Conduzir um teste piloto: Um teste piloto pode permitir que você descubra e corrija problemas em suas tarefas, na folha de exemplo, na estimativa de tarefa e configuração, antes de realizar a experiência com todos os participantes. No entanto, requer um participante adicional e tempo para aplicação do teste.

5. **Reunir participantes**: orienta-se a utilização de 5 participantes no estudo. Ao se utilizar menos participantes, pode ser que problemas existentes não sejam encontrados, mas alguns dados são ainda melhor do que nenhum.

6. **Iniciar a experiência**: deve ser informado aos participantes que é a linguagem que está sendo testada e não eles.

7. **Mantenha os participantes falando**: o participante deve ser orientado a falar sobre como estão pensando em resolver a tarefa. O avaliador também pode responder a dúvidas sobre a linguagem e discutir a solução com o participante. Isso deve ocorrer pois, é importante entender o quão bem a linguagem permite que as soluções sejam traduzidas em código.

8. **Entrevista com o participante**: após o teste, deve ser conduzida uma breve entrevista com o participante em que você possa discutir a linguagem, as tarefas etc. Pode-se ter algumas perguntas preparadas ou criar um questionário se houver muitos participantes.

9. **Analisar dados**: após os testes, os dados devem ser usados para identificar uma lista de problemas encontrados. De acordo com Kurtev et al. [36], foram definidos os seguintes tipos de problemas:

    - **Problemas cosméticos** são erros tipográficos e pequenas palavras-chave e diferenças de caráter que podem ser facilmente corrigidos, substituindo a parte errada.
    - **Problemas sérios** são erros estruturais que normalmente afetam como o código é estruturado, mas são pequenos o suficiente para que possam ser corrigidos com algumas mudanças.

- **Problemas críticos** são mal-entendidos fundamentais de como o código de estruturas de linguagem e grandes erros estruturais que exigem uma revisão do código. Após esta categorização você terá uma lista de prioridades para melhorar a linguagem.

Farias et al. [19] avaliaram o impacto da comunicabilidade e a usabilidade do *Scratch*. Mediram o grau de comunicabilidade da ferramenta, com o objetivo de mensurar sua qualidade de interação. Para avaliar os princípios de comunicação e usabilidade da ferramenta, foi utilizado o método baseado na teoria da Engenharia Semiótica [4] o Método de Avaliação da Comunicabilidade (MAC) [57]. O MAC é um método que permite a observação de usuários em ambiente controlado. Assim, torna-se possível identificar os problemas de comunicabilidade (rupturas) a partir da recepção e entendimento da metamensagem do designer diretamente pelo usuário. Os participantes tiveram que realizar tarefas no *Scratch* e ao final de cada teste os participantes receberam um Questionário do Participante. Esse questionário possuía questões relacionadas a percepção de aprendizado, percepção de facilidade de uso e percepção de utilidade do *Scratch*.

Os estudos apresentados nortearam a construção dos procedimentos e instrumentos para as avaliações da linguagem de programação *GoDonnie*, que estão descritos nos Capítulos 6 e 8.

3. METODOLOGIA DE PESQUISA

De acordo com Fonseca [20], metodologia é o estudo da organização, dos caminhos a serem percorridos, para se realizar uma pesquisa ou um estudo, ou para se fazer ciência. Para alcançar os objetivos propostos nesta dissertação, foi realizada uma pesquisa do tipo exploratória e de natureza qualitativa. A pesquisa do tipo exploratória tem por objetivo proporcionar maior familiaridade com o problema e torná-lo mais explícito ou constituir hipóteses [22]. As abordagens qualitativas são focadas no aprofundamento de fenômenos por exploração, a partir da perspectiva dos participantes [60].

Para responder aos objetivos que norteiam essa dissertação, a pesquisa foi organizada em 4 fases conforme segue (ver Figura 3.1):

Fase 1: foi realizada uma Revisão Sistemática da Literatura (RSL) para entender como a robótica está sendo utilizada para o ensino de programação para pessoas com deficiência visual.

Fase 2: com base na RSL, no referencial teórico e na Linguagem Logo, foi elaborada a primeira versão da linguagem *GoDonnie*. A linguagem sofreu alterações posteriormente, baseada no processo de design interativo proposto por Preece et al. [58]. Após cada avaliação (Fase 3 e Fase 4), voltava-se à elaboração da linguagem. Assim, erros foram corrigidos e melhorias foram implementadas.

Fase 3: com o objetivo de avaliar a linguagem *GoDonnie* no que se refere à usabilidade, foram realizadas 4 etapas de avaliação que contou com perfis de usuários diferenciados (pessoas com DV, pessoas sem DV e professores). Com os resultados obtidos com a avaliação com cada perfil, retornava-se a Fase 2 para acréscimo de melhorias na linguagem e correções de *bugs* no sistema.

Fase 4: para avaliar o ambiente de programação *GoDonnie*, verificar se esse pode ser utilizado e como pode ser utilizado para o apoio ao desenvolvimento de habilidades de orientação e mobilidade, foram realizadas 2 etapas de avaliação com usuários com deficiência visual. Com os resultados obtidos na avaliação com cada perfil, retornava-se a Fase 2 para acréscimo de melhorias a linguagem e correções de *bugs* no sistema.

Para coleta dos dados qualitativos nas avaliações, foram realizadas observação de uso, aplicação de questionários e entrevistas semiestruturadas. Para auxílio à análise de dados, houve gravação de áudio, vídeo, fotos e de interação na tela do ambiente.

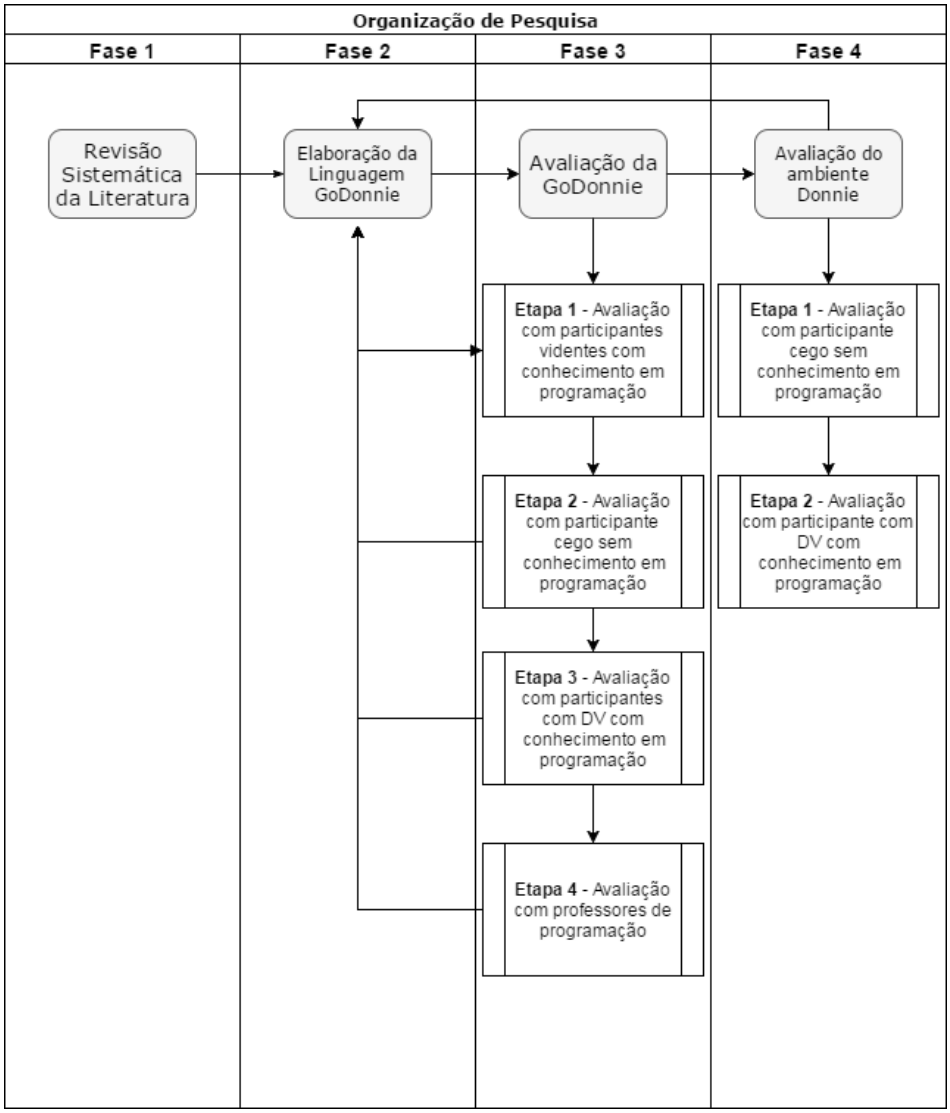


Figura 3.1: Organização da pesquisa
Fonte: a autora

Realizar pesquisas com muitos participantes é particularmente desafiante quando envolve pessoas com deficiência. Estudos relatam que testes com usuários deficientes visuais geralmente envolvem um grupo pequeno de usuários [14, 50, 52, 66, 67]. Além disso, deve-se considerar alguns fatores como: dificuldades de locomoção, disponibilidade e qualificação para os testes [50], dependência de agenda e disponibilidade de outras pessoas, quando requerem ajuda de terceiros para realizar as suas atividades, entre outras. De acordo com Lazar et al. [39], pode-se realizar estudos com poucos usuários (2-3) focando mais em observação de uso.

4. REVISÃO SISTEMÁTICA DA LITERATURA

Conforme Kitchenham [10], uma Revisão Sistemática da Literatura é um processo metódico para identificar, avaliar e interpretar as evidências científicas disponíveis e relevantes sobre um tema específico de interesse. O protocolo utilizado neste trabalho foi baseado em Kitchenham [10] e pode ser consultado em Oliveira et al. [53]. A RSL foi executada em maio de 2016.

Para este trabalho, a revisão sistemática teve como finalidade auxiliar no entendimento de como a robótica está sendo utilizada como apoio ao ensino de programação para pessoas com deficiência visual. A seguir, temos o resultado da RSL.

4.1 Resultado da RSL

A partir da aplicação de strings de busca em bases de dados, foram retornados 125 artigos. A saber, as strings continham termos relacionadas com deficiência visual, programação e robótica. No que se refere às bases de busca, foram utilizadas: ACM Digital Library, *ScienceDirect*, *IEEExplore* e *Scopus*. Após a aplicação dos critérios de seleção, restaram 9 artigos para leitura na íntegra. A Tabela 4.1 informa a quantidade de artigos encontrados, duplicados e aceitos em cada biblioteca digital.

Tabela 4.1: RSL: artigos retornados na busca

Base	**Número de estudos retornados**	**Número de estudos duplicados**	**Número de artigos selecionados**
Scopus	66	3	8
ACM	19	4	0
IEEExplore	7	5	1
ScienceDirected	33	0	0
Total	**125**	**12**	**9**

Fonte: a autora

A Tabela 4.2 apresenta os artigos selecionados para leitura, o ano e a conferência em que foram publicados. Foram identificados artigos entre os anos de 2008 a 2015.

Tabela 4.2: RSL: artigos selecionados

Referência	Ano	Conferência
[42]	2008	SIGCSE'08 - Proceedings of the 39th ACM Technical Symposium on Computer Science Education
[40]	2010	ASSETS'10 - Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility
[41]	2011	ACM Transactions on Computing Education
[26]	2012	IEEE Transactions on Learning Technologies
[30]	2013	TAAI 2013- Conference on Technologies and Applications of Artificial Intelligence
[56]	2013	ASEE Annual Conference and Exposition,Conference Proceedings
[31]	2014	IEEE SSCI 2014: IEEE Symposium Series on Computational Intelligence - RiiSS 2014: IEEE Symposium on Robotic Intelligence in Informationally Structured Space, Proceedings
[43]	2014	ASSETS14 - Proceedings of the 16th International ACM SIGACCESS Conference on Computers and Accessibility
[51]	2015	IEEE International Workshop on Robot and Human Interactive Communication

Fonte: a autora

A partir dos artigos selecionados, foram respondidas as questões que seguem. Para facilitar a leitura, ao apresentá-las, já serão discutidos resultados encontrados nos trabalhos.

1- Quais procedimentos metodológicos estão sendo utilizados no ensino de programação com robô para pessoas que com deficiência visual?

A seguir, as respostas para as questões secundárias de pesquisa que respondem a questão primária (1):

a. Quais metodologias de ensino estão sendo utilizadas para ensinar programação de robôs para pessoas com deficiência visual?

Dos estudos selecionados, 6 (seis) apresentam como metodologias de ensino o uso de oficinas de robótica [26, 31, 41, 42, 43, 56]. Apesar dos artigos [40, 51] apresentarem ambientes de robótica e programação para pessoas com deficiência visual, as avaliações apresentadas incluíram somente atividades com pessoas videntes. No estudo [30], os usuários que eram videntes tiveram seus olhos vendados. A relação entre quantidade de alunos, nível de escolaridade e conhecimento em programação dos trabalhos [26, 31, 43, 41, 42, 56], que tiveram a participação de pessoas com deficiência visual, estão ilustrados

na Tabela 4.3. A duração das oficinas variou entre os trabalhos, assim como os recursos e quantidade de participantes, conforme apresentado na Tabela 4.4.

Tabela 4.3: RSL: perfil dos alunos

Referência	**Nível de escolaridade**	**Quantidade de usuários**	**Conhecimento em programação**
[42]	Não informado	14	Não informado
[41]	Não informado	14	Não informado
[26]	Ensino fundamental	9	Não informado
[56]	Ensino fundamental e Ensino Médio	32	Alguns tinham experiência em programação
[31]	Ensino fundamental e Ensino Médio	7	Não tinham experiência
[43]	Não informado	10	3 não tinham experiência e 7 tinham alguma experiência

Fonte: a autora

Tabela 4.4: RSL: quantidade de oficinas Vs quantidade de tempo Vs organização

Referência	**Quantidade de Oficinas**	**Quantidade de Tempo**	**Organização**
[42]	1	4 dias	Equipes
[41]	3	-	Equipes (3 alunos)
[26]	-	2 semanas	Individual
[56]	-	4 horas	Equipes
[31]	-	90 min	Individual
[43]	1	4 dias (com duração de 3 a 4 horas por dia)	Equipes (2-3 alunos)

Fonte: a autora

No estudo de Kakehashi et al. [30], realizado com pessoas videntes vendadas, foram realizados dois experimentos: um para investigar a eficácia da informação tátil fornecida sobre os blocos de programação feitos de cedro japonês e equipados com leitor de *RFID*; e outro para verificar se o P-CUBE era mais fácil de programar do que utilizando a programação em texto com o software de *PC Beauto Builder2* (*software* japonês). O primeiro experimento foi realizado com 16 participantes com idades entre 20 e 38 anos, em

que todos estavam vendados. Nessa experiência foi registrado o tempo necessário para cada usuário distinguir os blocos concretos e a taxa de precisão no acerto. No segundo experimento, 10 dos usuários tinham visão normal, e não foi citado no trabalho se tinham conhecimento em programação e se estavam vendados. Nessa segunda experiência foi medido o tempo necessário para completar a tarefa em cada ambiente de programação. Esses experimentos não foram realizados com pessoa com deficiência visual, mas demonstraram que os blocos são fáceis de serem reconhecidos, assim como o P-CUBE mostrou ser um recurso que facilitou o entendimento de como programar.

Alguns trabalhos [26, 41, 42, 56] citaram o uso de tutoriais para apresentar conceitos de programação e a sintaxe da linguagem, utilizada nos respectivos estudos. No geral, esses tutoriais também continham a descrição das atividades a serem realizadas. As tarefas foram projetadas de tal forma que cada tarefa se baseava no conhecimento alcançado a partir das tarefas anteriores, aumentando sistematicamente o nível de dificuldade e, consequentemente, de habilidades necessárias para conclusão [26]. Ainda, durante as tarefas, o instrutor fazia perguntas que requeriam ao aluno dar uma resposta verbal [26]. Segundo Demo [16], a técnica de verbalização permite que o instrutor possa verificar o que e como o aluno está pensando em resolver um problema e, assim, verificar a fundamentação técnica que ocorre por meio de uma formulação que é própria de cada aluno. No trabalho de Kakehashi et al. [31], não é citado explicitamente o uso de tutoriais, mas descrevem a realização de uma aula experimental de programação na qual os participantes com deficiência visual tinham que resolver determinados exercícios, que incluíram a resolução com programa sequencial e, após, com programa com estruturas de condição e repetição.

Os trabalhos de [26, 41, 42, 56] descreveram atividades dos tutoriais fornecidos aos alunos, que estão sintetizados na Tabela 4.5. Pode-se verificar que há uma similaridade nas atividades. As atividades executam tarefas simples de movimentação do robô.

Tabela 4.5: RSL: atividades fornecidas aos alunos

Referência	**Atividades**
[42]	1. O robô deve se mover para frente por 1 segundo e parar. 2. O robô deve se mover para frente por 4 segundos, virar, e reproduz um som. 3. O robô deve se mover para frente até que toque em algo, e virar à direita. 4. O robô repete os seguintes passos 3 vezes: Avançar até tocar alguma coisa e, em seguida, vira à direita 90 graus. 5. O robô deve se mover para frente em direção a uma fonte de som, toca a fonte de som e para. 6. Enquanto não há som, o robô deve se mover para frente em direção a fonte de som, tocar fonte de som e depois girar para esquerda 90 graus. 7. O robô deve se mover para frente em direção à parede labirinto, e girar para esquerda 90 graus quando o sensor detecta por ultrasom a parede de 25 cm. Após girar à esquerda, o robô faz um som.
[41]	1. robô se movimenta e gira. 2. robô segue a linha preta. 3. robô, em seguida, lê a amostra de cor e para quando vê a carga. 4. robô, em seguida, pega a carga. 5. robô então se vira e para. 6. robô é capaz de seguir a linha de volta para outra amostra de cor e coloca a carga para baixo e mover para cima.
[26]	1. Torne o robô ON, mova para cima uma vez, torne o robô OFF. 2. Torne o robô ON, mova para cima 80cm, torne o robô OFF. 3. Torne o robô ON, gire para esquerda 45 graus, torne o robô OFF. 4. Mova para frente 300cm, gire para esquerda 90 graus, mova para frente 100cm. 5. Mova o robô em forma de um quadrado.
[56]	1. Torne o robô ON, mova para cima uma vez, torne o robô OFF. 2. Torne o robô ON, mova para cima 80cm, torne o robô OFF. 3. Torne o robô ON, gire para esquerda 45 graus, torne o robô OFF. 4. Mova para frente 300cm, gire para esquerda 90 graus, mova para frente 100cm. 5. Mova o robô em forma de um quadrado duas vezes.

Fonte: a autora

Há trabalhos que utilizaram maquetes em diferentes formatos e com variados materiais. No estudo de Kakehashi et al. [31], os participantes com deficiência visual comandavam um robô ao longo de um percurso traçado em material feito de E.V.A (Etil vinil acetato), com uma linha preta mostrando o caminho a traçar (Figura 4.1). No estudo de Ludi e Reichlmayr [42] foi criado um labirinto para o robô se deslocar (Figura 4.2), que ficava sobre uma mesa para que os participantes pudessem tatear o caminho, o robô e compreender, por meio do tato, a ação realizada pelo robô. Em Park e Howard [56], o ambiente era modificado de acordo com a atividade (Figura 4.3).

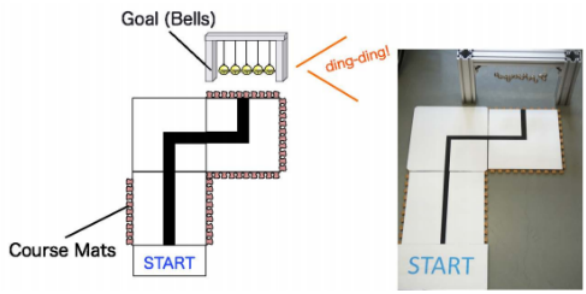


Figura 4.1: RSL: percurso com E.V.A
Fonte: Kakehashi et al. [31]

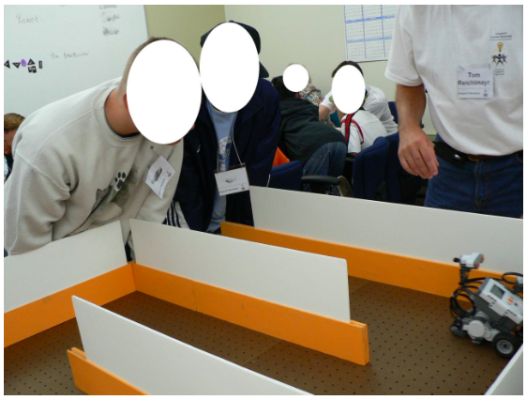


Figura 4.2: RSL: labirinto
Fonte: Ludi e Reichlmayr [42]

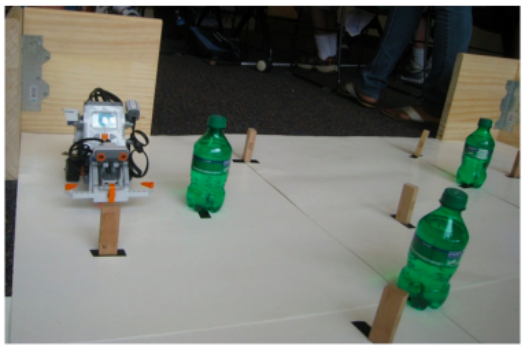


Figura 4.3: RSL: atividade de desafio
Fonte: Park e Howard [56]

b. Quais as características dos ambientes de programação que vêm sendo utilizados por pessoas com deficiência visual?

Dos nove trabalhos analisados, seis utilizam o *kit* de robótica *LEGO mindstorm NXT* [42, 40, 41, 26, 56, 43] . Os demais [30, 31, 51] criaram robôs próprios utilizando placa *Arduino*. A Figura 4.4 ilustra o robô do estudo de Ludi e Reichlmayr [41] e a Figura 4.5 ilustra o robô de Motoyoshi et al. [51].

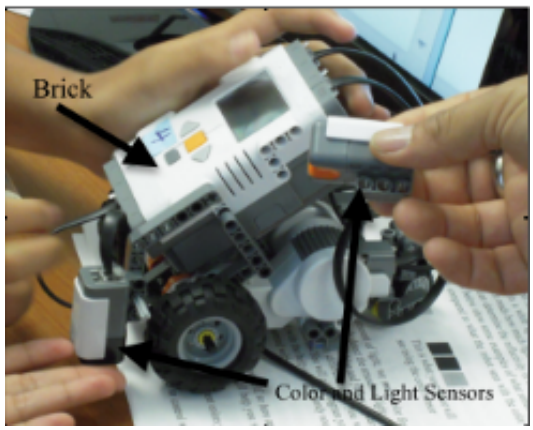


Figura 4.4: RSL: Lego mindstorm NXT
Fonte: Ludi e Reichlmayr [41]

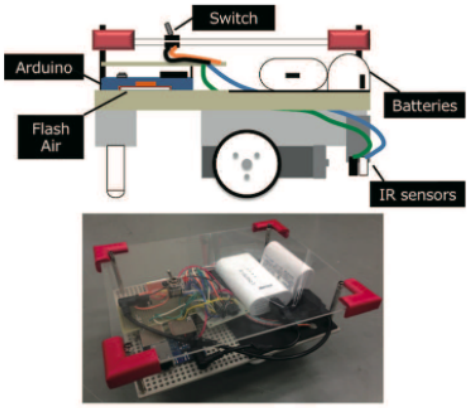


Figura 4.5: RSL: robô com Arduino
Fonte: Motoyoshi et al. [51]

Os estudos de [41, 42] utilizam o kit de robótica LEGO mindstorm NXT composto por três motores (cada um já montado em uma estrutura com caixa de redução e sensor de giro/velocidade) e sensores de toque, luz, ultrassônico e som. O ambiente de desenvolvimento foi o *BricxCC*[1] que possui código aberto. A linguagem utilizada foi o *NXC*[2], similar à linguagem C. Segundo os autores, a escolha foi motivada pela simplicidade da sintaxe, facilidade de aprender a utilizar e por ser reconhecida pelo software leitor de telas *Jaws*. Foi utilizado o sistema operacional *Windows* porque os estudantes estavam mais familiarizados.

Os estudos de [40, 43] utilizaram o *kit* de robótica *LEGO mindstorm NXT*, migrando do ambiente de programação de BrickCC para JBrick. O menu do JBrick foi simplificado para 5 opções, para minimizar o que é falado pelo leitor de telas, no artigo não foram mencionadas quais as opções; o BrickCC tinha 8 opções. Não foram citados os comandos da linguagem.

Os estudos de [26, 56] utilizaram o kit de robótica *LEGO mindstorm NXT* composto por dois motores com rodas e codificadores internos para cálculo de odometria, dois sensores de toque para detectar a entrada do usuário e incidente de colisão, um sensor de luz para detectar um marco no chão, e um sensor ultrassônico para detectar um objeto em frente do robô. Escolheram o *BricxCC* como ambiente de desenvolvimento. Os comandos para movimentar o robô podem ser vistos na Tabela 4.6. Para fornecer feedback háptico, utilizaram um controle remoto *Wii* (Wiimote). No trabalho de Howard et al. [26] foram incluídos um *feedback* resumido que foi incorporado em um agente inteligente chamado *Robbie*,

[1] http://bricxcc.sourceforge.net
[2] http://bricxcc.sourceforge.net/nbc/

que fornece *feedback* de áudio para o aluno depois que seu programa é executado. *Robbie* foi escrito na linguagem C++.

Tabela 4.6: RSL: comandos de programação

Comandos	**Descrição**
start_robot()	Liga o robô NXT
move_up(X)	Move o robô NXT para frente X quantidade de vezes
turnleft(X)	Gira o robô NXT para esquerda X quantidade de vezes
turnright(X)	Gira o robô NXT para direita X quantidade de vezes
stop_robot()	Desliga o robô NXT

Fonte: Park e Howard [56]

O robô criado por [30, 31, 51] é composto por uma placa microcontroladora *Arduino UNO*, um cartão *microSD* sem fio, uma campainha, dois motores, uma caixa de velocidades, e as baterias. Utilizaram os blocos de programação *P-CUBE* (ver Figura 4.6), conforme já descrito na questão secundária (a). Cada bloco tinha comandos associados para movimentar o robô, os comandos eram: Movimento (*forward, right, left, backward*), *Timer* (*1st, 2nd, 3rd, 4th*), *IF* (*IF START IRsensor(L) END, IF START IRsensor(R) END*) e *LOOP* (robô móvel repete os movimentos). A execução era feita da seguinte forma: a informação de tipo de bloco é obtida a partir da etiqueta *RFID* do bloco de programação, que é transmitida para o computador. Após, a informação era transmitida para o robô móvel, usando um cartão *microSD*. Assim, para que o robô executasse as funções, era necessário conectar o cartão *microSD* no mesmo. No estudo de Kakehashi et al. [31], depois de fazerem testes com participantes com deficiência visual, o robô foi remodelado para que as informações do *RFID* fossem enviadas para o robô, não sendo necessário o uso do cartão *microSD*.

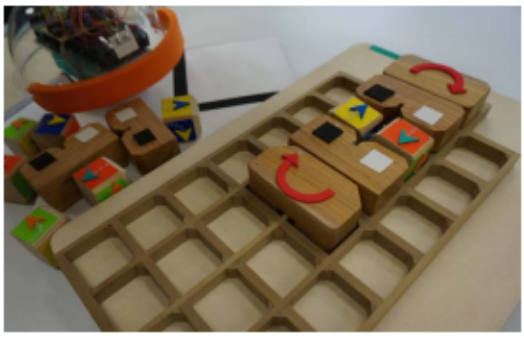


Figura 4.6: RSL: blocos de programação feitos de cedro japonês
Fonte: Motoyoshi et al. [51]

c. Quais são exemplos de boas práticas no ensino de programação de robô para pessoas que são deficientes visuais?

Foram identificadas 34 recomendações de boas práticas para o ensino de programação de robô para pessoas com deficiência visual. Essas recomendações foram agrupadas nas seguintes categorias, descritas a seguir: preparação para oficina (13), conteúdo e atividades (12), dinâmicas (4) e coleta de dados e instrumentos (5).

- **Preparação para oficina (13)**
  - Fornecer uma sessão de treinamento às pessoas que auxiliarão na oficina, para que saibam as estratégias necessárias para trabalhar com pessoas com deficiência visual [42].
  - Perguntar, previamente, como cada participante gostaria de receber o tutorial. Por exemplo, material em Braille ou com fontes ampliadas [41, 42].
  - Orientar os alunos sobre a sala, disposição dos objetos, tipo e locais dos equipamentos [42].
  - Manter espaço para circulação, por exemplo, entre as mesas, cadeiras, etc., para que os participantes com deficiência visual possam circular com mais segurança e autonomia [42].
  - Disponibilizar software do tipo leitores de tela e lupas virtuais para os participantes, que possam ser configurados por eles [42].
  - Controlar o nível de barulho do local onde são realizadas as atividades. Utilizar uma sala ampla ou várias salas menores [42].
  - Orientar que os participantes com deficiência visual façam uso de fones de ouvido [41].
  - Usar etiquetas em Braille para identificar os principais componentes do robô [42].
  - Orientar os participantes sobre as partes do robô para que se tornem familiarizados com o robô bem com o *download* de programas, etc [41].
  - Ter monitores (alunos de computação, engenharia, familiares, etc.) para acompanhar as atividades dos participantes, de forma que incentivem a participação ativa de todos nos seus grupos de trabalho [41, 42].
  - Colocar o labirinto, ou outra área para executar o robô, a uma altura que os participantes possam interagir [41].
  - Ter informações táteis para que os alunos possam identificar e reconhecer o cenário e o percurso do robô, por exemplo, utilizando materiais em E.V.A, maquetes táteis [31].
  - Disponibilizar ambientes de programação que possam ser ouvidos/vistos pelos participantes, tais como as interfaces e o código−fonte [41, 56].

- **Conteúdo e atividades (12)**
  - Fornecer tutoriais com exemplos de códigos comentados [26, 41].
  - Criar folhetos de referência rápida para os símbolos ou comandos. Isto é especialmente útil para alunos com deficiência visual que têm de aprender muitos novos comandos e sintaxes. Prever que os participantes possam escrever suas próprias notas, por exemplo, em um editor de textos no computador [41].
  - Fornecer um conjunto de comando simples (biblioteca), que possam ser construídos em etapas para que os alunos possam aprender mais facilmente codificações mais complicadas [56].
  - Colocar comentários no código, cuidando com a extensão das frases, de forma que não prejudique a leitura ao percorrer a tela [41].
  - Quebrar as linhas longas para não prejudicar as pessoas com baixa visão que precisam aumentar a tela [41].
  - Fazer atividades que possam ser realizadas de forma autônoma pelos participantes, independente de serem cegos ou com baixa visão [41].
  - Adequar as atividades à idade e diversão [41]. Pode-se realizar atividades de jogos, com música, desenho de formas geométricas, *Kick-the-can* em que o usuário programa o robô para chutar objetos pelo caminho [56].
  - Projetar atividades de forma que os participantes possam aprender com seus próprios erros e possam refazer suas estratégias de solução [43].
  - Fazer com que os tutoriais e desafios de projeto facilitem uma progressão na programação [41].
  - Apresentar informações usando meios como: meios orais e meios escritos, tanto no quadro para pessoas com baixa visão, como em apostilas, que podem ser impressas em Braille e/ou disponibilizadas no computador para pessoas cegas. Para alunos com baixa visão pode ser fornecida apostila com fontes ampliadas e também no computador [41].
  - Fornecer informações táteis e por meio de áudio [30, 51].
  - Fornecer *feedback* multimodais para que os alunos possam facilmente testar seu robô e corrigir/atualizar seus códigos [56].
- **Dinâmicas (4)**
  - Solicitar aos participantes que se revezem na programação do robô. O mesmo vale para iniciar e parar o programa no próprio robô [41].

  - Manter os participantes ativos para a realização das atividades. Nas tarefas em grupo, permitir que todos possuam funções na resolução do problema. Isso diminui o impacto de uma personalidade dominadora e evita que alunos fiquem isolados [41].
  - Permitir que os participantes sejam capazes de interagir com o ambiente de programação, ligar e desligar o robô, ativar motores e sensores, bem como executar o programa desejado [41].
  - Encorajar os participantes a relacionar com o seu próprio mundo as habilidades e desafios do robô. Por exemplo, podem relacionar os sensores com os seus próprios paradigmas de navegação [41].

- **Coleta de dados e Instrumentos (5)**

  - Aplicar pré e pós-questionários [26, 41, 42, 56]. Os exemplos de perguntas podem ser verificadas na Tabela 4.1.
  - Coletar *feedback* dos familiares dos participantes [41, 42].
  - Realizar entrevistas semiestruturadas após as atividades [43].
  - Coletar comentários dos participantes sobre a experiência na oficina [31].
  - Utilizar como medida de avaliação o número de tentativas para concluir cada tarefa [26].

Referência	**Pré-teste**	**Pós-teste**
[42]	Não citado	1. Qual seu interesse em robótica e programação. (escala de 1 (nenhum) a 5 (muito)). 2. Qual o nível de desafio das atividades. (escala de 1 (Difícil) a 3 (Fácil)). 3. O quão divertido achou a atividade. (não foi divertido, neutro, e muito divertido).
[41]	1. Tem planos de fazer um curso de computação na sua escola, se estiver disponível. (Sim, Talvez, Não) 2. Considera cursar computação na faculdade. (Sim, Talvez, Não)	1. Tem planos de fazer um curso de computação na sua escola, se estiver disponível. (Sim, Talvez, Não) 2. Considera cursar computação na faculdade. (Sim, Talvez, Não)

[26]	1. Experiência com computador. 2. Experiência em utilizar um controle de jogos como o Wiimote. 3. Experiência com programação. 4. Considera trabalhar com computadores ou robôs quando mais velho.	1. Quão útil foi a vibração do Wiimote para compreender o movimento do robô? 2. Quão útil foi o feedback de som para compreender o movimento do robô? 3. Quão útil foi determinar o que mudar em seu programa com base no que Robbie disse? 4. Qual você acha que foi mais útil para compreender o que o robô está fazendo? (Wiimote, Robbie, ambos, Nenhum) 5. Qual você acha que foi mais útil para saber como alterar o seu programa? (Wiimote, Robbie, ambos, Nenhum)
[56]	1. Quanto de experiência você tem em programação? (muito, alguma, um pouco, nenhuma) 2. Quanto você considera trabalhar com computadores ou robôs quando você formais velho? (muito, alguma, um pouco, nenhuma)	1. Quanto você acha que esta oficina ajudou a mostrar-lhe que você é capaz de trabalhar com computadores ou robótica? (muito, algum, um pouco, nenhuma) 2. Quanto tem esta oficina o encorajou a considerar trabalhar com computadores ou robótica quando crescer? (muito, algum, um pouco, nenhuma)

d. Quais foram as dificuldades/limitações no uso de robótica como apoio ao ensino de programação para pessoas que são deficientes visuais?

As limitações enfrentadas nos trabalhos [30, 51] foram relacionadas ao processo de transferência do programa, pois era necessário transferir o programa em um cartão *microSD* e conectar no robô para que ele executasse os comandos. Durante as atividades, uma pessoa vidente auxiliava na realização dessa tarefa. Os alunos comentaram que queriam executar as operações no robô com autonomia [31]. Outro problema foi as pessoas com deficiência visual conseguirem distinguir início de blocos (*IF*, *LOOP*) e finais de blocos (*IF*, *LOOP*). As diferenças dos blocos não foram suficientemente claras para os usuários [31].

Nos trabalhos [41, 42], o software *BricxCC* não era totalmente compatível com o software leitor de telas utilizado, que foi o *JAWS*. Portanto, se fazia necessário ter o auxílio de uma pessoa vidente para realizar algumas atividades, tais como auxiliar a encontrar uma determinada linha de código com base no erro do compilador.

No estudo de Howard et al. [26], houve uma incompreensão dos usuários em relação ao movimento do robô, mais especificamente, aos sinais de *feedback* que distinguem entre voltas à esquerda e à direita.

Algumas dificuldades foram relatadas por Ludi et al. [41, 43]. Houve problemas com a navegação no código construído pelos alunos e, como trabalho futuro, propuseram realizar um estudo com o uso de diferentes áudios para auxiliar na programação, tais como tons diferentes e avisos sonoros curtos. Foi observado que os participantes cegos que aceleraram a leitura do texto (leitor de telas), por vezes, se perderam no código, em termos de construção de *IF/THEN* e blocos de repetição, especialmente quando tentavam corrigir os erros. Outra dificuldade foi que alguns participantes não estavam familiarizados com o uso de pontuação, tais como chaves e colchetes, incluindo a sua localização no teclado. Como para a maioria dos participantes, a possibilidade de trabalhar com um robô foi completamente nova, isso pode ter influenciado os resultados do estudo.

Os resultados encontrados nesta RSL, buscaram responder quais procedimentos metodológicos estão sendo utilizados no ensino de programação com robô para pessoas com deficiência visual. Houve destaque para o uso de oficinas e tutoriais, uso de *kits* de robótica tais como o *LEGO mindstorm NXT*, junto com o ambiente de programação *BricxCC*. Além disso, foram identificadas recomendações de boas práticas no ensino, que foram categorizadas em: preparação para a oficina, conteúdos e atividades, dinâmicas de trabalho, e coleta de dados e instrumentos.

5. LINGUAGEM GODONNIE

A linguagem de programação *GoDonnie* foi baseada na linguagem Logo [64], que, conforme já mencionado, tem seu uso apoiado por uma metodologia de ensino e uma filosofia educacional, que propõe o uso de computadores como ferramenta no processo educacional [71].

Na criação da *GoDonnie*, a principal contribuição da Logo foi o uso mais próximo da língua natural para movimentar um objeto em um cenário. Assim, foram adaptados comandos para deslocar o robô para frente e para trás, e rotacioná-lo para direita e para esquerda. A *GoDonnie* também possui comandos de seleção, repetição, procedimento e atribuição, que são comuns a linguagens de programação de propósito geral. Entretanto, apesar de similaridades na relação de comandos disponíveis, essas linguagens não atendiam ao propósito desta pesquisa, que tem como público-alvo pessoas com deficiência visual, iniciantes em programação. Faltava-lhes uma melhor integração com software leitores de tela, comandos que evitassem o uso de teclas simultâneas, além de comandos que melhor permitissem ao usuário compreender o cenário, como o robô se relaciona com esse ambiente e se desloca nele. Ou seja, comandos que permitissem ao usuário vivenciar um ambiente para melhor compreendê-lo a partir do robô, desenvolvendo atividades de orientação e mobilidade por meio de programação.

A partir dessas limitações, foram definidos comandos específicos para o público-alvo da aplicação. O comando ESPIAR fornece informações sobre a localização e cor dos objetos no cenário, assim como a distância, em passos, entre esses objetos e o robô. O comando POS indica a posição do robô no cenário, por meio das coordenadas X e/ou Y e/ou ângulo, para o qual está direcionado o robô. O comando ESTADO traz as informações do comando POS e informa qual foi o último comando de deslocamento e rotação realizados pelo robô. O comando FALAR permite que mensagens de texto, conteúdos de variáveis ou informações relacionadas ao robô e ao cenário possam ser expressadas por meio sonoro. O comando DISTÂNCIA informa a quantidade de passos entre o robô e algum objeto que esteja na sua frente, na lateral direita, na lateral esquerda e atrás. Desta forma, permite que o usuário consulte se há alguma possível colisão. O comando COR informa a quantidade de objetos de uma determinada cor. A saber, a característica de cor foi a opção para identificar diferentes objetos no cenário, ampliando a variação de atividades que podem ser realizadas no cenário.

Também foram realizadas alterações na sintaxe de comandos existentes no Logo. Por exemplo, no *LOGOWritter*, no *SuperLogo*, no *NetLogo* e no *Lego-LOGO*, o comando de seleção SE e o de repetição REPITA possuem colchetes para delimitar início e fim de blocos. Cabe ressaltar que os símbolos { e } são lidos nos leitores de tela como “abre chaves” e “fecha chaves”, não sendo explicitada a relação entre o bloco de comandos e

o comando a que se refere. No caso do comando SE, o delimitador de início e de fim são, respectivamente, as palavras ENTÃO e FIM SE. Para o comando REPITA, optou-se por incluir a palavra VEZES, que, ao indicar a quantidade de vezes que um bloco deve ser repetido, já indica que há um início de repetição, e FIM REPITA para indicar o término desse bloco. O estudo de Stefik e Siebert [65] indicou também o uso do termo "REPEAT x TIMES" como preferência de não programadores. No caso da *GoDonnie*, essa escolha também torna o comando mais intuitivo quando são utilizados leitores de tela, evitando o uso de símbolos que requeiram o uso de mais de uma tecla, no caso, a tecla *shift*.

O comando de procedimento no *SuperLogo* e no *NetLogo* utiliza a palavra reservada FIM e END, respectivamente, para delimitar o fim do procedimento. Na *GoDonnie* utiliza-se a palavra FIM juntamente com o comando ao qual se refere, no caso, "FIM APRENDER", mantendo o padrão já citado para os comandos SE e REPITA. Ainda, a chamada do comando de procedimento no *SuperLogo* e no *NetLogo* utiliza parênteses que exige o uso de teclas simultâneas. Na *GoDonnie*, a intenção foi não usar símbolos para separar o nome de procedimentos e seus parâmetros, ou utilizá-los sem teclas simultâneas. Nesta versão da *GoDonnie*, por restrições de implementação, utiliza-se a segunda opção, fazendo uso de colchetes. O Apêndice A apresenta o manual da *GoDonnie*, contendo a descrição dos comandos com exemplos de uso.

Os estudos [3, 65] destacam erros de programação cometidos por não programadores. Dentre esses, está o uso errôneo de parênteses e colchetes, por exemplo, *while (a == 0]*, ou a utilização de chaves ao invés de parênteses *if {a == b}*. Ainda, o estudo de Stefik e Siebert [65] indica que instruções sem parênteses e chaves dão maior precisão ao usuário. Com base nesses estudos, a sintaxe da *GoDonnie* evita o uso de caracteres que requeiram o uso de teclas simultâneas. Entretanto, permaneceu o uso da tecla *SHIFT* para as operações matemáticas de multiplicação e adição, e aspas duplas para transmitir texto para o *Text-To-Speech* (comando FALAR).

Na *GoDonnie* optou-se por trabalhar somente com o tipo de dado inteiro. Dessa forma, não é possível que o robô caminhe 1 passo e meio, por exemplo. Essa alternativa foi adotada para que fosse mais simples para o usuário compreender o deslocamento do robô. Então, quando existe alguma situação que resulta em um dado *Float*, o sistema descarta (trunca) o que vem depois da vírgula, ficando apenas com a parte inteira do dado.

A Tabela 5.1 apresenta uma comparação entre a linguagem *GoDonnie* e variações da linguagem Logo. Já a Tabela 5.2 compara a *GoDonnie* com algumas linguagens de propósito gerais.

Por fim, a *GoDonnie* foi desenvolvida para ambiente *Linux* no qual o leitor de telas nativo é o *Orca*. A sua sintaxe está baseada no idioma Português do Brasil para o melhor uso da interface sonora por meio de leitor de telas. A *GoDonnie* não diferencia letras minúsculas de maiúsculas, assim como os comandos podem ser digitados com acentua-

ção ou sem. Além disso, os comandos de Movimentação e Rotação podem ser utilizados abreviados ou por extenso, por exemplo, pode ser utilizado PF ou para frente.

Tabela 5.1: Comparação GoDonnie com Logo

	LogoWritter	**SuperLogo**	**NetLogo**	**Lego-Logo**	**GoDonnie**
Para frente	PF	PF	FORWARD	FORWARD or FD	PF
Para trás	PT	PT	BACK	BACK or BK	PT
Girar esquerda	VE	PE	LEFT	LEFT or LF	GE
Girar direita	VD	PD	RIGHT	RIGHT or RT	GD
Pausa	–	ESPERE	–	WAIT n	ESPERAR n
Print	–	ESCREVA [Este é um exemplo]	–		FALAR "Este é um exemplo"
Repetição	REPETE n [pf 3]	REPITA n [pf 3]	REPEAT n [pf 3]	REPEAT n [pf 3]	REPITA n VEZES pf 3 FIM REPITA
Repetição	–	–	WHILE condição [;; comandos]		ENQUANTO condição FAÇA comandos FIM ENQUANTO
Seleção	–	SE somente usado dentro de procedimentos APRENDA TRIANGULO: X SE :X [PARE] FIM	IF comparação [comandos] IFELSE comparação [se cumpre] [se não cumpre]	IF comparação [se cumpre] [se não cumpre]	SE comparação ENTÃO ... FIM SE SE comparação ENTÃO se cumpre SENÃO não cumpre FIM SE
Procedimento	TO nome	APRENDA nome: val comandos FIM Chamada nome (1)	TO nome comandos END Chamada nome	Define nome [[parametros] [comandos]]	APRENDER nome: val1 FAÇA comandos FIM APRENDER Chamada nome [1]

Fonte: a autora

Tabela 5.2: Comparação da GoDonnie com Linguagens de propósito geral

	C	**C++**	**Python**	**Java**	**Visual basic**	**Basic**	**GoDonnie**

Seleção (IF)	IF (condição){
	comandos;
	}
		IF (condição){
		comandos;
		}
			IF condição:
			comandos
				IF (condição){
				comandos;
				}
					IF condição
					THEN
					comandos
					End If
						IF condição
						THEN
						GOTO
						nºlinha do
						programa desejada
							SE condição
							ENTÃO
							comandos
							FIM SE

Seleção (IF-else)	IF (condição){
	comandos1;
	} else {
	comandos2;
	}
		IF (condição){
		comandos1;
		} else {
		comandos2;
		}
			IF condição:
			comandos1
			else:
			comandos2
				IF (condição)
				{
				comandos1;
				} else {
				comandos2;
				}
					IF condição
					THEN
					comandos1
					ELSE
					comandos2
					END IF
						–	SE comparação
							ENTÃO
							se verdadeiro
							SENÃO
							se falso
							FIM SE

Repetição (FOR)	FOR
	(inicialização;
	condição de teste;
	incremento)
	{
	comandos;
	}
		FOR
		(inicialização;
		condição de teste;
		incremento)
		{
		comandos;
		}
			FOR variável
			IN
			objeto iterável:
			comandos
				FOR
				(inicialização;
				condição de teste;
				incremento)
				{
				comandos;
				}
					FOR variável =
					inicio TO fim
					Step incremento
					comandos
					Next
						FOR
						variável =
						inicio
						TO fim
						STEP incremento
							PARA
							inicialização;
							condição de teste;
							incremento ou
							decremento
							FAÇA
							comandos
							FIM PARA

Repetição (WHILE)	WHILE
	(condição){
	comandos;
	}
		WHILE
		(condição){
		comandos;
		}
			WHILE
			(condição):
			comandos
				WHILE
				(condição)
				{
				comandos;
				}
					WHILE
					(condição)
					comandos
					End while
						–	ENQUANTO
							condição
							FAÇA
							comandos
							FIM ENQUANTO

Print	printf (“Este é um exemplo ”);	cout« ” Este é um exemplo”	print ("Este é um exemplo")	System.out.println ("Este é um exemplo");	MsgBox "Este é um exemplo"	PRINT "Este é um exemplo!"	FALAR "Este é um exemplo"
Criação e atribuição de variável	TipoDaVáriavel variável; variável =valor;	TipoDaVáriavel variável; variável =valor;	variável = valor	TipoDaVáriavel variável; variável =valor;	Dim variável As TipoVariavel variável = valor	LET variável = valor	Criar variável variável= valor
Operadores aritméticos	+ soma. - subtração. * multiplicação. / divisão.	+ soma. - subtração. * multiplicação. / divisão.	+ soma. - subtração. * multiplicação. / divisão.	+ soma. - subtração. * multiplicação. / divisão.	+ soma. - subtração. * multiplicação. / divisão.	+ soma. - subtração. * multiplicação. / divisão.	+ soma. - subtração. * multiplicação. / divisão.

Fonte: a autora

6. AVALIAÇÃO DA LINGUAGEM GODONNIE

A avaliação da linguagem de programação *GoDonnie* foi conduzida pela autora desta dissertação, acompanhada de 2 alunos de iniciação científica que receberam treinamento e auxiliaram a controlar o tempo, filmar, tirar fotos e simular a execução do robô.

A avaliação da *GoDonnie* ocorreu sem o uso do ambiente de programação, foi realizada durante 5 meses e compreendeu 4 etapas. Na primeira etapa, participaram 4 usuários videntes e experientes em programação. Teve como objetivo a avaliação do manual da *GoDonnie* e dos tipos de atividades que seriam aplicados aos usuários com DV. Esse perfil foi importante para identificar problemas antes da avaliação com esses usuários. Na segunda etapa, foi realizada a avaliação com um usuário cego e sem experiência prévia com programação, para verificar se a linguagem seria fácil de aprender por programadores iniciantes, e se os materiais concretos estavam fáceis de entender. Na terceira etapa, foram realizadas avaliações com um participante cego e outro com baixa visão, ambos com experiência em programação, para verificar se programadores experientes conseguiriam utilizar a linguagem e se teriam sugestões de melhorias. Nessa etapa houve uma avaliação da linguagem *GoDonnie*, dos materiais táteis e do manual da linguagem. Na quarta etapa, participaram professores de programação, com o objetivo de verificar a usabilidade da linguagem.

Para as Etapas 1, 2 e 3, foram desenvolvidas atividades com diferentes níveis de dificuldade, de forma que os comandos da linguagem pudessem ser validados. Foi utilizado um pequeno objeto, que representava o robô, e um mapa tátil, produzido em material de *E.V.A* (Figura 6.1). A partir desses recursos, foram introduzidos conceitos de programação e de robótica aos participantes. Os recursos táteis foram adotados para que o usuário com DV pudesse acompanhar o robô, tendo a percepção da localização e movimentação do robô no cenário.

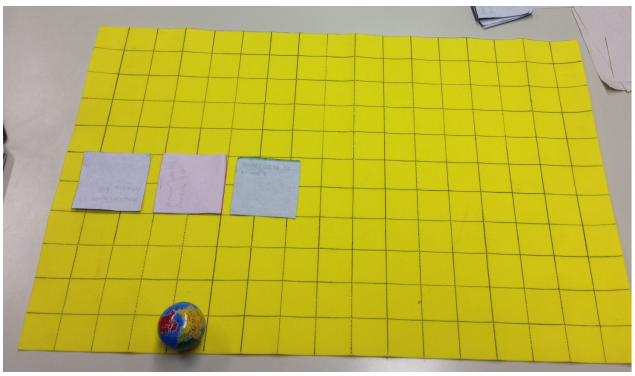


Figura 6.1: Mapa tátil
Fonte: a autora

As avaliações foram realizadas por meio de observação e entrevistas. Também foi solicitado que, durante as atividades, os participantes pensassem em voz alta para que os avaliadores pudessem entender o que estão pensando, o que estão tentando fazer, e a razão de suas ações (técnica *think-aloud*). Na Etapa 4, que foi realizada com professores, houve somente entrevista e a leitura do manual da linguagem. Os instrumentos utilizados nas avaliações, bem como o modelo de termo de consentimento, podem ser verificados nos Apêndices B, C, D, E, F.

Esta seção descreve as etapas de avaliação da linguagem de programação *GoDonnie* e as estratégias para ensiná-la.

6.1 Etapa 1 - Avaliação com participantes videntes e com experiência em programação

Essa etapa teve a participação de 4 pessoas com faixa etária entre 19 e 34 anos. Destes, 3 são estudantes de cursos de graduação e um de pós-graduação em Ciência da Computação. Dentre as linguagens de programação já utilizadas por esses participantes, destacam-se *C*, *C++*, *Python*, *Java*, *Haskell*, *HTML*, *CSS*. Desta amostra de participantes, um não tinha experiência prévia com robótica móvel e outro já tinha lecionado para pessoas que são cegas e utilizado software do tipo leitor de telas.

A avaliação com cada participante ocorreu com um intervalo de 1 semana, para permitir que os avaliadores pudessem identificar e corrigir problemas na sintaxe e na semântica da *GoDonnie*, no enunciado, no nível de dificuldade das atividades, bem como no manual de apoio ao uso da linguagem. As atividades foram realizadas sem que os usuários estivessem vendados, de forma que também pudessem validar os materiais concretos (mapa tátil e simulador do robô). Os participantes leram o manual antes da realização das atividades e puderam consultá-lo durante a resolução das mesmas. O tempo de duração que será informado para cada participante incluiu a leitura do manual, a realização das atividades e a entrevista. Havia um gabarito, que somente os avaliadores tinham posse, o qual incluía os comandos que se desejava que fossem utilizados em cada atividade. Uma vez que se poderia utilizar diferentes estratégias na resolução, os avaliadores questionaram cada participante sobre sua lógica de programação e sobre sua preferência no uso dos comandos. Essa entrevista ocorreu em diferentes momentos da avaliação.

A primeira avaliação foi realizada pelo primeiro participante e continha 9 atividades, sendo que 3 eram do tipo guiada. As demais atividades foram distribuídas em nível fácil (2), intermediário (2) e difícil (2). O tempo total para realização da avaliação foi de 01h06min. Esse participante optou por repetir comandos em sequência ao invés de utilizar comandos de repetição. Ao ser questionado, informou que considerou mais fácil fazer programas sequenciais. Em uma das atividades, esperava-se o uso de comando de seleção,

que não foi utilizado porque, segundo o participante, ele estava visualizando o percurso e podia fazer uso dos comandos de deslocamento e de giro do robô, já tendo a condição verdadeira de seleção. Os avaliadores discutiram se deveriam rever o processo e vendar os participantes. Optou-se por manter o processo, pois o objetivo era verificar e discutir se os comandos refletiam as ações esperadas para o robô. Houve sugestões relacionadas ao enunciado de algumas atividades e a retirada de uma questão de nível intermediário, porque sua resolução já estava sendo contemplada em outra. A partir desta avaliação foi gerada a segunda versão das atividades, para que o próximo participante fizesse uso dos comandos previstos no gabarito.

A segunda avaliação foi aplicada ao segundo participante que já tinha experiência com pessoas cegas. Portanto, conhecia as necessidades do público-alvo da *GoDonnie*. Ele fez uso de comandos de repetição e de seleção em algumas atividades, conforme o esperado. Nas atividades em que a resolução foi diferente da esperada, foi solicitado que a mesma fosse refeita, fazendo uso de determinados comandos. Isso fez com que a avaliação se estendesse, sendo finalizada no tempo de 2h48min. Uma das atividades do nível difícil não foi concluída e, durante a técnica *think-aloud*, foi observado que a dificuldade estava relacionada à formulação da lógica do algoritmo e não à aplicação dos comandos da *GoDonnie*. Houve diversas sugestões para o manual da *GoDonnie* no que se refere à ordenação e explicação dos comandos, bem como sugestões de reformulação das atividades. Este participante destacou que, devido à diversidade no uso dos comandos e no tempo despendido para refazer determinadas atividades, poderia haver uma sobrecarga cognitiva no usuário final, caso o procedimento fosse mantido. Desta forma, para a próxima avaliação foram definidas 5 atividades passo-a-passo, e uma para cada nível de dificuldade, totalizando 8 atividades. Também foi definido que o participante poderia somente explicar o uso de diferentes comandos, não necessitando refazer as atividades.

A terceira avaliação foi aplicada ao terceiro e ao quarto participante. O terceiro participante executou todas as atividades em 2h. Utilizou uma variedade de comandos, entretanto, pouco interagiu durante a realização das atividades. Apesar da técnica de *think-aloud* ter sido incentivada, não teve sucesso com esse participante, e os avaliadores decidiram replicar a avaliação com o quarto participante, que a realizou em 2h45min. O quarto participante utilizou a técnica de *think-aloud* e concluiu as atividades com sucesso. Os resultados da terceira avaliação confirmaram a adequação dos comandos, das atividades e do manual, não havendo propostas de alteração ou melhorias.

6.1.1 Discussão dos resultados

As avaliações com os participantes videntes e experientes em programação permitiram validar os comandos da *GoDonnie*, os tipos de atividades, a estratégia de uso do

mapa tátil e do objeto que simula o robô. O manual também pode ser revisado. Também serviram para que os avaliadores pudessem vivenciar o processo de avaliação e as formas de coleta e análise de dados, antes de realizarem com DV. Os instrumentos utilizados podem ser verificados no Apêndice C.

6.2 Etapa 2 - Avaliação com participante cego sem conhecimento em programação

Intencionalmente, foi convidado um participante (P1) que pudesse participar do ciclo de *design* da *GoDonnie*. Esse participante é do gênero masculino, possui 23 anos, é aluno do curso de Graduação em Psicologia da PUCRS e utiliza o leitor de telas *NVDA*. Não tem experiência em programação e nem em robótica. Possui alto interesse em aprender sobre robótica e programação. A Figura 6.2 apresenta P1 realizando uma atividade que simulava o uso do robô no mapa tátil.

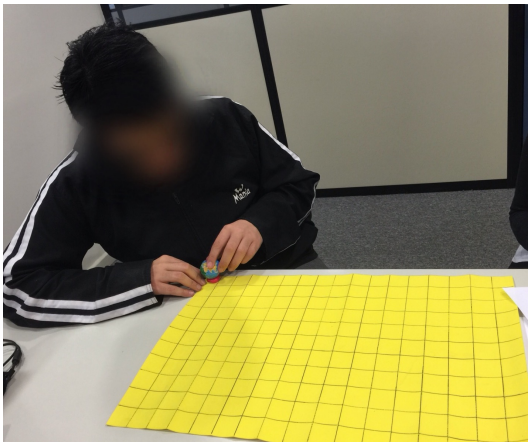


Figura 6.2: Avaliação com usuário cego sem conhecimento em programação
Fonte: a autora

A avaliação foi realizada em diferentes sessões, contendo um conjunto de atividades educativas destinadas ao desenvolvimento de competências básicas de programação e de O&M. Foi desenvolvido um instrumento para avaliar se era fácil aprender os comandos e lembrar seus usos, avaliar os materiais concretos e registrar observações (Apêndice D). As avaliações foram gravadas em vídeo, de forma que pudessem ser registrados os movimentos que P1 fazia com seu corpo para simular os giros do robô. Os vídeos também permitiram observar como o P1 movimentava o robô no mapa tátil e como compreendia o ambiente que estava representado neste mapa. Para avaliar se P1 lembrava do uso dos comandos, a cada comando era perguntado se havia sido fácil aprender e, na sessão seguinte, se esses comandos foram lembrados. No instrumento, os avaliadores sinalizavam

se a resposta se referia a sintaxe e/ou semântica dos comandos, bem como se os comandos eram lembrados com ou sem ajuda dos avaliadores.

Foram realizadas 7 avaliações, sendo 4 presenciais e 3 não presenciais. As avaliações presenciais foram realizadas em dias diferentes, e tiveram como objetivo explicar os comandos e simulá-los no mapa tátil, tendo níveis de dificuldade crescente. As avaliações não presenciais tiveram como objetivo a resolução de listas de atividades para fixação e revisão dos comandos utilizados nas avaliações presenciais, essas avaliações foram sem a utilização de mapa tátil e sem a presença dos avaliadores.

A primeira avaliação foi presencial e teve como objetivo explicar o robô, o mapa tátil e formas básicas de localiza-lo e movimenta-lo no cenário, bem como comandos de áudio e de retorno de localização do robô no mapa e dos objetos que estão ao seu redor. Incluiu explicações sobre o plano cartesiano, e a confecção e validação de um instrumento que, inicialmente, foi utilizado para conferir a rotação do robô (Figura 6.3). Esse recurso se fez necessário porque o participante estava acostumado a utilizar o mecanismo de horas ao invés de graus. Os avaliadores desempenharam o papel de *debug*, emitindo sinais sonoros, além de mensagens de sucesso e de alerta. O usuário preferiu resolver as atividades sem registrá-las por escrito. A cada comando dado pelo P1, os avaliadores deslocavam o robô no mapa tátil de forma que o participante pudesse sentir os movimentos do mesmo.

Essa sessão teve duração de 1 hora e foram trabalhados 11 comandos, considerados de fácil aprendizado pelo participante. Quatro comandos (PF, GD, GE, SOM) foram explicados somente uma vez, seis comandos (PT, CORES, DISTÂNCIA, ESPIAR, FALAR, ESPERAR) foram explicados duas vezes e um comando (POS) foi explicado três vezes. Houve sugestão de alteração no instrumento de conferência dos ângulos, bem como no mapa tátil, que deveria ter mais relevo entre as linhas e colunas. Para o instrumento de conferência de ângulos, sugeriu que tivesse uma marcação para identificar cada quadrante no círculo. A técnica *think-aloud* foi utilizada para compreender a resolução das atividades, e permitiu verificar o que o participante utilizou de abreviaturas para lembrar comandos que foram definidos por extenso (exemplo DT para DISTÂNCIA). A primeira avaliação também permitiu corrigir mensagens sonoras de prevenção e tratamento de erro. O participante sugeriu que o comando CORES pudesse ser substituído por COR, uma vez que indica a existência de objetos de determinada cor.

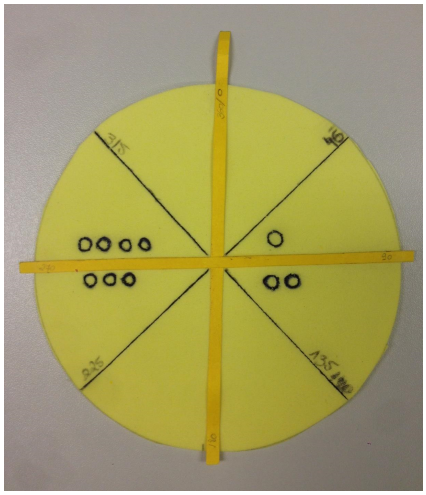


Figura 6.3: Instrumento de conferência dos ângulos
Fonte: a autora

A segunda avaliação também foi presencial e teve duração de 1 hora. Foram revisados os comandos utilizados na avaliação anterior e o P1, inicialmente, não lembrou de três comandos que estavam relacionados à rotação (GE, GD) e que haviam sido explicados somente uma vez, além do comando de pausa do robô (ESPERAR), que foi explicado duas vezes. Na segunda avaliação foram trabalhados 4 comandos e, após cada comando, o P1 executou uma atividade com o auxílio do mapa tátil. O participante considerou 3 comandos (ESTADO, HISTÓRICO, REPITA) de fácil aprendizado e um comando (SE) de aprendizado intermediário. Ainda assim, o comando ESTADO foi explicado duas vezes, o HISTÓRICO foi três vezes, o REPITA foi seis vezes e o SE foi dezesseis vezes. A necessidade de rever os conceitos se deu porque o participante solicitou variadas aplicações reais desses comandos, mostrando-se interessado e motivado em resolver mais desafios. As atividades de *loop*, relacionadas ao comando REPITA, foram simuladas na palma da mão do participante de forma que ele pudesse acompanhar o incremento da variável de controle. O P1 continuou sugerindo alterações no instrumento de conferência dos ângulos, para que as marcações tivessem mais relevo. O mapa tátil foi aprovado.

A terceira avaliação foi presencial e teve duração de 1h50min. Foram revisados os comandos da avaliação anterior e, apesar da maior complexidade desses, o P1 somente teve questionamentos com relação à sintaxe do comando REPITA e do SE, este último quando havia condições falsas. Na terceira avaliação foi trabalhado o comando de criação de variáveis (CRIAR) e outro comando de repetição (PARA), que foram considerados de fácil aprendizagem pelo participante. O comando CRIAR foi explicado três vezes. Para melhor compreender o funcionamento do comando de repetição PARA foi explicado 4 vezes. O participante executou os movimentos no espaço real, como se fosse o robô, e depois movimentou o robô no mapa tátil. O participante passou a escrever os comandos em um editor de textos enquanto utilizava a técnica de *think-aloud*. O uso de leitor de telas permitiu

validar a importância de manter acentuação nos comandos, de acordo com a escrita em Português. O P1 fez uso do mapa tátil na execução das atividades. Destaca-se que o participante se divertiu ao simular o robô com seu próprio corpo, indicando que compreendeu o uso desse comando.

A quarta avaliação foi não presencial; o participante realizou as atividades sozinho e encaminhou os exercícios resolvidos por *e-mail*. Foram realizadas 13 atividades para revisar os comandos trabalhados nas avaliações anteriores. Foram definidas 5 atividades de nível fácil, 6 de nível intermediário e 2 de nível difícil. O participante continuou utilizando um editor de textos. Não fez uso do mapa tátil, indicando que está executando as atividades em nível mental. Considerou que as atividades atenderam aos níveis de dificuldade indicados. Somente uma atividade do nível difícil não foi concluída com sucesso. O P1 se mostrou motivado na realização das atividades e demonstrou estar satisfeito com seu aprendizado na área de programação. Essa satisfação foi explicitamente informada em registro escrito e via depoimento oral.

A quinta avaliação foi não presencial; o participante realizou as atividades sozinho e encaminhou os exercícios resolvidos por *e-mail*. Foram realizadas 21 atividades: 5 de elaboração de programas (4 de nível fácil e uma de nível difícil, realizadas com sucesso), 3 para verificar correção de programas (2 de nível fácil, realizadas parcialmente certas, e uma de nível intermediário, realizada com sucesso) e 13 para corrigir sintaxes de comandos (nível intermediário, errou somente 3). P1 continuou resolvendo as questões utilizando um editor de textos, informando que não fez uso do mapa tátil. Considerou que os níveis de dificuldades estavam adequados. Continuou mostrando-se motivado em aprender programação usando a *GoDonnie* e solicitou o envio de mais listas de exercícios.

A sexta avaliação foi não presencial; o participante realizou as atividades sozinho e encaminhou os exercícios resolvidos por *e-mail*. Foram realizadas 7 atividades de elaboração de programas, sem indicação de níveis de dificuldade. Além de revisar os conteúdos anteriores, foram reforçados os operadores aritméticos e de comparação. O P1 acertou integralmente 3 atividades, resolveu parcialmente 2 atividades e não realizou com sucesso outras 2 atividades. Os erros cometidos estiveram relacionados à atribuição de variáveis, nas quais o P1 informou alguns dados de entrada por extenso ao invés de dados numéricos. Na Tabela 6.1 pode ser visto um resumo dos comandos vistos em cada avaliação à distância.

Tabela 6.1: Comandos vistos em cada sessão a distância

Sessão	**Comandos**	**Sessão**	**Comandos**	**Sessão**	**Comandos**
Quarta	PF	Quinta	PF	Sexta	CRIAR
	PT		PT		OPERADORES
	GD		SOM		FALAR "x"
	GE		DISTÂNCIA		FALAR x
	ESPERAR		POS		SE ENTÃO
	FALAR "x"		FALAR "x"		SE ENTÃO SENÃO
	REPITA		SE ENTÃO		
	SE ENTÃO		SE ENTÃO SENÃO		
	SE ENTÃO SENÃO				

Fonte: a autora

A sétima avaliação foi presencial e teve duração de 1h30min. Essa avaliação mostrou que o participante lembrou do comando CRIAR, mas não lembrou do comando de repetição PARA. Isso pode ter ocorrido porque não houve muitas atividades com o último comando. A sétima avaliação teve o objetivo de trabalhar outro comando de repetição (ENQUANTO) e o de criação de procedimentos (APRENDER). O participante considerou o comando ENQUANTO de aprendizado intermediário e APRENDER de nível fácil. O comando ENQUANTO foi explicado 6 vezes e o APRENDER 1 vez. O participante resolveu as atividades utilizando um editor de textos. Mostrou-se motivado com a possibilidade de ensinar novos comandos para o robô e ciente do aumento da complexidade a cada atividade proposta. Houve a resolução de listas extras de exercícios, que não estão sendo contabilizadas, porque se referem à revisão dos comandos e a aplicação em diferentes problemas. Ressalta-se que o participante continuou com interesse em aprender mais sobre robótica e programação. A Tabela 6.2 apresenta os comandos vistos em cada avaliação presencial e as dicas fornecidas em cada dia de avaliação.

Tabela 6.2: Comandos vistos em cada avaliação presencial e dicas fornecidas

		1ª Avaliação	**2ª Avaliação**	**3ª Avaliação**	**7ª Avaliação**
Avaliações	**Comandos**	**Dicas 1 dia**	**Dicas 2 dia**	**Dicas 3 dia**	**Dicas 4 dia**
1ª Avaliação	PF	1	0	0	0
	PT	2	0	0	0
	GD	1	0	0	0
	GE	1	0	0	0
	ESPERAR	2	1	1	1
	FALAR "x"	2	0	0	0
	SOM	1	1	1	1
	ESPIAR	2	0	0	1
	DISTÂNCIA	2	1	1	1
	POS	3	0	0	0
	CORES	2	0	0	1
2ª Avaliação	ESTADO	–	2	0	1
	HISTÓRICO	–	3	0	1
	REPITA	–	6	0	1
	SE	–	16	1	0
3ª Avaliação	CRIAR	–	–	3	0
	PARA	–	–	4	1
7ª Avaliação	ENQUANTO	–	–	–	6
	FALAR x	–	–	–	3
	APRENDER	–	–	–	1

Fonte: a autora

6.2.1 Discussão dos resultados

As avaliações permitiram verificar que a *GoDonnie* pode ser utilizada como recurso para ensinar conceitos básicos de programação para usuários cegos, iniciantes nesta área. Também ficou demonstrado a importância do mapa tátil e do robô manipulável como recursos para aprender conceitos de programação e robótica para programadores iniciantes. Nos comandos mais complexos, houve a simulação dos movimentos do robô na palma da mão do participante cego, bem como o uso do seu corpo para simular o robô no espaço. Esses procedimento facilitaram o entendimento de comandos mais complexos. O uso de editor de texto foi importante para fixar a sintaxe dos comandos e para permitir uma maior abstração em nível mental. Também permitiu verificar a entonação dos comandos pelo leitor de telas.

Sobre a sintaxes dos comandos, pode-se observar que o participante não teve dificuldades em utilizar comandos com acentuação porque já estava acostumado a escrevê-

los. Fez uso de quebra de linha para escrever comandos isolados (exemplos, CRIAR, PF, PT, GD, GE), mas escreveu comandos de repetição (REPITA, ENQUANTO) e de seleção (SE) em linha contínua. Essa liberdade de escrita da *GoDonnie*, bem como a marcação de blocos (FAÇA e FIM), foi ressaltada positivamente pelo usuário.

6.3 Etapa 3 - Avaliação com participantes deficientes visuais com conhecimento em programação

A avaliação foi realizada com 2 participantes que possuem deficiência visual, um com baixa visão (P2) e outro com cegueira adquirida (P3). O P2 tem 44 anos e o P3 tem 39 anos. Ambos do gênero masculino. Quanto à experiência em programação, o P2 possui nível intermediário de conhecimento nas linguagens *C*, *C++*, *Python*, *Java*, *Pascal*, *Delphi* e *Logo*, e o P3 possui nível básico de conhecimento nas linguagens *Java* e *C*. Apesar de ambos terem conhecimentos em programação, não possuem o hábito de programar. Além disso, os participantes também não possuem experiência com robótica.

Foram realizadas 2 avaliações com o P2, uma presencial e outra não presencial. Com o P3, foi realizada somente a avaliação presencial, pois não houve retorno da avaliação a distância. O processo de avaliação presencial foi o mesmo adotado ao P2 e P3. Foram explicados os comandos da *GoDonnie*, fazendo uso do mapa tátil e do simulador do robô. Os participantes realizaram 5 atividades guiadas (passo-a-passo) de programação, as mesmas já validadas na Etapa 1 com os participantes videntes. Os participantes fizeram uso do mapa tátil durante as atividades de programação. As respostas foram emitidas de forma sonora e registradas por escrito pelos avaliadores. Houve observação de uso durante a realização dessas atividades. Após, foi realizada uma entrevista estruturada sobre a linguagem e as atividades com os participantes. Por último, eles receberam uma lista de atividades de programação para realizarem, juntamente com o manual da *GoDonnie*, para realizar de forma não presencial. A lista de atividades conteve 19 questões de programação organizadas em categorias. Após, em cada categoria havia questões relacionadas à avaliação de usabilidade e de satisfação de uso. Estas atividades correspondem a avaliação não presencial. O Apêndice E apresenta os instrumentos de coleta de dados e as atividades que foram executadas durante a avaliação presencial e a distância.

As atividades relacionadas a avaliação presencial foram realizadas com sucesso por P2 e P3. Foi observado que P2 lembrou com facilidade da sintaxe dos comandos da *GoDonnie*, mas teve um pouco de dificuldade com a lógica, por exemplo, ao comparar valores. Já o P3 não teve dificuldades na formulação da lógica, mas, por vezes, esqueceu a sintaxe dos comandos da *GoDonnie*. Nesse último caso, os avaliadores forneceram dicas sobre os comandos para que P3 decidisse qual deveria ser utilizado e como. Estas dificuldades também foram relatadas pelos participantes. Após a realização das atividades

presenciais, houve uma entrevista estruturada para avaliar a compreensão dos comandos e do enunciado das atividades. Os participantes consideraram a *GoDonnie* fácil de entender, e destacaram positivamente o uso do idioma Português. Não sentiram falta de outros comandos. Também consideraram as atividades adequadas. Não tiveram sugestões nessa etapa para as atividades e para o manual da linguagem.

A avaliação não presencial teve por objetivo a revisão e fixação dos comandos de forma que pudessem ser resolvidos sem a presença dos avaliadores. Houve a disponibilização do manual, e as atividades faziam referência às partes em que os comandos envolvidos estavam descritos. Essa estratégia foi adotada para que o participante consultasse o manual. O instrumento estava organizado em exercícios e questões de avaliação sobre facilidade de aprender, facilidade de usar e adequação do comando a sua sintaxe e ação. A Tabela 6.3 apresenta a relação entre as categorias e os comandos. A maioria dos comandos foi considerada fácil de aprender e de utilizar, bem como adequada à ação realizada. Destaca-se que o comando COR foi considerado parcialmente fácil de aprender porque, inicialmente, o P2 não havia utilizado os parâmetros desse comando. Após, considerou este comando de fácil uso e adequado. Os operadores matemáticos foram considerados não adequados, apesar de terem sido considerados fáceis de aprender e de usar. A sugestão do P2 foi de que se pudesse incluir precedência entre esses operadores. Outra questão salientada pelo P3 é que a marcação de blocos de procedimento poderia ser sinalizada com os termos INICIO e FIM, não havendo problemas se fossem mantidos os termos FAÇA e FIM, que já eram utilizados para sinalizar blocos de comandos. A saber, nesta etapa da avaliação, o comando de criação de procedimento (APRENDER) estava definido com INICIO e FIM, e os demais comandos (PARA, ENQUANTO, REPITA) já utilizavam os termos FAÇA e FIM. Atualmente, a sintaxe da *GoDonnie* padronizou a sintaxe de bloco de comandos como FAÇA e FIM.

Tabela 6.3: Relação entre categorias e comandos

Categorias	**Comandos**
Movimentação e Rotação	PF, PT, GD, GE
Variáveis e Operadores	CRIAR, (+) Adição, $(-)$ subtração, $(*)$ multiplicação, $(/)$ divisão, (=) atribuição.
Áudio	FALAR, SOM
Informação/Percepção do ambiente	COR, DISTÂNCIA, POS
Condição/seleção	SE
Repetição	PARA, REPITA, ENQUANTO
Procedimento	APRENDER
Variados	ESPERAR

Fonte: a autora

6.3.1 Discussão dos resultados

As avaliações com participantes com deficiência visual e experiência em programação permitiram verificar que a *GoDonnie* pode ser utilizada como recurso para resolução de problemas na área de programação de robô. É importante ressaltar que o conhecimento prévio desses participantes mostrou que os comandos da *GoDonnie* permitem a resolução de diferentes problemas combinando diferentes soluções. A avaliação com esse perfil de participante confirmou que a linguagem manteve as características da linguagem Logo, no que se refere a facilidade de aprendizagem e de uso para uma linguagem de programação com fins educacionais.

6.4 Etapa 4 - Avaliação com professores de programação

Foram convidados 16 professores de programação da FACIN/PUCRS de forma intencional, sendo que destes, 7 professores aceitaram o convite. Na Tabela 6.4 o perfil dos professores é descrito. Os professores possuem experiência de 15 a 27 anos no ensino de programação. Além disso, 2 professores já lecionaram disciplinas de programação para pessoas com DV, e outros 2, já lecionaram outras disciplinas. Dos 7 professores, 3 possuem experiência no ensino da linguagem Logo e 2 já ensinaram programação com uso de robótica.

Foram realizadas entrevistas de forma presencial com cada professor. As entrevistas tiveram, em média, a duração de 45 min, na qual foram apresentados os comandos da *GoDonnie* e aplicado o questionário sobre a linguagem em forma de entrevista. O questionário continha questões relacionadas a Usabilidade (Consistência (2), Facilidade de uso (9), Relação entre o sistema e o mundo real (3) e Ajuda e Documentação (4)) e Facilidade de programação (4). Foi utilizada a escala de *Likert* de 5 pontos (concordo totalmente, concordo parcialmente, não concordo nem discordo, discordo parcialmente, discordo totalmente). Os professores foram orientados a escolher um item da escala e fazer comentários a respeito. Além disso, o questionário continha 2 questões sobre as decisões de design da *GoDonnie* e uma questão aberta para comentários gerais sobre a linguagem (Apêndice F). Para auxiliar a análise de dados, as entrevistas foram gravadas por meio de áudio.

As informações obtidas nas entrevistas foram analisadas qualitativamente por meio de análise de conteúdo, realizada de duas formas: uma análise vertical e, posteriormente, horizontal [18]. Na análise vertical, foi analisado o que cada professor expressou na sua entrevista, selecionando os conteúdos que se destacaram para cada questão. Na análise horizontal, foram analisados separadamente, cada questão da entrevista, considerando todos os professores entrevistados, revendo-se para cada item e os aspectos destacados por

Tabela 6.4: Perfil dos professores

Id	Faixa-etária	Quanto tempo ensina programação	Linguagens que ensina	Linguagens que já ensinou	Conhecimento em ensino de programação com uso de robótica	Já lecionou em disciplinas de programação com pessoas DV
Prof1	41-50	20 anos	Java	C	Não	Não
Prof2	41-50	15 anos	C, Java, Visual Basic	Logo	Não	Sim
Prof3	61-70	27 anos	Prolog, C++, Alice	Java, C, Basic	Não	Sim
Prof4	41-50	26 anos	C	C, Pascal, Java, Assembly, C++, Cobol	1 ano	Não (*)
Prof5	41-50	20 anos	Java	C, Logo, Visual basic, Basic	Não	Não
Prof6	41-50	16 anos	Java	Assembly, Visual basic, Basic, Quick basic, Logo	Não	Não
Prof7	41-50	17 anos	C, Java, Python	Assembly	10 anos	Não (*)

(*) Esses professores lecionaram outras disciplinas para pessoas com DV
Fonte: a autora

cada professor. A seguir, apresenta-se uma síntese do que foi obtido nas entrevistas para cada questão.

Nas questões relacionadas a Usabilidade, na categoria de Consistência, procurou-se verificar se a linguagem estava consistente no sentido de coerência entre os comandos e o que eles fazem. Na categoria de Facilidade de uso, buscou-se verificar se a linguagem era simples, fácil e concisa. Na categoria de Relação entre o sistema e o mundo real, examinou-se se a linguagem é clara, compreensível e se existe correspondência entre os comandos da *GoDonnie* com linguagens similares ou com termos em Português. Na categoria de Ajuda e Documentação, procurou-se verificar se o manual da *GoDonnie* pode ser utilizado para o aprendizado da mesma, se é de fácil leitura e se possui uma ordem lógica na apresentação dos comandos.

Na categoria de Facilidade de programação, buscou-se verificar se a *GoDonnie* é adequada para o ensino de programação e de programação de robô.

Nas questões de decisões de design procurou-se identificar a importância de acrescentar novos tipos de dados ou entradas na linguagem. E na questão de comen-

tários gerais sobre a linguagem, buscou-se identificar algum caso que durante a avaliação não tivesse sido coberto.

6.4.1 Consistência

Na questão 1.1, foi questionado: “Os comandos da *GoDonnie* possuem significados consistentes?”. A maioria dos professores concordou com a questão: concordo totalmente (Prof2, Prof3, Prof5, Prof7); concordo parcialmente (Prof4 e Prof6); discordo parcialmente (Prof1). Os professores que concordaram totalmente citaram que os comandos são claros e objetivos para o perfil de usuário, e que está de acordo com as outras linguagens de programação. O Prof1, que discordou parcialmente, em um primeiro momento afirmou que haveria sobreposição entre comandos, em especial entre os comandos POS e FALAR, pois o usuário poderia desejar armazenar a posição do robô em uma variável ou utilizar o comando POS para falar a posição do robô. Entretanto, as funções desses comandos não se sobrepõem, de forma que, se o usuário desejar ouvir a posição do robô, terá que combinar os comandos FALAR e POS. O Prof4, que concordou parcialmente, teve dúvidas nas atribuições das variáveis, que podem ocorrer no momento da criação das mesmas. A avaliadora, então, explicou novamente o funcionamento do comando CRIAR. A saber, uma variável pode ser criada e receber um valor inteiro, que é resultado de comandos como de COR, DISTÂNCIA e POS. Já o Prof6, que também concordou parcialmente, comentou que a sintaxe do comando APRENDER poderia ser melhorada, pois utiliza colchetes na chamada do procedimento criado. Além disso, comentou sobre a pertinência do comando DISTÂNCIA retornar números inteiros, já que não faria sentido retornar números *Float* devido à falta de precisão e ao fato de que os passos são dados em inteiros. Este professor também considerou que o manual pode ser de difícil compreensão tanto por usuário com deficiência visual quanto por quem não possui essa deficiência. A partir das sugestões, foram feitas alterações no manual no que se refere à sua organização e à apresentação de contexto de uso.

Seguem alguns depoimentos:

- *Tem alguns comandos que eles têm uma espécie de duplo sentido. Aquela situação tem o comando posição, tu pode perguntar ao robô posição x, posição y, posição z e isso está feito aqui para que esse resultado seja armazenado numa variável. Então tu coloca lá x = pos x. Em outras situações eu acho que poderia usar falar posição para se comunicar com o usuário. E eu acho que o usuário iria querer isso.* (Prof1)
- *Acho que está bem de acordo com as outras linguagens de programação. (...).* (Prof2)
- *São claros e objetivos para o perfil de usuário, e também não só para esse perfil deve funcionar.* (Prof5)

- *Eu não acho que uma pessoa que tenha deficiência visual ou não, consiga entender todo esse manual como ele está, ele vai precisar de alguém explicando algo antes, informando o contexto.* (Prof6)

Na questão 1.2, foi perguntado: “Há coerência entre os comandos da *GoDonnie* e o que eles fazem?”. A maioria dos professores concordou com a questão: concordo totalmente (Prof1, Prof2, Prof3, Prof4, Prof5, Prof7); concordo parcialmente (Prof6). O Prof1, que havia discordado parcialmente com relação à *GoDonnie* ter significados consistentes, e o Prof4, que havia concordado parcialmente, consideram que há coerência entre os comandos e o que eles fazem. O Prof5, apesar de concordar totalmente, teve um pouco de dúvida quanto ao comando ESTADO, por estar mais acostumado com o termo *status*. O Prof6, que concordou parcialmente com relação à *GoDonnie* ter significados consistentes, concorda parcialmente com relação à existência de coerência entre os comandos e o que eles fazem. Esse professor sugeriu que o manual fosse reorganizado de forma que houvesse separação de comandos, os que resultam em uma fala com informações e os comandos que resultam em ações sobre o ambiente ou sobre variáveis. Ele também sugeriu categorias: “aqueles que retornam a informação sobre o ambiente com instruções para o humano e não para a máquina, e aqueles que são para a máquina. Por exemplo, o HISTÓRICO e ESPIAR, não tem interação com a linguagem em si, são informações para o humano.”. Seguem alguns depoimentos:

- *Sim, está bem condizente com a operação. Não tive nenhuma dúvida ao tu me explicar o significado, só aquele do estado é que estamos acostumado com status.* (Prof5)

6.4.2 Facilidade de uso

Na questão 2.1, foi questionado: “A sintaxe da linguagem *GoDonnie* é de fácil entendimento?”. A maioria dos professores apontou que concordou com a questão: concordo totalmente (Prof1, Prof2, Prof3, Prof5); concordo parcialmente (Prof4, Prof6, Prof7). O Prof1, mesmo tendo concordado totalmente, e o Prof6, que concordou parcialmente, reforçaram que o comando APRENDER poderia não ter colchetes na chamada do comando. Ressalta-se que o Prof6 já havia comentado sobre o uso dos colchetes nesse comando. O Prof4 continuou com dúvidas com relação às possibilidades de atribuição de variáveis. Ao serem mostradas as explicações do manual, ele informa que estão de acordo, mas sugere que o texto que informa uma atribuição resultante do comando COR, também informe que a variável recebeu tantos objetos de tal cor ao invés de informar que recebeu uma determinada quantidade de cores. O Prof7, que concordou parcialmente, questionou se os comandos com abreviatura seriam fáceis de serem lembrados. Destacou, ainda, que havendo mudanças na sintaxe, seria importante haver a participação do usuário final na

escolha da definição dos comandos. Ressalta-se que os comandos com abreviatura foram inspirados no Logo, e a avaliação com usuários reais apontaram que são de fácil compreensão e memorização. De toda a forma, na *GoDonnie* pode-se escrever os comandos por extenso, conforme descrito no Capítulo 5.

Seguem alguns depoimentos:

- *A linguagem está bem reduzida. Ela não tem a parte sintática difícil de abre chaves, fecha chaves, vírgulas, parênteses. Ela foi toda simplificada para um jeito minimalista. (...) Se fosse para mostrar um lugar de mudança, (...) seria o APRENDER e passar parâmetros. O método está ok, mas na hora de chamar tem os colchetes. (...)* (Prof1)
- *O manual deve explicar melhor que retorna um valor. O comando criar estava certo, mas a parte de CRIAR d= cor verde, CRIAR c= distância f e CRIAR posx= pos x não ficou claro. (...) A variável c armazenará a quantidade de objetos de cor verde.* (Prof4)
- *Sim, é bem fácil. Não tive problema de entendimento.* (Prof5)
- *Sim, exceto naquele caso dos colchetes do aprender.* (Prof6)
- *Em função de muitas abreviaturas, eu não sei se isso não faz com que a pessoa esqueça (...). Por exemplo, o que que é PF? É para frente, mas será que ele vai memorizar isso? (...) Não sei se seria mais fácil FRENTE só e só TRÁS? E em extenso GIRAR DIREITA e GIRAR ESQUERDA?.* (Prof7)

Na questão 2.2, foi perguntado: “A semântica da linguagem *GoDonnie* é de fácil entendimento?”. A maioria dos professores concordou com a questão: concordo totalmente (Prof2, Prof3, Prof4, Prof5, Prof6, Prof7); concordo parcialmente (Prof1). O Prof1, que concordou parcialmente, indicou algumas questões relacionadas, principalmente, às variáveis. Dentre essas, questionou a necessidade de haver um comando para excluir uma variável, bem como se as variáveis seriam globais. A *GoDonnie* não possui um comando para limpar variáveis e nem para excluí-las. Além disso, as variáveis são criadas dentro de um procedimento, não são reconhecidas fora dele.

Seguem alguns depoimentos:

- *(...) tu cria uma variável Criar a. Não existe um destruir a ou esquecer a? (...) se tu está dentro do aprender, tu vai poder utilizar uma variável que está fora? (...)* (Prof1)
- *Pelos exemplos que tu me apresentasse me pareceu que sim. Acho que a pessoa não vai ter nenhum problema, claro que ela vai ter que ter conhecimento de lógica né, mas semanticamente concordo totalmente.* (Prof5)
- *Ela é bem extensa, bastante coisa, mas não necessariamente tudo o usuário precisa utilizar. Mas é de fácil compreensão.* (Prof7)

Na questão 2.3, foi questionado: “A sintaxe da *GoDonnie* possui um conjunto de comandos que permite a resolução de problemas computacionais?”. A maioria dos professores concordou com a questão: concordo totalmente (Prof1, Prof2, Prof3, Prof5); concordo parcialmente (Prof4, Prof6, Prof7). O Prof1, que concordou totalmente, fez algumas indagações sobre o uso de vetores e listas, mas considera que não são necessárias no contexto da *GoDonnie*. Já os Prof4 e Prof7, que concordaram parcialmente, citaram o fato de a *GoDonnie* trabalhar somente com o tipo inteiro de variável. Além disso, Prof4, adicionou que achou uma limitação a quantidade de cores que o robô identifica. É importante ressaltar que a *GoDonnie* trunca resultados considerando somente a parte inteira. Essa informação foi incluída no manual. Seguem alguns depoimentos:

- *Eu acho que sim. Ela tem uns comandos para o robô se mover, girar. Ela tem o comando de sensor para pegar os dados do ambiente. Ela tem a parte aritmética, a parte lógica (...). Tem coisas aqui que eu não vi, mas também eu acho desnecessárias nesse contexto, por exemplo, vetores, lista, mas, ao mesmo tempo, não me parece que precisa. Então, tu não resolve todos os problemas computacionais. Precisou de um array e um vetor tu não faz, mas o que está disposta a fazer está coerente, não me parece que esteja faltando algo. Concordo totalmente, se tu pensar que é nesse mundo do robô se mexendo.* (Prof1)

- *Como tu tens algumas limitações com a quantia de cores que identifica, quanto ao tipo de variável que são só inteiros, eu acho que tem certas limitações.* (Prof4)

- *(...) eu acho que ela potencializa o poder da linguagem LOGO. Ela tem o mesmo poder de expressão. Problemas computacionais é muito amplo.* (Prof6)

- *(...) quando tu fala em operadores e variáveis, tu só fala em variáveis inteiras, então tu pode ter operações aqui de divisão, multiplicação onde tu vai ter um erro. (...) Eu estou analisando duas coisas, o robô no espaço físico e no espaço virtual, então, esse tipo de coisa, vai que o usuário queira fazer alguma coisa andar 3 dividido por 5? Isso vai resultar 0 (zero) para ele. E daí não vai acontecer nada. Daí tem que ver qual é o feedback que tu tem em relação quando ele faz esse tipo de operação: o que retorna para ele? Porque ele pode ter pensado uma coisa e feito o cálculo errado. (...)* (Prof7)

Na questão 2.4, foi questionado: “A semântica da *GoDonnie* permite melhor utilizar o conjunto de comandos para resolução de problemas computacionais?”. A maioria dos professores concordou com a questão: concordo totalmente (Prof2, Prof3, Prof5, Prof6, Prof7); concordo parcialmente (Prof1, Prof4). O Prof4, concordou parcialmente, pelo mesmo motivo da questão anterior, citando a limitação nos tipos de variáveis e quantidade de cores, que são identificadas no ambiente. O Prof1 concordou parcialmente, porque considera que ainda há sobreposição em alguns comandos e que o usuário optará por um ou

outro conforme sua preferência de estilo de programação. Esse professor fez relato similar na questão 1.1, quando afirmou que poderia haver comandos com duplo sentido.

Seguem alguns depoimentos:

- *Tem uma dificuldade na semântica aqui, num primeiro contato (...) parecem um pouco parecidos. Espiar é uma coisa, Histórico é outra, Posição é diferente. Parece que tem um pouco de sobreposição das coisas, mas eu acredito que o usuário acostuma a usar um certo comando e deixa outros de lado. (...)* (Prof1)
- *Sim, todos eles são bem simples. Não vejo problema, concordo plenamente.* (Prof3)

Na questão 2.5, foi perguntado: "É fácil realizar as atividades com a *GoDonnie*?". Todos os participantes concordaram totalmente. O Prof1, apesar de concordar totalmente, questionou se o comando COR possui um bom nome, e questionou se o comando TEM VERDE não seria mais intuitivo que COR VERDE. Entretanto, esse comando teve aceitação dos usuários com deficiência visual, tendo sido alterada a sua sintaxe de CORES para COR (ver Seções 6.2 e 6.3). Considera-se, ainda, que TEM VERDE remeteria a uma resposta binária, sim ou não, ao invés de informar a quantidade de objetos que haveria naquela cor, que é o objetivo do comando COR. O Prof7, que também concordou totalmente, retornou à questão quanto à abreviatura de alguns comandos, conforme já discutido na questão 2.1.

Seguem alguns comentários:

- *(...) As atividades de mover, girar, ver o ambiente, procurar alguma coisa de cor vermelha, me parece que eu tenho as unidades de ação bem delimitadas. Acho que está tranquilo. Qual é o comando para procurar algo vermelho? (...) Uma coisa que eu me perguntaria é se esse comando <COR> tem um nome bom. Porque COR VERDE parece que tu está mandando fazer alguma coisa e o que tu está fazendo na verdade é uma pergunta "Tem verde?". Talvez TEM VERDE daria melhor ideia do que tu está realmente fazendo do que dizer COR VERDE. (...)* (Prof1)
- *Acho que sim, teria que ver para um problema, acho que qualquer tipo de problema poderia ser solucionado porque os comandos são básicos, né. A partir dali poderia derivar uma série de soluções.* (Prof5)
- *(...) são palavras que traduzem o que tu realmente quer fazer, o objetivo. Por isso que esse GE e GD eu usaria algo mais objetivo.(...)* (Prof7)

Na questão 2.6, foi questionado: "Os comandos da *GoDonnie* são simples?". Todos os participantes concordaram totalmente com a questão, e somente o Prof7 apresentou um questionamento relacionado aos parâmetros do comando DISTÂNCIA e como deslocar o robô até determinado objeto. A saber, o comando DISTÂNCIA combinado com os parâmetros F, FD, FE, T, TE, TD, retorna a distância para um objeto mais próximo que está à

frente, à frente e à direita, à frente e à esquerda, atrás, atrás e à direita, atrás e à esquerda do robô. Desta forma, combinado com os comandos de PF ou PT, pode-se deslocar o robô até o objeto.

Segue parte do depoimento:

- *São, mas por exemplo, a distância volta a ter tudo aquilo F, FD, FE, T, TE, TD isso aqui ele está solicitando a distância até um objeto? E se eu quiser ir até o objeto? (...) Tá, ok.* (Prof7)

Na questão 2.7, foi questionado: "Os comandos da *GoDonnie* são concisos?". A maioria dos professores concordou que os comandos são concisos: concordo totalmente: Prof1, Prof2, Prof3, Prof4 Prof5, Prof7; concordo parcialmente: Prof6. O Prof5 concorda totalmente, mas destaca que um programa pode requerer várias linhas para solucionar um problema. O Prof6, apesar de ter concordado parcialmente, informa que considera que os comandos não sejam concisos e traz sugestões ao comando SOM. O Prof7, que concordou totalmente, comentou que os comandos são fáceis, apesar dos comandos de repetição e seleção poderem trazer dificuldades. Também salientou a importância em haver delimitador de fim de bloco para que o usuário possa melhor identificar quando finaliza uma sequência de comandos. Ressalta-se que a *GoDonnie* possui esses delimitadores, conforme especificado no Capítulo 5.

Seguem alguns depoimentos:

- *Concordo totalmente, eu não sei se isso vai depender da natureza do problema, se o problema for um pouco mais complexo de ser solucionado pode ser que tu tenha que aumentar.* (Prof5)
- *<O comando> SOM, aqui não poderia ser SOM volume, e como parâmetro eu colocaria 0 para desligado e 100 para ligado. E poderia regular até com níveis de volume.* (Prof 6)
- *São, com exceção dos comandos de repetição e seleção, mas, é claro, que isso é normal.(...) E precisa ter o FIM REPITA para ele saber que acabou, onde termina a lógica (...).* (Prof7)

Na questão 2.8, foi questionado: "Os comandos da *GoDonnie* são fáceis de lembrar como usar?". A maioria dos professores concordou com a questão: concordo totalmente (Prof3, Prof4, Prof5, Prof6, Prof7); concordo parcialmente (Prof1, Prof2). O Prof5, concordou totalmente, mas ressaltou que o uso da linguagem pode requerer o uso do manual. O Prof6 e o Prof7, que concordaram totalmente, possuem dúvidas quanto à facilidade de lembrar comandos que são usados abreviados. Seguindo sugestão do Prof6, foram incluídas no manual as formas por extenso dos comandos de deslocamento e giro. O Prof1,

que concordou parcialmente, questionou alguns nomes de comandos, voltando a citar o comando COR e fazendo uma sugestão ao comando DISTÂNCIA. O Prof2, que também concordou parcialmente, citou que o comando POS compreende os eixos X e Y, mas ficou em dúvida se lembraria de como usar o argumento A para referenciar o ângulo. Com relação as dúvidas do Prof1, Prof2 e Prof7, cabe destacar que, na avaliação com usuários com deficiência visual (Etapas 2 e 3, descritas nas Seções 6.2 e 6.3), as abreviaturas dos comandos de deslocamento e de giro, o comando DISTÂNCIA e o POS foram considerados fáceis de lembrar e de aprender.

Seguem alguns depoimentos:

- *(...) DISTÂNCIA F, ele vai me dizer quando eu posso andar até ter alguma coisa na frente (...) não é uma palavra boa, mas a primeira que me vem à cabeça é LIVRE F, LIVRE E. Vai livre até algo.* (Prof1)
- *Eu só fiquei na dúvida com o comando POS. (...) eu entendo o plano eixo, mas o ângulo eu fiquei na dúvida se eu lembraria como usar isso.* (Prof2)
- *(...) o manual subtrai algumas coisas que são mais complexas dos comandos das linguagens, então concordo totalmente.* (Prof5)
- *(...) os comandos PF e PT podem ser escritos Para frente e Para trás, mas isso não está dito no manual, poderia ser informado: você pode usar PF ou Para frente. Por causa da questão do mnemônico, se você usar bastante PF e PT tu vai lembrar que é para frente e para trás, mas colocando a informação no manual será mais fácil de associar.* (Prof6)
- *(...) acredito que com o uso se torna fácil, é como linguagem de programação traduzida para o Português.* (Prof7)

Na questão 2.9, foi questionado: “Os comandos da *GoDonnie* permitem que um problema possa ser resolvido de diferentes formas?”. A maioria dos professores concordou com a questão: concordo totalmente (Prof1, Prof2, Prof3, Prof5, Prof6, Prof7); concordo parcialmente (Prof4). O Prof4, que concordou parcialmente, comentou que poderia haver um estudo para ver se a *GoDonnie* teria poder computacional comparando a uma máquina de *Turing*. Este estudo não foi realizado.

Seguem alguns depoimentos:

- *Com certeza, um problema no mundo do robô né? Porque aí é só como tu vai recompor a sequência de comandos, ai tu tem várias formas.* (Prof1)
- *Concordo plenamente, pois tem comandos alternativos.* (Prof2)
- *Sim, tu pode fazer, por exemplo, usando repetição (...)* (Prof5)
- *Sim, toda programação envolve isso.* (Prof7)

6.4.3 Relação entre o sistema e o mundo real

Na questão 3.1, foi questionado: “A linguagem *GoDonnie* é clara?”. Com exceção do Prof1, os demais professores concordaram que é totalmente clara (Prof2, Prof3, Prof4, Prof5, Prof6, Prof7). O Prof1, que concordou parcialmente, discutiu sobre a definição de alguns nomes de comandos porque há aqueles que já são comumente conhecidos e podem ter sido adaptados para o contexto da *GoDonnie*, e outros que foram criados. Desta forma, esse professor se questiona se teriam sido criados com a mesma denominação.

Segue parte do depoimento do Prof1:

- *(...) As ações são claras, mas tem que pensar melhor nos nomes. É engraçado os nomes das coisas de programação: criar, repita. Isso aí a gente usa a tanto tempo que se torna padrão. Agora, comandos que são diferentes, aí tem que dar uma repensada porque não temos histórico de usar esses comandos. O exemplo é o próprio comando HISTÓRICO, que para ti tu precisa ver o que fez até agora. Como seria um bom nome para isso? Os comandos de criar, repita, tem 50 anos de história para isso, mas comando histórico não tem referência, o que seria um nome melhor? Vão surgir várias ideias. Uma delas que pensei agora é PASSADO. É difícil ver um nome porque não temos no que se basear. Como dar nome para algo que tu nunca viu antes? (...)* (Prof1)

Na questão 3.2, foi questionado: “Os conceitos utilizados nos comandos da *GoDonnie* são compreensíveis?”. A maioria dos professores concordou com a questão: concordo totalmente (Prof1, Prof2, Prof3, Prof5, Prof7); concordo parcialmente (Prof4, Prof6). O Prof5, que concordou totalmente, questiona se a linguagem poderá atender às necessidades de uso do grupo de usuários com deficiência visual. Destaca-se que, a linguagem foi testada junto a usuários com deficiência visual, que demonstraram que conseguiram resolver problemas por meio de seus comandos (ver Seção 4.2 e 4.3). O Prof4, que concordou parcialmente, comentou que, inicialmente achou os conceitos um pouco complicados. Mas salientou que, talvez ao utilizar com mais frequência, sua avaliação poderia ser diferente. Destaca-se que esse mesmo professor considerou a linguagem clara, concisa, simples, fácil de lembrar como usar e de realizar atividades. O Prof6, que concordou parcialmente, destacou a importância de incluir um contexto de uso da *GoDonnie*, explicando seu objetivo e ambiente, por exemplo, que foram acrescentados ao manual.

Alguns alguns comentários:

- *(...) é um pouco complicado de início, ela não é 100%. Talvez, conforme eu fosse usando, talvez, eu pudesse responder de maneira diferente.* (Prof4)

- *Só não sei se os comandos atendem à tudo que é requerido para esse perfil de usuário, por exemplo, somente com os sons, falas integradas isso vai ser suficiente para a pessoa conseguir abstrair e entender?* (Prof5)

- *(...) colocar o contexto ajudaria muito.* (Prof6)

Na questão 3.3, foi questionado: "Existe correspondência entre os comandos da *GoDonnie* com linguagem similar (com a linguagem de programação ou com o Português, por exemplo)?". A maioria dos professores concordou com a questão: concordo totalmente (Prof2, Prof3, Prof5, Prof6, Prof7); concordo parcialmente (Prof4); não concordo nem discordo (Prof1). O Prof6 comparou a *GoDonnie* com a linguagem humana e, nesse sentido, concorda totalmente. O Prof4, que concordou parcialmente, e também comparou com linguagem natural, ressaltou que a *GoDonnie*, como as demais linguagens de programação, precisa seguir uma semântica e uma sintaxe, sem ambiguidades. O Prof5 comparou a *GoDonnie* com o Logo e considera que há semelhanças. O Prof1 preferiu não se posicionar, pois não tem muita experiência com linguagens similares na área de robótica. Destaca-se que o Prof7, que possui mais de 10 anos de experiência em robótica, considera que a *GoDonnie* possui correspondência com linguagens similares dessa área.

Seguem alguns depoimentos.

- *(...) devido a ser uma linguagem de programação, que tem que ter uma sintaxe bem definida, tem que ter uma semântica bem definida. Caso tu não use dessa forma, ela não vai ser ou compilada ou executada. No mundo real, tem um monte de imprecisões, então ela é parcialmente do princípio do que tem no mundo real.* (Prof4)

- *Eu identifiquei muito com a época que eu ensinei o Logo. Claro as linguagens tendem a serem mais elaboradas do ponto de vista de elementos, caracteres especiais toda a semântica dessas <linguagens> convencionais.* (Prof5)

6.4.4 Ajuda e Documentação

Na questão 4.1, foi questionado: "O manual da *GoDonnie* é adequado ao aprendizado da mesma?". A maioria dos professores concordou com a questão: concordo totalmente (Prof1, Prof2, Prof3, Prof5, Prof7); concordo parcialmente (Prof4); discordo parcialmente (Prof6). O Prof1, que concordou totalmente, salienta a importância de haver atividades práticas para complementar o aprendizado da *GoDonnie*. O Prof2 manifestou preocupação quanto à necessidade do usuário ser alfabetizado e sugeriu incluir figuras das telas. O Prof7 também havia indicado essa possibilidade, mas considerou o manual adequado, já que é direcionado a usuários com deficiência visual. O Prof4, que concordou parcialmente, comentou que está bom, mas são necessários alguns ajustes, já citados nas

questões anteriores (em especial, os relacionados à atribuição de variáveis). O Prof6, que discordou parcialmente, acrescentou que tem um espaço para melhorias. A saber, esse professor já havia indicado sugestões quanto à organização de alguns comandos, bem como que fosse incluído um contexto de uso da *GoDonnie*.

Seguem alguns depoimentos.

- *Eu não sei a minha dúvida. Estou pensando numa criança. Uma pessoa que tenha compreensão sim, mas se for uma criança não sei se há uma faixa etária, se tem que ser alfabetizado, se não, talvez tu pudesse colocar telinhas, desenho do que tu quer dizer. Mas, para crianças com deficiência visual, vai ter que passar por leitor de tela. Dentro desse cenário, sendo uma criança que irá ter um leitor de tela, eu acho que sim, concordo plenamente.* (Prof2)
- *Sim porque tu coloca os exemplos e eles são bem fáceis de serem compreendidos.* (Prof5)
- *Para pessoas que enxergam, eu ilustraria isso, para saber o que ele está tentando fazer em determinada situação. Porque, às vezes, tu lê todo o texto, e uma figura iria resumir tudo. Mas não sei para o deficiente visual, como seria. Aí já não valeria a pena para ele, para pessoa cega essa é a melhor maneira, como tu está descrevendo.* (Prof7)

Na questão 4.2, foi questionado: “Os problemas apresentados nas atividades do manual da *GoDonnie* estão adequados para apresentar exemplos de soluções?”. A maioria dos professores concordou com a questão: concordo totalmente (Prof2, Prof3, Prof7); concordo parcialmente (Prof4, Prof5, Prof6); discordo parcialmente (Prof1). Os professores Prof1, que discordou parcialmente, e Prof4 e Prof5, que concordaram parcialmente, sugeriram ter exemplos de programas mais completos e mais complexos no manual. O Prof6 sugeriu incluir um cenário de uso e os exemplos fazerem referência a esse.

O manual traz exemplos de acordo com o comando que está sendo explicado. Há comandos que requerem o uso de outros, mas no manual, esses estão explicitamente definidos. Desta forma, optou-se por incluir uma seção de desafios, que contém atividades extras, que requerem a resolução de problemas mais complexos.

Seguem alguns comentários.

- *Esse ponto eu discordo, porque os comandos que estão dados aqui, eles me ensinam o que acontece: se eu disser vai para frente, está dito você vai para frente. Mas isso não é propriamente resolver um problema. Se eu disser vai atrás do objeto verde, isso eu não sei exatamente fazer. O que tu está me dando é detalhes das ações, mas resolver um problema um pouco mais complexo, como ir até o objeto vermelho mais próximo, isso é bem mais complicado. De repente, dá para fazer um capítulo*

fora, porque não é ligado a um comando, um exemplo mais prático, como eu acho o objeto vermelho mais perto. Vamos começar com um comando assim para descobrir tal coisa, depois vou me virar na direção dele. Aí tu está me ensinando a compor o simples para chegar em algo mais complexo. E isso eu não vi no manual. O manual explica coisas pequenas. (Prof1)

- *(...) eu faria um cenário, e os comandos sempre iriam se referir aquele cenário. Os exemplos serão todos utilizando esse cenário que tem tantos objetos azuis, verdes e vermelhos. A variável X sempre vai armazenar a cor azul. Acho que facilita muito se os exemplos forem todos para o mesmo cenário. (...)* (Prof6)
- *Exemplifica bem o que tu está querendo dizer.* (Prof7)

Na questão 4.3, foi questionado: "O manual da *GoDonnie* é de fácil leitura?". A maioria dos professores concordou com a questão: concordo totalmente (Prof1, Prof2, Prof3, Prof4, Prof7); concordo parcialmente (Prof5); não concordo nem discordo (Prof6). O Prof5, que concordou parcialmente, e o Prof6, que não concordou nem discordou, possuem preocupação quanto ao público-alvo, se esses irão compreender o manual. O Prof5 questiona, ainda, se o manual não estaria muito denso, pois é um texto corrido. Sugeriu que fosse incluído um menu com links de navegação entre os comandos.

A partir de sugestões, foram incluídos links de navegação, que foram testados junto ao usuário cego (P1), participante da pesquisa.

Seguem alguns comentários.

- *Parece ser fácil, mas não sei se é para mim, que sou uma leitora convencional. (...) Não sei se sequencial não torna ele <o manual> pesado (...). Ter um comando em cada página, que ele <o usuário> possa ser remetido para uma página e não para uma sequência um atrás do outro. Ter um menu com comandos e a página que está o comando que possa selecionar.* (Prof5)
- *Eu enxergo e consigo ler o manual, (...) não sei como uma pessoa cega encararia o manual. (...)* (Prof6)

Na questão 4.4, foi questionado: "O manual da *GoDonnie* possui uma ordem lógica na apresentação dos comandos?". A maioria dos professores concordou com a questão: concordo totalmente (Prof2, Prof4, Prof5, Prof6, Prof7); concordo parcialmente (Prof1, Prof3). O Prof1, que concordou parcialmente, sugeriu alterar a ordem de alguns comandos, da seguinte forma: Primeiro: SAIR, Segundo: FALAR, Terceiro: CRIAR, Quarto: operadores, Quinto: movimentação, giro do robô e percepção do ambiente e, por último, comandos de programação de repetição, seleção, criação de procedimento. O Prof3, que concordou parcialmente, salientou o cuidado na elaboração dos exemplos de forma que não sejam utilizados comandos que ainda não foram explicados.

Seguem alguns depoimentos:

- *(...) Eu to pensando assim: tu é uma pessoa que não enxerga, tem dificuldades sérias de visão, (...) a primeira coisa que eu iria ensinar para ela, como é que tu sai daqui. (...) eu ensinaria é o comando FALAR (...) Vamos começar a criar variáveis. (...) Posso somar umas coisas? Ai tu cai na parte aritmética. Ai já vai fazer contas. Aí tu começa a fazer coisas no sentido "Têm um robô sabia?" O robô está virado para algum lugar, ele pode andar para frente e para trás. Aí tu entra com o assunto robô. Depois vem os comandos de programação mais sofisticados: REPETE, PARA, IF. São os comandos mais complicados, mas até chegar neles tu já está mexendo teu robô para lá e para cá e xeretando o ambiente. Eu reordenaria as coisas.* (Prof1)
- *Essa é a lógica que eu ensino programação.* (Prof2)
- *Se a pessoa tem um background de programação, o jeito que tu está apresentando no manual é exatamente o que é esperado por quem já conhece noções de programação.* (Prof6)
- *Sim, segue uma ordem de passo a passo, como ele deve desde criar variável até movimentação e assim por diante.* (Prof7)

6.4.5 Facilidade de programação

Na questão 5.1, foi questionado: "A *GoDonnie* está adequada para ser utilizada por pessoas com deficiência visual?". A maioria dos professores concordou com a questão: concordo totalmente (Prof2, Prof3); concordo parcialmente (Prof1, Prof4, Prof7); não concordo nem discordo (Prof5, Prof6). Os professores Prof2 e Prof3, que concordam totalmente, já ministraram disciplinas na área de programação para alunos com deficiência visual. O Prof5 e Prof6, que não concordaram nem discordaram, salientaram que não possuem experiência com pessoas com DV. Prof1 e Prof4, que concordaram parcialmente, afirmam que essa questão vai depender do ambiente em que a linguagem estiver inserida. Se o ambiente não for adequado, não vai adiantar ter uma linguagem adequada. O Prof7, que também concordou parcialmente, acredita que, para as pessoas com DV, a parte de localização está adequada para identificar o que tem no ambiente, mas talvez seja mais complexo para realizar determinadas tarefas, como, por exemplo, pegar um objeto e levar para o outro lado.

- *(...) Como é que uma pessoa com dificuldade de enxergar vai chegar e vai usar. Por exemplo, a pessoa com dificuldade de visão está escrevendo um programa, vai chegar um momento que ela vai precisar voltar um pouco antes e botar um outro comando*

que esqueceu na hora. Como é que ela volta antes e insere um comando no meio do que ela já tinha? Isso é fácil para ela? No manual não tenho essas informações. Na hora de ambiente eu não sei como é que é. (...) (Prof1)

- *Aparentemente sim. Agora, como eu não conheço muito pessoas que tenham essa deficiência, se ela de fato vai ser suficiente. A minha percepção é que essas pessoas têm uma sensibilidade muito grande e uma capacidade de abstrair, aí muito provavelmente sim. (...)* (Prof5)

- *(...) tu tem um ambiente virtual, que tu tem um robozinho, que vai poder levar para lá e para cá. Para <o usuário> ele, é algo que ele não vê nada, acho que é mais para localização dele. Acho que para localização é fantástico para ele conseguir fazer, para identificar o que tem no ambiente. Mas não para realizar determinadas tarefas, tipo, eu quero pegar esse quadradinho aqui e levar para o outro lado. Claro, que se explicar direito talvez ele consiga. Mas, para dizer o que tem no ambiente, acho que sim. Acredito que seja uma linguagem de fácil domínio e que seja útil para ele.* (Prof7)

Na questão 5.2, foi questionado: "A *GoDonnie* está adequada para o ensino de programação para usuários com deficiência visual iniciantes em programação?". A maioria dos professores concordou com a questão: concordo totalmente (Prof2, Prof3, Prof5, Prof6, Prof7); concordo parcialmente (Prof1, Prof4). O Prof1, que concordou parcialmente, acredita que os comandos estão de acordo, consistentes, mas, novamente, salienta que depende do ambiente em que será executado. O Prof4 afirma que concordou parcialmente, por não ter conhecimentos suficientes sobre o ensino de programação junto a usuários com DV.

Seguem alguns depoimentos.

- *Pelo mesmo motivo: os comandos me parecem que estão legais, consistentes, mas como é que vai ser o ambiente para usar, pode ser um ambiente que seja bem complicado. (...)* (Prof1)

- *Concordo plenamente. Na verdade, ela é uma linguagem muito parecida com uma linguagem de iniciantes para programação, então eu acho que ela tem muitas características adaptadas.* (Prof2)

Na questão 5.3, foi questionado: "A *GoDonnie* está adequada para programação de robô?". A maioria dos professores concordou com a questão: concordo totalmente (Prof2, Prof3, Prof5, Prof6, Prof7); concordo parcialmente (Prof4); não concordo nem discordo (Prof1). O Prof1, que não concordou nem discordou, e o Prof4, que concordou parcialmente, afirmaram que não tem muita experiencia com programação de robôs. Os professores Prof2, Prof3, Prof5 e o Prof6, apesar de não possuírem experiência em programação de robô, consideram que a *GoDonnie* está adequada a este propósito. O Prof7, que

concordou totalmente e possui experiência de 10 anos em robótica, afirma que a *GoDonnie* atende ao robô virtual, mas tem ressalvas quanto ao robô físico porque podem haver interferências nos giros do robô, bem como na precisão dos deslocamentos, por exemplo.

Seguem alguns depoimentos.

- *Dentro dos objetivos fixados, concordo plenamente.* (Prof3)
- *Parece que sim. São comandos bem orientados, o robô em si tem todo esse outro lado de interface que me parece que sim. (...)* (Prof5)
- *Sim, no sentido que a origem disso tudo é ser baseada na linguagem Logo que tem essas características.* (Prof6)
- *Robô virtual, sim. Robô físico, tenho algumas dúvidas. Por exemplo, no virtual tu tem uma coordenada pré-definida na tua área, e no robô físico tu pode ter um motor que ande mais rápido que o outro e tu ter uma trajetória. O robô físico pode não seguir o trajeto em linha reta, fazer uma trajetória curva e tu não ter esse feedback. Então, para um ambiente virtual, acho que ele funciona 100%. Mas para o ambiente real, teria que ser feito vários testes (...). Mundo real, concordo parcialmente. Mundo virtual, concordo totalmente.* (Prof7)

Na questão 5.4, foi questionado: “A *GoDonnie* não requer muitas ações para resolução de problemas simples?”. Todos os professores concordaram totalmente. O Prof1 e o Prof5, apesar de concordarem, informam que depende dos problemas que desejam resolver. O Prof1 ainda destacou que há ações que são da linguagem e outras que são do robô.

Seguem alguns depoimentos.

- *A linguagem está pensada para o tamanho certo <para fazer um quadrado>, tu vai usar um repita 4 vezes etc. Não vejo problema. Pergunta: não tem a ver com a linguagem e sim onde o robô está. O Robô sabe que tem um objeto 3 passos para frente, eu mando ele andar 5 passos. Ele empurra ou ele bate? <vai andar 3 passos e parar, e sinalizar o que executou>. Essa é uma questão de comportamento do robô, não da linguagem.* (Prof1)
- *Agora não sei se ela é suficiente para problemas de maior complexidade, movimentos diferentes, porque não é só movimento, tem outras ações, outras percepções que não sejam só movimentação, né? (...) Interagir com outros elementos, aí tem outras coisas que dá para ficar inventando.* (Prof5)

6.4.6 Decisões de design

Foram tomadas algumas decisões de design, que podem trazer limitações ao uso da *GoDonnie*. Essas estão centradas em duas decisões principais:

- Utilizar somente dados inteiros
- Não possuir um comando para entrada de dados via teclado (*input*)

Para avaliar essas questões junto aos professores, foram elaboradas as seguintes perguntas semi-abertas, que são discutidas na sequência.

A Questão 1 contém: "A versão atual da *GoDonnie* trabalha somente com o tipo inteiro. Desta forma, não podem ser atribuídos valores fracionários (float), alfanuméricos ou alfabéticos. Sabedor de que a *GoDonnie* é uma linguagem de programação para manipular um robô, qual o impacto dessas decisões de design?". As possibilidades de respostas foram: Alto, Intermediário, Baixo e Sem impacto. A Tabela 6.5 apresenta as respostas dos professores. Importante destacar que o Prof3 que já teve experiência com ensino de programação para alunos com DV, acha que não há impacto. Entretanto, o Prof7, que tem mais de 10 anos de experiência na área de robótica, considera que esse impacto é intermediário.

Tabela 6.5: Questão 1 - Decisões de design quanto a Tipos de dados

	Alto	**Intermediário**	**Baixo**	**Sem impacto**
Tipo Float	Prof5	Proff1, Prof4, Prof7	Prof2	Prof3
Tipo Alfanumérico	Prof5	Prof6	Prof2	Prof3
Tipo Alfabéticos	Prof5	Prof6	Prof2	Prof3

Fonte: a autora

Seguem alguns comentários:

- *O meu instinto diz assim: tu consegue sobreviver com os inteiros que tu tens, não precisa alfabético nem alfanumérico. Não tem muita utilidade para o robô isso. Eu acho que daria para passar para float, então eu acho que é intermediário.* (Prof1)
- *Acho que ela deveria, é uma limitação grande. Principalmente, alfanuméricos e alfabéticos. Porque eu fico pensando se o robô quisesse se comunicar com outro robô, muito provavelmente seria com um alfanumérico ou algum outro elemento que faz parte do ambiente, como o sofá, pela forma, cor é isso. Detecção de outros dados sensíveis, se em um ambiente tem uma pessoa aí é o RG dela. Alguma coisa assim. Mas que é uma evolução natural da linguagem.* (Prof5)

- *Me parece que a medida de um passo é relativa, então medir um 1,3 passos ou 1 passo e meio é desnecessário, não acrescenta muito. Então, eu acho que ela não deveria. Ele poderia manipular tipo alfanumérico e tipo alfabético, no sentido de dar nomes aos objetos. Se eu quisesse concatenar o objeto verde com vermelho. Disparar ações de fala concatenando strings e inteiros.* (Prof6)

Os resultados sugerem a inclusão dos tipos de dados *Float*, alfanumérico e alfabético. Essa questão será examinada também com o público-alvo no Capítulo 8.

A questão 2 contém: "Na versão atual da *GoDonnie*, a entrada de dados é realizada pelo robô, que captura as informações sobre ele (localização, ângulo de referência relacionado ao cenário, distância para obstáculos, por exemplo) e sobre o cenário (quantidade, cor e posição de objetos, por exemplo). Desta forma, não há entrada de dados via teclado. Qual o impacto dessa decisão de design?". A maioria dos professores acredita que seja necessário ter entrada de dados via teclado Prof1, Prof2, Prof4, Prof5, Prof6 (Intermediário). O Prof7 acha que é baixa prioridade ter entrada de dados via teclado, e o Prof3 acredita que não há impacto em não existir entrada de dados via teclado.

Seguem alguns comentários:

- *(...) que dados eu iria querer do teclado? Eu ia querer saber números, como: me diga quanto devo virar à direita agora, no meio do programa. Me diga quantos passos eu tenho que andar? Aí eu informo lá: 5 passos. Ou seja, o programa se adapta se eu pedir coisas diferentes, senão o programa é sempre fixo, de acordo com o que o ambiente permite sentir. Aí tu precisaria ter um comando input que vai ler um número sempre. CRIAR A = input e A mostra para o usuário. Até não muda muito. Mas eu não acho que precise, é um poderia só.* (Prof1)
- *Intermediário. Não é vital, poderia.* (Prof5)
- *Intermediário, porque acrescentar uma feature dá mais possibilidades às pessoas, não quer dizer que ela tenha que usar, mas tem a possibilidade.* (Prof6)
- *Baixo. Acho que foge um pouco da programação ter o input. Melhor abstrair esse tipo de informação.* (Prof7)

Os resultados sugerem a inclusão da entrada de dados via teclado (*input*). Essa questão será examinada também com o público-alvo no Capítulo 8.

6.4.7 Comentários gerais sobre a *GoDonnie*

Prof5 e Prof6, sugeriram fazer avaliação da *GoDonnie* com os professores junto ao ambiente de programação. Prof3 questionou se o manual vai ser fornecido em Braille.

Outro questionamento foi se por enquanto o robô é abstrato, se tudo só ocorre na máquina. A avaliadora respondeu que existe robô virtual e físico. Também questionou se o usuário terá objetivos a cumprir, ou irá fazer o que imagina na cabeça. Foi respondido que serão fornecidas atividades de programação para o usuário executar com a linguagem. Seguem alguns comentários:

- *Tem dois comentários: o primeiro é que a gente que tem experiência, carrega é os nomes dos comandos, achando que os nomes podem não estar bons porque não tem nada para te apoiar; a minha outra dúvida é que ambiente tu está dando para o teu usuário programar, se esse ambiente é mais confortável ou menos confortável, porque é o ambiente que vai deixar ele construir programas mais complicadinhos. Mas da linguagem em si, está fechadinha, ela me parece ok, ela me parece que tem o que devia. Eu fiquei pensando, se o usuário digita 15 - 20 linhas, como é que ele volta para editar algo e botar algo lá no meio. Como ele se localiza? Isso ia ser difícil, ainda mais se eu não enxergo. (...) Tu já ouviu falar em uma linguagem chamada Basic? Tu escreve num lugar e vai cair onde tu está aqui. Sugiro que tu tenha outros comandos. Por exemplo, se começar com número, é para jogar lá para cima. Se quiser falar uma linha específica pode usar: Falar linha 20. Se não tiver linha 20, informar que não tem, se tiver informar: 20 b = 33. Se quiser falar várias linhas, poderia ter: Falar 30−50. No teu mundo faz sentido. Poderia ter ainda comando para deletar linhas (...). E para renumerar (...). Salvar e carregar arquivo, Salvar programa1, carregar programa1.* (Prof1)

- *Acho que nada, tudo que tu me falaste ficou bem claro, só fiquei curiosa pra ver isso funcionando e pra ver como vai ser o efeito, porque uma coisa é tu achar que é assim, mas quando tu trabalha com uma pessoa que tem essa dificuldade é bem mais difícil.* (Prof2)

- *Não tenho nada que já não tenha falado. Ela é simples, mas talvez para próximas versões precisaria ter outras características para talvez resolver problemas mais complexos. E só vai conseguir responder essa pergunta quando estiver utilizando ela e com as pessoas com deficiência visual.* (Prof4)

- *Tanto o manual quanto o teu questionário estão bem completos. Se eu tivesse alguma sugestão é, de repente, nesse estudo de caso, haver a possibilidade de a pessoa ver a linguagem funcionando, ver o ambiente virtual. Mesmo que ela não seja a pessoa para o qual aquele ambiente foi construído, mas ter essa percepção. E, de repente, trabalhar orientado a um exemplo, aí, em cima desse exemplo, ir explorando os outros conceitos que vão agregando e chegar ao total de conceitos. Orientado a um problema, tem esse problema e vou mostrar os comandos em função desse problema.* (Prof5)

- *Achei fantástico, muito legal. Acho que uma ótima tentativa que tem tudo para dar certo. Com relação ao APRENDER ter colchetes, fazer um retângulo, por exemplo, poderia dizer que é uma palavra reservada que não possa utilizar, não vejo motivo para tu ter que criar mais um empecilho para o usuário, como colchetes, chaves ou ponto e vírgula(...).* (Prof7)

6.4.8 Discussão dos resultados

A partir desta avaliação, foi criada uma nova versão do manual, em que foi modificada sua organização e apresentação. A saber, foram incluídos links para facilitar a leitura do documento. Além disso, foi adicionado, no início do manual, o contexto da linguagem e uma seção com problemas mais complexos. Quanto às modificações sugeridas pelos professores para os nomes de comandos, foi optado por manter como estão por já terem sido validados pelo público-alvo. Já quanto à retirada dos colchetes do comando APRENDER, será estudada uma maneira de realizar essa modificação. Os resultados encontrados na presente avaliação sugerem que sejam adicionados os tipos de dados *Float*, alfabéticos e alfanuméricos. Além disso, indicou a necessidade da inclusão da entrada de dados via teclado. Essas questões estão sendo consideradas no estudo da próxima versão da *GoDonnie*.

7. AMBIENTE DE PROGRAMAÇÃO DONNIE

O ambiente de programação *Donnie* é composto por 3 módulos, conforme Figura 7.1. O Módulo 1 corresponde à linguagem de programação *GoDonnie*, já vista no Capítulo 5 e ao ambiente de desenvolvimento que suporta essa linguagem. O Módulo 2 se refere ao simulador gráfico 2D no qual se pode visualizar a execução dos comandos, ou seja, o robô virtual se deslocando no cenário. E o Módulo 3 corresponde ao robô *Donnie* e ao cenário físico. O uso do módulo 1 é obrigatório e pode ser utilizado com o Módulo 2 ou com o Módulo 3, ou seja, não pode ser utilizado com o Módulo 2 e Módulo 3 ao mesmo tempo. No contexto desta dissertação, apenas o módulo 1 e módulo 2 serão avaliados.

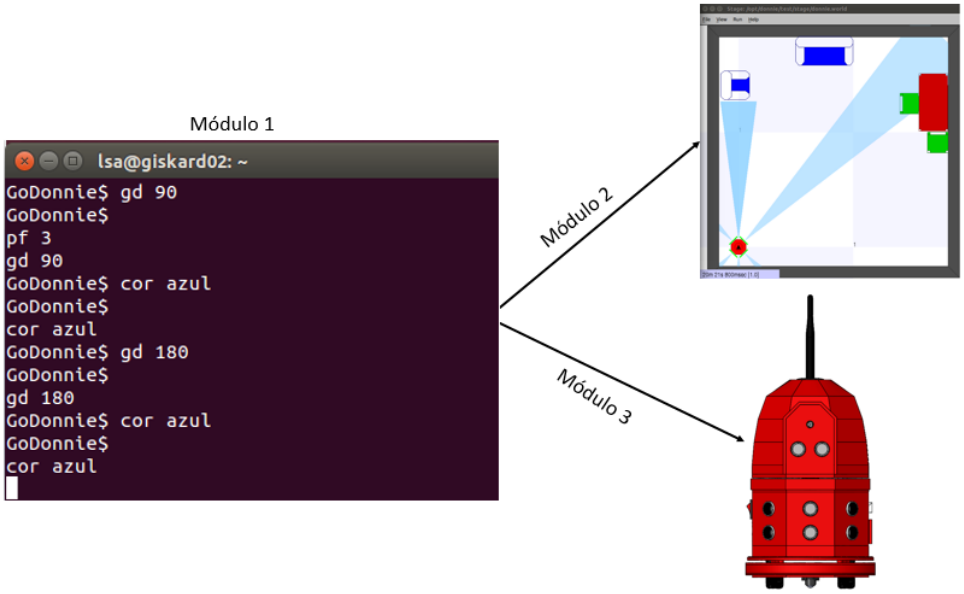


Figura 7.1: Ambiente do projeto
Fonte: a autora

É importante destacar que o ambiente do projeto foi foco de atividades relacionadas aos projetos aprovados nos editais FACIN 2013, PEC-DES 2014, 2015 e 2016, os quais iniciaram a construção do robô *Donnie*. Estes projetos apoiaram a pesquisa por meio da concessão de bolsas de iniciação científica e de recursos para construção do robô.

A seguir serão descritos cada módulo do ambiente de programação *Donnie*.

7.1 Módulo 1: *GoDonnie* e ambiente de desenvolvimento

O ambiente de desenvolvimento inclui um interpretador (Client), um editor e simulador 2D. O ambiente possui dois tipos de *feedbacks*: mensagens faladas que são executadas com o recurso de *Text-to-speech* (*TTS*) da *Google* e sons icônicos (sons de passos, giros e colisões) que são arquivos mp3 carregados automáticamente no sistema. Alguns comandos combinam *feedbacks* de mensagens faladas e sons icônicos, e outros somente mensagens faladas (Exemplo: Tabela 7.1). Além disso, é utilizado o leitor de telas Orca que é nativo do Sistema operacional utilizado *Linux*. O leitor é utilizado para que o usuário consiga digitar de forma autônoma os comandos da linguagem, assim o Orca vai reproduzindo o que o usuário digita no terminal.

Tabela 7.1: *Feedbacks* sonoros do sistema

Comando	**Som icônico**	**Mensagem falada**
PF	som de passos	Andei n passos para frente.
GD	som de giro	Girei n graus para direita.
ESPIAR	som de giro	A 40 graus a esquerda: 1 objeto de cor verde a 2 passos. A 90 graus a direita: 1 objeto da cor vermelha a 4 passos.
ESTADO	–	Comando 1 foi PF 3, andou 2, bateu, posição [2,0,0]

Fonte: a autora

7.2 Módulo 2: simulador gráfico 2D

O simulador gráfico 2D (Módulo 2) funciona em conjunto com o Módulo 1. A parte gráfica do simulador é configurável, ou seja, pode-se adicionar novos objetos e mudar a disposição dos objetos (Figura 7.2). No momento, é possível utilizar somente objetos com as cores azul, verde e vermelha. Dentre os objetivos para criação do Módulo 2, têm-se:

- permitir o uso do ambiente sem haver um robô físico e um cenário físico.
- permitir que pessoas com baixa visão ou videntes possam acompanhar visualmente o pensamento de quem está programando o robô.

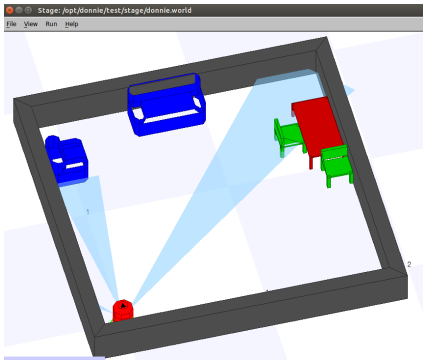


Figura 7.2: Simulador Gráfico 2D de outro ângulo
Fonte: a autora

7.3 Módulo 3: *Donnie* e cenário

No contexto dessa dissertação não foram realizadas atividades no Módulo 3 do ambiente, pois o mesmo ainda não estava integrado à linguagem *GoDonnie*. Apenas foram realizadas as definições do cenário. A título informacional o *Donnie* e cenário estão descritos a seguir.

O robô *Donnie* (Figura 7.3) foi construído com tecnologia de hardware *open source* (*Arduino*, *Raspberry Pi* e peças impressas em 3D). Ele é composto por 1 auto-falante, 1 câmera, 7 sensores ultra-sônicos e 2 “para-choques”. O *Donnie* requer conectividade *wi-fi* para se conectar ao ambiente de desenvolvimento.

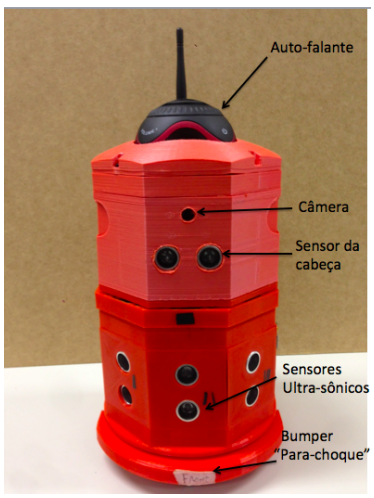


Figura 7.3: Donnie
Fonte: a autora

O recurso de auto-falante é utilizado para emitir informações sonoras tais como alerta de colisão e de identificação de obstáculos, som de passos para frente e para trás, som de rotação para direita e para esquerda. O som do robô é independente da interface sonora do ambiente virtual. O *Donnie* possui um botão físico para que o auto-falante seja ligado e desligado. Desta forma, se o auto-falante estiver ligado o som será emitido somente pelo robô físico.

A câmera é utilizada para identificar os objetos do cenário. Assim, o robô pode identificar objetos de diferentes formas nas cores vermelho, verde e azul. A câmera fica posicionada na cabeça do robô e executa giros de até 180 graus. Essa limitação na rotação é decorrente do *servo* motor utilizado no robô.

Os sensores ultrassônicos são utilizados para evitar a colisão e detectar a distância dos objetos ao redor do robô. O sensor da cabeça detecta a distância até os objetos identificados pela câmera. Os demais sensores estão localizados à frente, frente e à direita, frente e à esquerda, atrás, atrás e à esquerda, e atrás e à direita. A diferença entre o sensor da frente e o da cabeça, é que o da cabeça está em uma estrutura que gira 180 graus.

O robô possui, ainda, recursos como “para-choque”, que detecta a colisão física que pode ocorrer quando os sensores ultrassônicos de distância não detectarem os objetos.

O cenário é o ambiente físico em que o robô *Donnie* se desloca (Figura 7.4). Esse ambiente foi criado com material em MDF, tendo 3 modelos de peças. Dois modelos se referem às paredes que possuem 40x20cm e 40x40cm. O outro modelo se refere à peça para conectar as paredes e mede 10x10cm. Desta forma, o cenário físico pode ser ajustado conforme as necessidades do ambiente. Além disso, no cenário podem ser adicionados objetos para serem identificados pelo robô, podendo ser nas cores vermelha, verde e azul. As peças para compor o cenário foram definidas no escopo dessa dissertação.

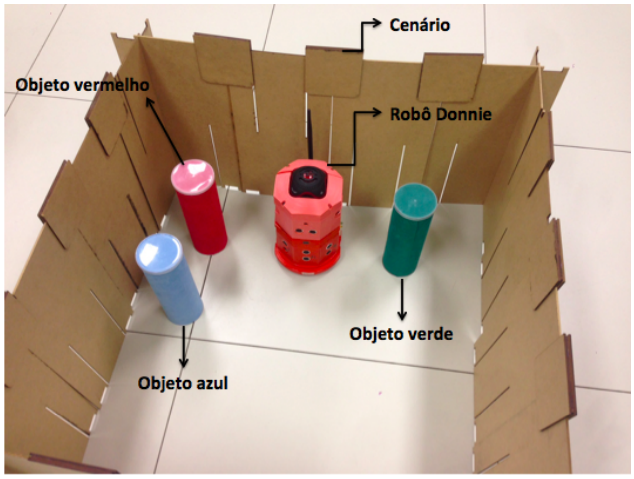


Figura 7.4: Cenário para deslocamento do Donnie
Fonte: a autora

8. AVALIAÇÃO DO AMBIENTE DE PROGRAMAÇÃO DONNIE

A avaliação do Ambiente de Programação *Donnie* (APD) inclui a avaliação do Módulo 1 e Módulo 2. A saber, o Módulo 3, que corresponde ao robô físico, não foi avaliado porque, conforme já descrito, até o momento desta etapa, não estava disponível para avaliação junto a usuários finais. As avaliações dos Módulos 1 e 2 foram conduzidas pela autora desta dissertação, acompanhada de 2 alunos de iniciação científica, que receberam um treinamento e registraram a avaliação por meio de gravação de vídeo, áudio, fotografia, além de controlarem o tempo e fazerem anotações sobre suas observações. Esses alunos não são os mesmos que realizaram as avaliações descritas no Capítulo 6.

A equipe realizou um estudo piloto para verificar o protocolo de avaliação, que incluiu o entendimento sobre o TCLE, o instrumento de coleta dos dados, o questionário e as tarefas a serem realizadas. Essa avaliação preliminar ocorreu com um participante vidente, que utilizou somente o Módulo 1 do ambiente, ou seja, não visualizou o cenário e o robô virtual, que são elementos do Módulo 2. Durante o piloto, pode-se observar que o mapa tátil poderia ser maior, permitindo que o usuário pudesse ter uma melhor compreensão do cenário, dos objetos e da posição do robô. Essa questão não foi apontada pelo participante, mas foi observada pela equipe de avaliação. Assim, foram disponibilizados 4 conjuntos de mapas de tamanho 60cm x 40cm, totalizando 120 x 80, e 748 quadros, que correspondem aos passos do robô.

A avaliação com os usuários finais contou com a participação de um participante cego e outro com baixa visão, que atuaram nas avaliações anteriores da *GoDonnie*, descritas no Capítulo 6 como P1 e P2, respectivamente. O participante cego (P1) foi caracterizado como não programador, pois sua experiência na área foi adquirida na etapa de avaliação da *GoDonnie*. O participante com baixa visão (P2) foi caracterizado como programador, pois possui experiência em programação anterior ao uso da *GoDonnie*.

Para a avaliação, foram criadas atividades de programação e um questionário. Havia 6 atividades de programação e 1 para exploração do manual da *GoDonnie*. A Tabela 8.1 apresenta as atividades.

Tabela 8.1: Atividades

Atividade	Tipo	Descrição
1	Programação	O robô está em um cenário que possui o tamanho de 34 passos na vertical e 34 passos na horizontal. Esse cenário possui alguns objetos que estão distribuídos no espaço. Você tem o desafio de conseguir identificá-los e informar onde estão. O robô está na posição x=10, y=0 e virado para o norte, do lado esquerdo do cenário. Os objetos estão localizados na parte superior do cenário. Precisamos colocar o robô mais para o meio do cenário. Como fazer isso? Precisamos saber quantos objetos existem. Como fazer isso? E quantos objetos têm com a cor azul? E com a cor verde? E com a cor vermelha? Agora, vamos ver como poderíamos representar esse cenário no mapa tátil. Aqui está o mapa. Esses objetos representam os objetos do cenário. Você pode distribuí-los? Se precisar espiar novamente o cenário, fique à vontade para usar o computador.
2	Programação	Em um curso de programação sobre a GoDonnie, foi ensinado que o comando DISTÂNCIA informa a distância do robô até um objeto. Há um objeto na frente do robô. Verifique a distância do robô para este objeto. Lembre-se que é necessário informar ao robô que ele deve falar a informação.
3	Manual	Nos exercícios 1 e 2 foram utilizados os comandos FALAR, PF, GD, GE, POS e DISTÂNCIA. Verifique a explicação desses comandos no manual.
4	Programação	Supomos que foi oferecido um curso para ensinar programação usando a GoDonnie. Durante esse curso, um aluno digitou alguns comandos. Eu vou ler os comandos para que você digite. Existem erros nessa programação? PF 5 PF3. Existe erro. Qual seria? OK, vamos digitar e ouvir a explicação sobre o erro. Corrija o programa.
5	Programação	No cenário, há objetos nas cores vermelha, verde e azul. Informe se há objetos na cor verde, informe a quantidade. Se não houver, informe que não há objetos nessa cor.
6	Programação	O robô fez um quadrado de tamanho 3. Mas ficou pequeno. Faça um programa para que o robô faça um quadrado maior, sendo que o tamanho máximo do lado deve ser 6.

Fonte: a autora

O questionário conteve questões relacionadas aos seguintes critérios: 1. Facilidade de uso (1.1 até 1.7); 2. Utilidade (2.1 até 2.5); 3. Prevenção e tratamento de erros (3.1 até 3.6); 4. Interface sonora (4.1 até 4.12); 5. O&M (5.1 até 5.5); 6. Programação (6.1 até 6.5); 7. Ajuda e documentação (7.1 até 7.4) e 8. Satisfação de uso (8.1 até 8.3), totalizando 48 questões fechadas. A cada grupo de questões, havia uma questão de preenchimento op-

cional para o caso do participante desejar expor alguma observação. Para essas questões, foi utilizada a escala de *Likert* de 5 pontos (concordo totalmente, concordo parcialmente, não concordo nem discordo, discordo parcialmente, discordo totalmente). A Tabela 8.2 relaciona a atividade com o critério de avaliação e as respectivas questões do questionário. Após o preenchimento dessas questões, era apresentado o restante do questionário, que compunham os critérios já citados, além de 2 questões sobre decisões de design da *GoDonnie*, 3 questões gerais sobre a linguagem e 1 opcional para fazer uma conclusão geral sobre a *GoDonnie*. O Apêndice G apresenta o questionário que foi aplicado.

Tabela 8.2: Relação entre Atividade e questões do questionário

Atividade	**Categoria do questionário**	**Questões**
1 e 2	Facilidade de uso	1.6
	Utilidade	2.2 até 2.5
	Prevenção e tratamento de erros	3.1 e 3.6
	Interface sonora	4.1 até 4.12
	Orientação e Mobilidade	5.1 até 5.5
	Satisfação de uso	8.2
3	Ajuda e documentação	7.1 até 7.4
4	Prevenção e tratamento de erros	3.2 até 3.5
5 e 6	Facilidade de uso	1.7
	Utilidade	2.1
	Programação	6.5

Fonte: a autora

Nas questões relacionadas à usabilidade, buscou-se verificar:

- Facilidade de uso: se a linguagem *GoDonnie* (sintaxe e semântica), interface sonora e mensagens são de fácil entendimento.
- Utilidade: se é possível compreender os objetos do cenário e a relação desses objetos com o cenário, e se é possível a linguagem ser utilizada para resolver diferentes problemas.
- Prevenção e tratamento de erros: se o ambiente *Donnie* evita que erros aconteçam, se o usuário compreende o erro e consegue corrigi-lo.
- Ajuda e documentação: se o manual da *GoDonnie* é de fácil leitura, se pode ser utilizado para o aprendizado da mesma e se possui uma ordem lógica na apresentação dos comandos.
- Satisfação de uso: se é agradável e prazeroso utilizar o ambiente *Donnie*

As demais categorias tiveram os seguintes objetivos:

- Interface Sonora: buscou-se identificar se os *feedbacks* icônicos e de mensagens faladas estavam adequados, se eram de fácil compreensão e necessários para melhor entendimento da execução.
- O&M: examinou-se se o ambiente *Donnie* pode ser um recurso utilizado previamente pelo usuário para explorar um ambiente, se o usuário compreende a relação dos objetos com o robô, onde está o robô no ambiente, a direção do robô e se o usuário consegue externalizar o seu plano de ação para deslocar o robô de um ponto a outro no ambiente.
- Programação: procurou-se verificar se o ambiente de programação *Donnie* estava adequado para ser utilizado por pessoas com DV e se estava adequado ao ensino de programação.
- Decisões de design: procurou-se identificar a importância de acrescentar novos tipos de dados ou entradas na linguagem.
- Questões gerais: buscou-se verificar se os usuários recomendariam o uso do ambiente de programação *Donnie* a outros usuários, se achavam que o ambiente pode ser utilizado por iniciantes em programação e se desejariam continuar utilizando-o. A questão opcional serviu para indicarem algum caso que durante a avaliação não tivesse sido coberto.

Sobre o procedimento de avaliação, destacam-se as seguintes etapas:

- Preparação da sala: disposição do computador, do mapa tátil e dos objetos que representaram os objetos do cenário. O robô já ficava na posição inicial para a realização das tarefas. Os demais objetos ficavam à disposição para que o usuário os posicionassem conforme realizavam as tarefas.
- Teste dos equipamentos de coleta de dados: câmera de vídeo e de áudio do smartphone, e software *Simple Screen Recorder* para captura de tela.
- Preparação do notebook: instalação do ambiente de programação *GoDonnie*, habilitação do leitor de telas *Orca* e ativação do software de captura de tela. Também foi acoplado um teclado *desktop* para facilitar a digitação dos usuários com DV.
- Inicialização do ambiente de programação da *GoDonnie*: posicionamento do robô no cenário virtual.

Um mapa tátil de E.V.A foi utilizado para representar o cenário virtual, assim como um pequeno objeto para representar o robô. Esses materiais foram os mesmos utilizados

na avaliação da linguagem *GoDonnie*, descrita no Capítulo 6. Ressalta-se, entretanto, que o mapa tátil foi aumentado após a realização do teste piloto. Ainda, optou-se por utilizar objetos concretos de formatos diferentes para representar os tipos diferentes de objetos que estavam no cenário virtual. Os objetos concretos tinham cores diferentes para facilitar a observação por parte dos avaliadores e para uso do participante com baixa visão. A Figura 8.1 representa os objetos utilizados. Além desses, também foi utilizado o instrumento de conferência dos ângulos, já mencionado no Capítulo 6.

Figura 8.1: Peças que representam os objetos do cenário. Da esquerda para a direita: objetos azuis, objetos vermelhos e objetos verdes
Fonte: a autora

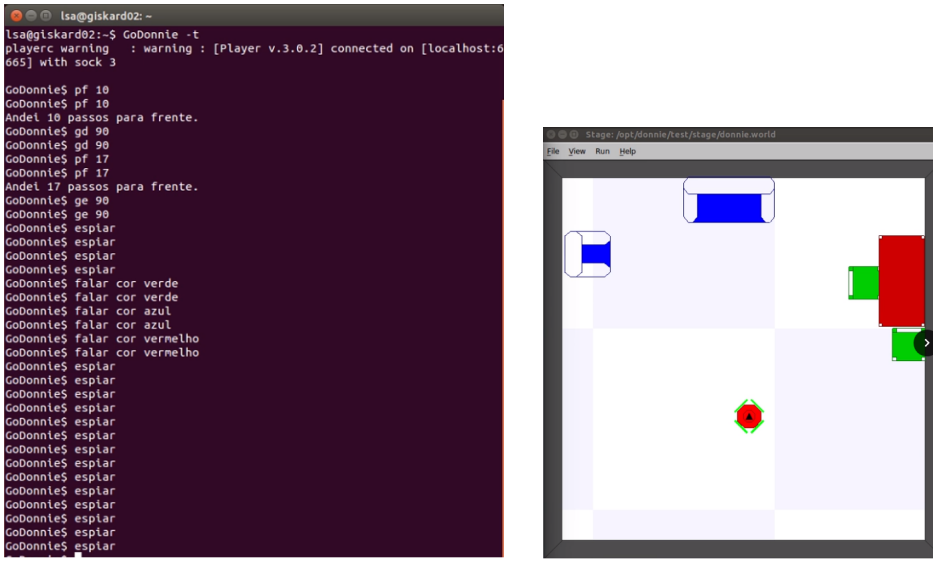
Previamente à avaliação, os participantes treinaram a utilização do *Linux* com o leitor de telas *Orca*. Isso foi necessário para que os participantes se sentissem familiarizados com o ambiente antes da avaliação, já que eram usuários do sistema operacional *Windows* e do leitor de telas *NVDA*. A descrição de como foi realizado o treinamento pode ser visto no Apêndice H.

A seguir estão descritas as duas etapas de avaliação.

8.1 Etapa 1 - Avaliação com participante cego sem conhecimento em programação

A descrição do perfil do P1 encontra-se no Capítulo 6. A avaliação teve duração de 2h06min. O participante realizou todas as atividades com sucesso. Na atividade 1, o P1 utilizou os comandos de deslocamento (PF) e rotação (GD e GE) para ir para o centro do cenário. Após, para descobrir o que existia no ambiente utilizou o comando ESPIAR. Para identificar quantos objetos havia de cada cor, utilizou o comando COR. Quando solicitado ao participante que montasse o mapa tátil, o participante fez uso novamente do comando ESPIAR. A saber, o comando foi utilizado 9 vezes e não 18 vezes conforme indicado na Figura 8.2a. Essa diferença se refere ao fato de o sistema estar repetindo a linha que é digitada pelo usuário. A Figura 8.2b representa o cenário com a posição atual do robô.

A Figura 8.3a ilustra o cenário virtual enquanto a Figura 8.3b ilustra o cenário construído pelo participante P1. A Figura 8.4 representa o mapa tátil construído pelo parti-



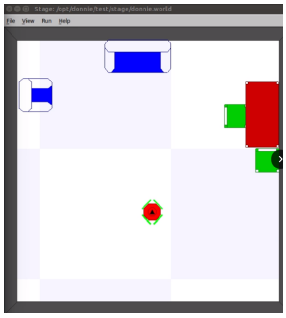
(a) comandos utilizados pelo P1 (b) cenário com a posição atual do robô

Figura 8.2: Atividade 1 - Resposta P1
Fonte: a autora

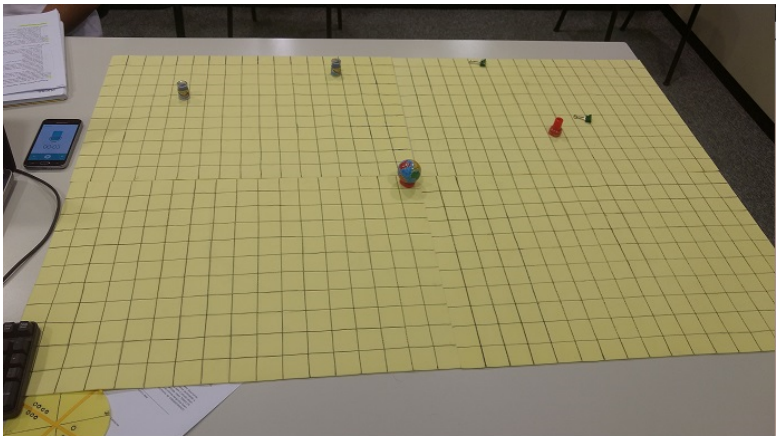
cipante, visto por outro ponto de vista. Pode-se verificar que o usuário conseguiu montar um mapa próximo ao cenário real. Dessa forma, demonstra que conseguiu realizar um mapa mental do ambiente com o *feedback* sonoro fornecido pela *GoDonnie*. A seguir, a descrição fornecida pelo comando ESPIAR utilizada pelo usuário para montar o mapa tátil:

> A 60 graus à direita: um objeto de cor verde à 18 passos. A 60 graus à direita: um objeto de cor vermelha à 16 passos. A 30 graus à direita: um objeto de cor verde à 29 passos. A 15 graus à esquerda: um objeto de cor azul à 23 passos. A 45 graus à esquerda: um objeto de cor azul à 28 passos.

O participante utilizou o instrumento de conferência de ângulos (Figura 6.3) para entender os graus informados pelo *feedback*.



(a) Cenário virtual



(b) Mapa tátil

Figura 8.3: Mapa tátil montado pelo P1
Fonte: a autora

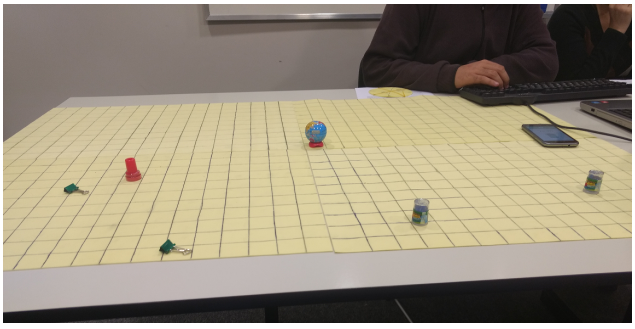
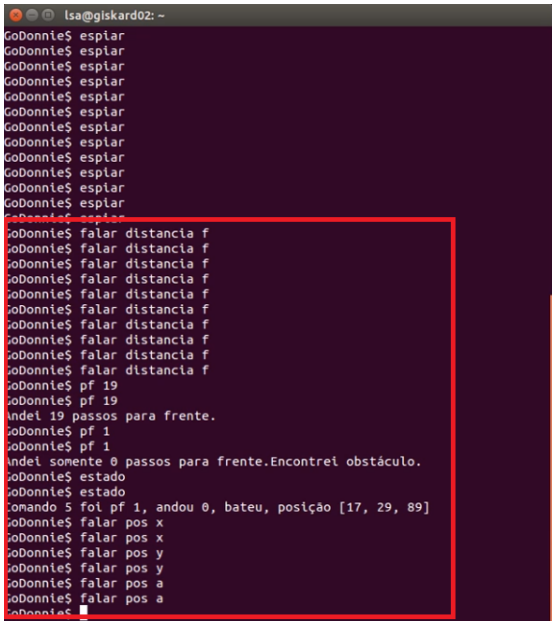


Figura 8.4: Mapa tátil montado por P1 visto de outro ângulo
Fonte: a autora

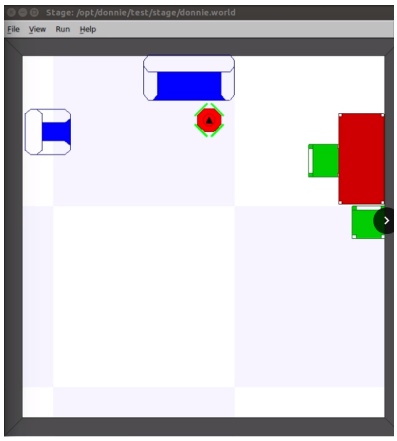
Durante a atividade, o participante sugeriu que os comandos COR VERDE, COR AZUL e COR VERMELHO também trouxessem as informações de posição, como apresentado pelo comando ESPIAR. Notou-se que o usuário ficou um pouco impaciente na execução do comando ESPIAR, pois este, como faz uma varredura de 180 graus do am-

biente, possui uma execução de aproximadamente 51 segundos. O participante também questionou por que a voz do Terminal de Programação, que é fornecida pelo *Orca*, é diferente da voz fornecida pelos *feedbacks* sonoros (*TTS*), que é da *Google*. A saber, a voz apresentada no Terminal de Programação é uma voz masculina e a do *TTS* é uma voz feminina. O participante tem preferência pela voz apresentada pelo *TTS*.

Na atividade 2, o participante utilizou o comando FALAR e DISTÂNCIA F para saber a distância até um objeto a frente do robô e, em seguida, usou o comando de deslocamento PF para ir até o objeto. Para identificar qual a posição do robô no eixo X e Y e o ângulo, o P1 utilizou o comando POS X, POS Y e POS A. O participante lembrou com facilidade do comando POS X e POS Y, sendo necessário olhar no manual o comando POS A. A Figura 8.5a representa os comandos utilizados pelo P1 enquanto a Figura 8.5b representa o cenário com a posição atual do robô.



(a) comandos utilizados pelo P1



(b) cenário com a posição atual do robô

Figura 8.5: Atividade 2 - Resposta P1
Fonte: a autora

O retorno de áudio apresentado pelo comando DISTÂNCIA F estava mais baixo que os outros comandos. Assim, não foi possível ouvir com clareza o retorno do comando. Apesar disso, o usuário realizou a atividade com sucesso. Destaca-se que não se pode observar o motivo dessa incoerência de volumes, tendo em vista que outras informações faladas pelo TTS estavam com volume definido pelos avaliadores.

Na atividade 3, o participante consultou o manual para verificar as explicações dos comandos utilizados nas atividades anteriores. O participante considerou que as explicações estão de acordo, porém, salientou que é necessário adicionar mais exemplos para

enriquecer o manual. Esta questão também foi apontada pelos professores, conforme descrito no Capítulo 6.

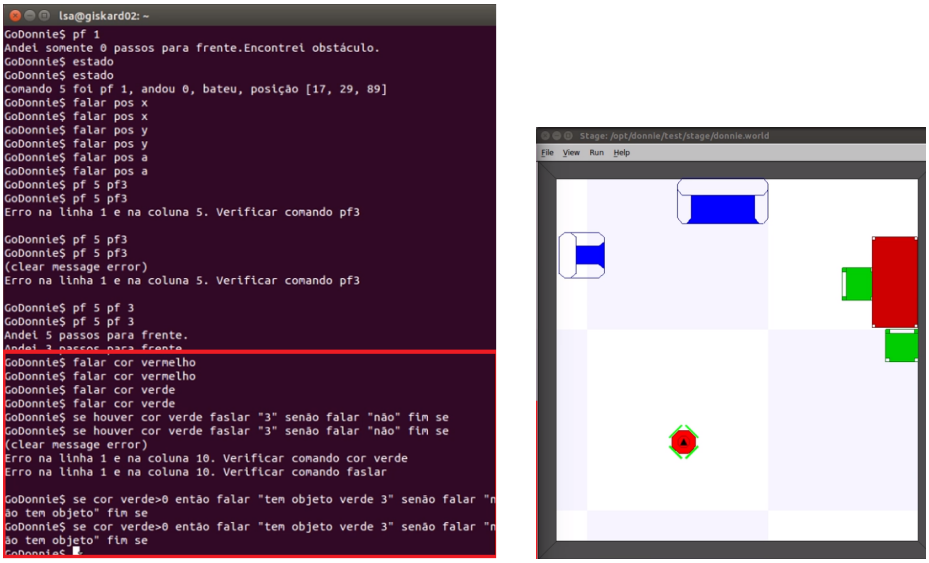
Na atividade 4, os avaliadores afastaram um pouco o robô do objeto sofá do cenário. O participante indicou que os comandos possuíam erro antes de digitá-los. A mensagem do sistema confirmou sua percepção e o usuário corrigiu o erro com facilidade. A Figura 8.6a representa os comandos utilizados pelo P1, enquanto a Figura 8.6b representa o cenário com a posição atual do robô.

(a) comandos utilizados pelo P1

(b) cenário com a posição atual do robô

Figura 8.6: Atividade 4 - Resposta P1
Fonte: a autora

Na atividade 5, os avaliadores afastaram um pouco o robô dos objetos do cenário. Era esperado que o participante utilizasse o comando SE para dizer se existia ou não objeto de cor verde. O P1, primeiramente, utilizou o comando COR VERDE. Após, verificou que necessitava utilizar o comando SE, mas não lembrou exatamente da sintaxe do comando digitando no terminal “SE houver cor verde falar “3” SENÃO falar “não” FIM SE”. Depois, com algumas dicas da avaliadora e lendo o manual, relembrou a sintaxe e a lógica desse comando. A Figura 8.7a representa os comandos utilizados pelo P1, enquanto a Figura 8.7b representa o cenário com a posição atual do robô.

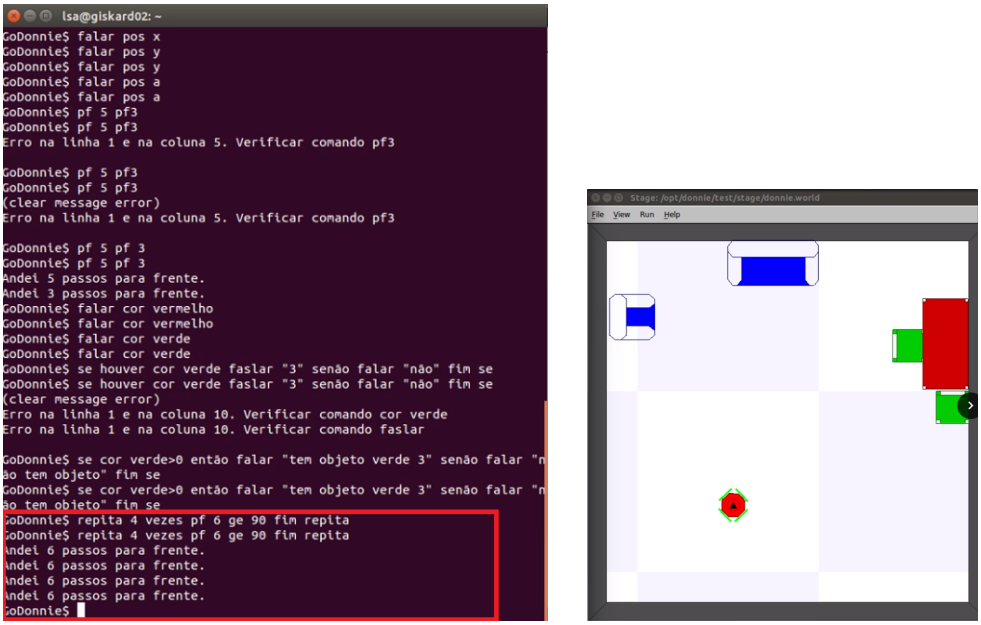


(a) comandos utilizados pelo P1 (b) cenário com a posição atual do robô

Figura 8.7: Atividade 5 - Resposta P1

Fonte: a autora

Na atividade 6, foi solicitado que o participante fizesse um quadrado de tamanho de lado 6. O participante utilizou o comando REPITA. Após, ao ser questionado se existia outra forma de resolver o problema, comentou que poderia ser utilizada uma sequência de comandos PF e GD, ou o comando ENQUANTO. A Figura 8.8a representa os comandos utilizados pelo P1, enquanto a Figura 8.8b representa o cenário com a posição atual do robô. No caso, a posição e sentido final do robô permanecem os mesmos porque foram realizados os movimentos para formar um quadrado.



(a) comandos utilizados pelo P1 (b) cenário com a posição atual do robô

Figura 8.8: Atividade 6 - Resposta P1
Fonte: a autora

Conforme já descrito, havia aplicação de partes do questionário intercalada à realização das atividades. Segue uma análise das respostas ao questionário:

8.1.1 Facilidade de uso

O participante P1 concordou totalmente que a sintaxe e semântica da *GoDonnie*, sua interface sonora e o cenário virtual são de fácil entendimento. Concordou totalmente que é fácil realizar as atividades no ambiente. Além disso, comentou que consultar o manual facilita a resolução de problemas. Também concordou que os comandos da *GoDonnie* permitem resolver um problema de diferentes formas. Quanto às mensagens sonoras, o P1 discordou parcialmente que são de fácil entendimento, pois, durante a avaliação, ao utilizar o comando DISTÂNCIA F, a informação de áudio, fornecida pelo *TTS*, foi falada com volume mais baixo. Essa informação se refere à distância em passos até um objeto a frente do robô. Ou seja, não é um som icônico, e sim um arquivo de áudio, que é executado conforme o volume de som do computador.

8.1.2 Utilidade

O participante P1 concordou totalmente que a linguagem pode ser utilizada para resolver diferentes atividades. Concordou totalmente que o uso da *GoDonnie* permite compreender objetos em um ambiente virtual, mas salientou que seria interessante o comando COR fornecer a localização dos objetos. Também concordou totalmente que o uso do ambiente *Donnie* permite compreender a relação dos objetos com o ambiente virtual. Destaca-se que essa sugestão do participante não havia sido indicada em avaliação anterior, descrita no Capítulo 6.

teste

8.1.3 Prevenção e Tratamento de erros

O participante concordou totalmente que a *GoDonnie* evita que erros aconteçam. Além disso, concordou totalmente que a *GoDonnie* permite que o usuário compreenda quando ocorre um erro, o tipo de erro, como corrigi-lo e corrija o erro, a partir das informações fornecidas. Foi perguntado ao usuário se as regras de prevenção de erro, por exemplo, as de interrupção de deslocamento quando haveria uma colisão futura, são adequadas. O usuário concordou totalmente, mas acrescentou que poderia haver um barulho de colisão. Salienta-se que existe um som icônico de colisão, mas este não foi identificado e reconhecido pelo participante P1.

8.1.4 Ajuda e Documentação

Quanto ao manual da *GoDonnie*, o participante concordou totalmente que o mesmo é adequado ao aprendizado da linguagem e que os exemplos são úteis para o aprendizado dos comandos. Entretanto, novamente, sugeriu que houvesse mais exemplos no manual. Além disso, concordou totalmente que o manual é de fácil leitura e que possui uma ordem lógica de apresentação. Destaca-se que a versão do manual se difere daquela que foi utilizada na avaliação descrita no Capítulo 6, pois foi refeita considerando as sugestões dos professores de programação, descritas nesse mesmo Capítulo.

8.1.5 Satisfação de uso

O participante P1 concordou totalmente que é agradável utilizar a *GoDonnie* para aprender programação. Também concordou que o uso da *GoDonnie* é interessante para compreender um espaço real. Por fim, concordou totalmente que o uso da *GoDonnie* é prazeroso.

8.1.6 Interface sonora

Quanto à interface sonora, o participante P1 concordou totalmente que os sons icônicos (passos, giros, colisão) e as mensagens faladas são intuitivas, mesmo não tendo reconhecido o som de colisão na atividade 2. Além disso, concordou totalmente que os sons icônicos e as mensagens faladas são necessárias para a compreensão da execução dos comandos da *GoDonnie*, sendo complementares e necessários. Também concordou totalmente que os sons icônicos e as mensagens faladas são agradáveis, assim como a sua velocidade de execução. Ressalta-se que a velocidade de execução utilizada foi a *default* do *Google*, não tendo sido implementado um método de configuração da mesma. O P1 concordou parcialmente que as mensagens faladas auxiliam o entendimento dos sons icônicos, e sua resposta foi novamente baseada na execução do comando DISTÂNCIA F, já explicado anteriormente. Apesar disso, concordou totalmente que as mensagens faladas são necessárias para um melhor entendimento dos sons icônicos.

8.1.7 Orientação e Mobilidade

O participante P1 concordou totalmente que a *GoDonnie* é um recurso que pode ser utilizado previamente pelo usuário para explorar um ambiente. Ainda, concordou totalmente que a *GoDonnie* permite compreender a relação dos objetos com o robô, onde está o robô no ambiente e a direção do robô no ambiente. Além disso, o P1 concordou totalmente que a *GoDonnie* permite que o usuário externalize seu plano de ação mental para deslocar o robô de um ponto a outro do ambiente.

8.1.8 Programação

O participante P1 concordou totalmente que a *GoDonnie* é adequada para ser utilizada por pessoas com DV, mas, novamente, destacou que o retorno do comando DISTÂN-

CIA F deve ser corrigido. Além disso, concordou totalmente que a *GoDonnie* é adequada para o ensino de programação para iniciantes em programação e para programação de robô. Também concordou totalmente que a *GoDonnie* não requer muitas ações para realização de problemas simples, assim como permite diferentes formas de resolução de um mesmo problema.

8.1.9 Decisões de design

Salientou-se para o usuário as decisões de design tomadas, que podem trazer limitações ao uso da *GoDonnie*. Essas estão centradas em duas decisões principais:

- Utilizar somente dados inteiros
- Não possuir um comando para entrada de dados via teclado (*input*)

Para avaliar essas questões junto ao usuário final, foram elaboradas duas questões semiabertas, que são discutidas na sequência.

A primeira questão foi sobre a escolha por manipular dados somente do tipo números inteiros, não sendo possível trabalhar com números com vírgula e nem com letras. Assim, não é possível mandar o robô andar para frente 2 passos e meio, e nem solicitar que seja armazenado um nome em uma variável. O participante P1 informa que, para ele, não existe impacto de a *GoDonnie* somente manipular tipos inteiros. Não realizou comentários sobre isso.

A segunda questão foi sobre a entrada de dados ser realizada pela captura de informações pelo robô, diferente de outras linguagens de programação em que o usuário pode informar dados de entrada. Por exemplo, o programador pode fazer um programa que pergunte a idade de uma pessoa e, quando a pessoa digitasse a idade, o robô poderia andar essa quantidade de passos. O participante P1 acredita que o impacto é alto, pois a *GoDonnie* deveria permitir a entrada de dados via teclado.

8.1.10 Questões gerais

Para avaliar a *GoDonnie* de forma geral, foram elaboradas quatro questões abertas, que são discutidas a seguir:

Na primeira pergunta foi questionado: "A programação de robôs vem sendo utilizada como estratégia para a aprendizagem de programação por usuários iniciantes. Você considera que a *GoDonnie* pode ser utilizada por pessoas com deficiência visual que sejam

iniciantes em programação?". O participante considera que as áreas de robótica e de programação são avançadas, e que a *GoDonnie* representaria uma ”porta aberta para se ter um sistema mais inclusivo, mais acessível´´, propiciando melhor atendimento às pessoas com DV interessadas nesses temas.

Na segunda pergunta foi questionado: "Você recomendaria o uso da *GoDonnie* para outras pessoas que desejam aprender a programar robôs?". O participante P1 concorda totalmente e destaca que: *isso faz aprender outro meio de ensino além do tradicional. Aprender de uma outra forma, uma forma mais informática. Abre um novo campo de ensino, de noção de espaço, de números, de estratégias.*

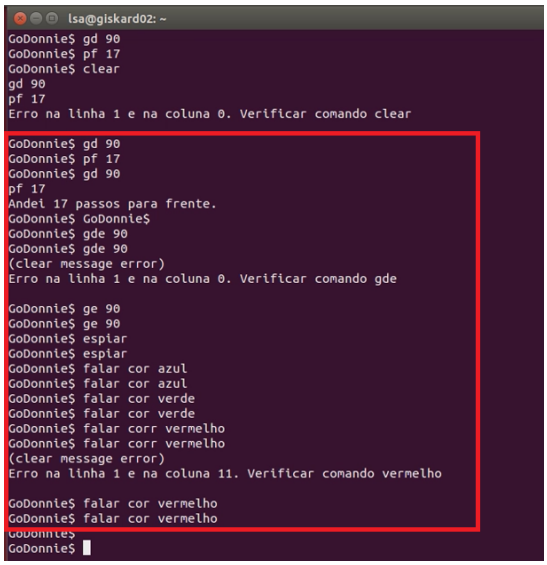
Na terceira pergunta foi questionado: "Você gostaria de continuar utilizando a *GoDonnie*?". O participante P1 informa que sim e justifica: *Concordo totalmente, porque eu queria aprender mais sobre outras formas de programação, dominar mais o manual e tentar ir um pouco mais além do que já aprendi*.

Na quarta pergunta foi questionado: "Se desejar, faça outros comentários sobre a *GoDonnie*". O participante P1 comentou que seria interessante, ao trabalhar com alunos iniciantes, ter atividades que comecem do nível básico até o intermediário. Comentou também que gostou do *feedback*, pois o mesmo falava tudo que ocorria. Resolveu avaliar o ambiente com nota 8,0 numa escala de 0,0 à 10,0 (por opção própria). E justificou: *porque eu queria que fosse a voz da mulher <TTS> ao invés do homem <Orca>. Ainda, tem alguns símbolos no manual que não são lidos pelo Orca, mas seriam lidos pelo NVDA. (...) A respeito daquela voz, eu acho que poderia falar mais alto <DISTÂNCIA>. E eu quero mais exemplos no manual.*

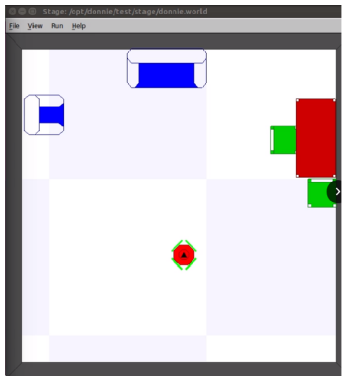
8.2 Etapa 2 - Avaliação com participante com baixa visão com conhecimento em programação

A descrição do perfil do P2 encontra-se no Capítulo 6. A avaliação teve duração de 1h40min, e foram utilizados os módulos 1 e 2. O participante fez uso de uma lupa (física, não virtual) para melhor visualizar o mapa virtual, representado pelo Módulo 2. O participante realizou todas as atividades com sucesso. Na atividade 1, o P2 utilizou os comandos de deslocamento (PF) e rotação (GD e GE) para ir para o centro do cenário. Após, para descobrir o que existia no ambiente, precisou consultar o manual da *GoDonnie*. Foi diretamente na “Seção 7: comandos de visualização do ambiente”, encontrou o comando ESPIAR e o executou. Para identificar quantos objetos havia de cada cor, utilizou o comando COR, não sendo necessário consultar o manual. Para montar o mapa tátil, o participante preferiu utilizar repetidamente a lupa para relembrar o cenário virtual ao invés de usar o comando ESPIAR.

Os comandos utilizados podem ser verificados na Figura 8.9a, e a Figura 8.9b representa o cenário com a posição atual do robô.



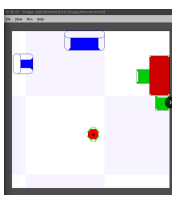
(a) comandos utilizados pelo P2



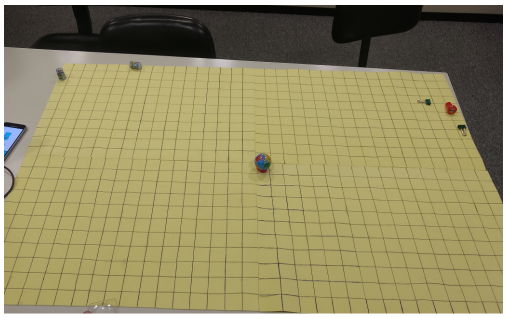
(b) cenário com a posição atual do robô

Figura 8.9: Atividade 1 - Resposta P2
Fonte: a autora

A Figura 8.10a representa o cenário com a posição atual do robô e a Figura 8.10b apresenta o mapa tátil que o participante montou. A Figura 8.11 apresenta o mapa tátil montado pelo participante visto de outro ângulo. Pode-se verificar que o usuário conseguiu montar um mapa próximo ao cenário real. Dessa forma, demonstra que conseguiu realizar um mapa mental do ambiente por meio da interface gráfica do ambiente.



(a) Cenário virtual



(b) Mapa tátil

Figura 8.10: Mapa tátil montado pelo P2
Fonte: a autora

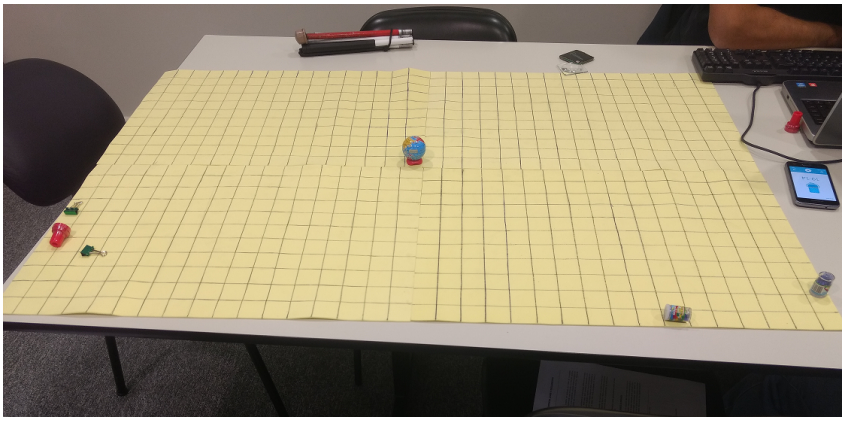
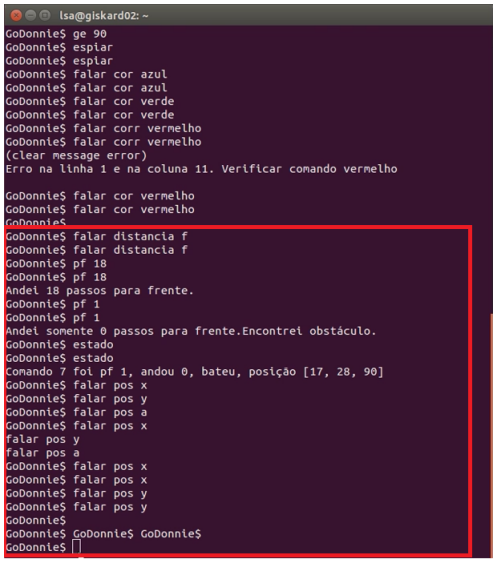
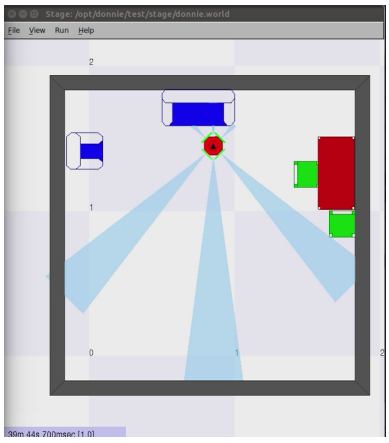


Figura 8.11: Mapa tátil montado por P2 visto de outro ângulo
Fonte: a autora

Na atividade 2, o participante P2 lembrou que devia utilizar o comando DISTÂNCIA, mas não lembrava do argumento. Por este motivo, consultou o manual e então executou o comando junto com o comando FALAR. Para identificar qual a posição do robô no eixo X e Y e o ângulo, o participante, primeiramente, utilizou o comando ESTADO. Após, pesquisou no manual e encontrou os comandos POS X, POS Y e POS A, que foram executados na *GoDonnie*. A Figura 8.12a representa os comandos utilizados pelo P2, enquanto a Figura 8.12b representa o cenário com a posição atual do robô.



(a) comandos utilizados pelo P2

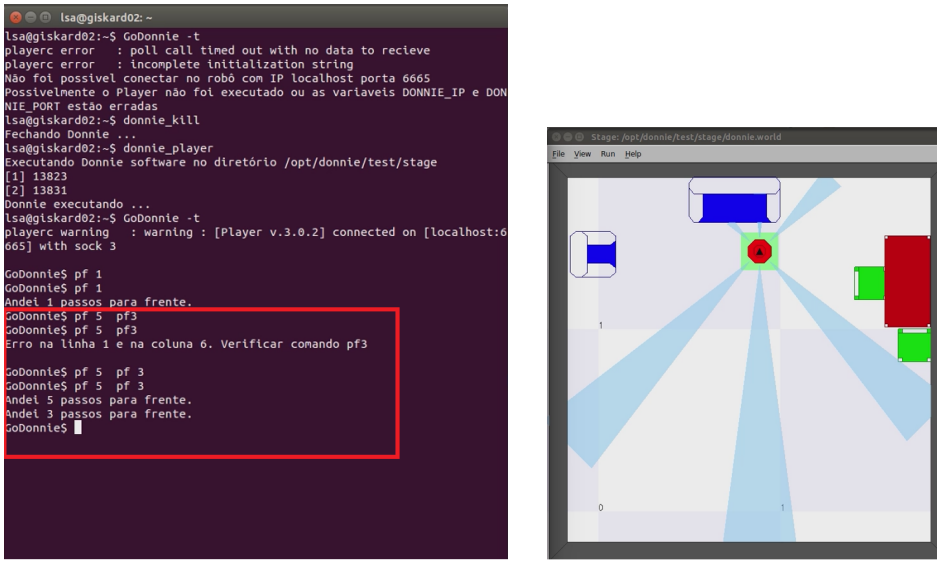


(b) cenário com a posição atual do robô

Figura 8.12: Atividade 2 - Resposta P2
Fonte: a autora

Para atividade 3, que solicitava que fossem relidas as explicações dos comandos utilizados nas atividades anteriores, o participante consultou rapidamente o manual e informou que estavam adequadas. Salientou que os exemplos são fundamentais. Destaca-se que esse participante, ao contrário do participante P1, não releu cada parte do manual. Considera-se que essa postura foi adotada porque ele já havia consultado o manual para realizar as atividades anteriores.

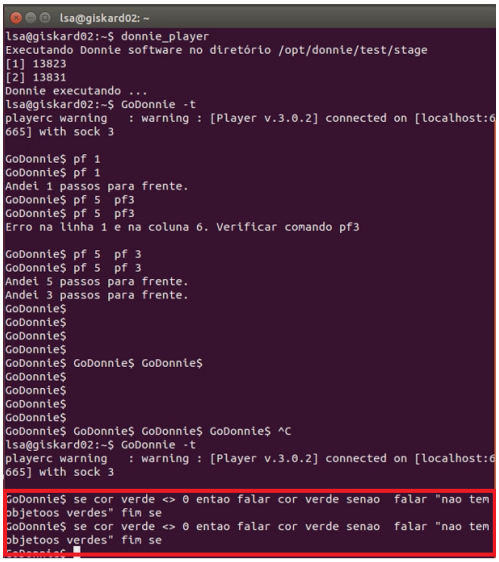
Na atividade 4, os avaliadores afastaram um pouco o robô do objeto sofá do cenário. O participante P2 também identificou que havia erro de digitação da sequência de comandos PF 5 PF3, antes de digitá-los. Após a mensagem do sistema, o usuário corrigiu o erro com facilidade. A Figura 8.13a representa os comandos utilizados pelo P2, enquanto a Figura 8.13b representa o cenário com a posição atual do robô.



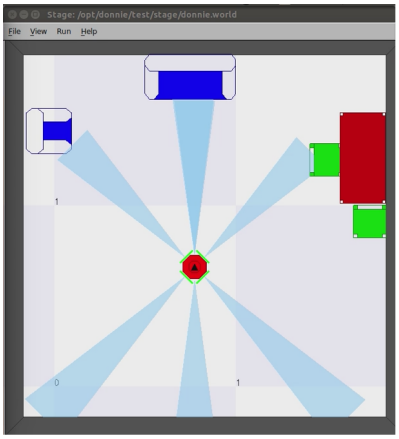
(a) comandos utilizados pelo P2 (b) cenário com a posição atual do robô

Figura 8.13: Atividade 4 - Resposta P2
Fonte: a autora

Na atividade 5, os avaliadores afastaram um pouco o robô dos objetos do cenário. Era esperado que o participante utilizasse o comando SE, para dizer se existia ou não objeto de cor verde. O participante P2, primeiramente, questionou se poderia usar o comando ESPIAR. Após, verificou que necessitava utilizar o comando SE, mas, ao não lembrar da sintaxe, consultou o manual. Não teve dúvidas com relação à lógica do comando. Somente a forma como escrevê-lo. Também consultou o comando COR para confirmar como utilizar dentro do comando SE. A Figura 8.14a representa os comandos utilizados pelo P2, enquanto a Figura 8.14b representa o cenário com a posição atual do robô.



(a) comandos utilizados pelo P2

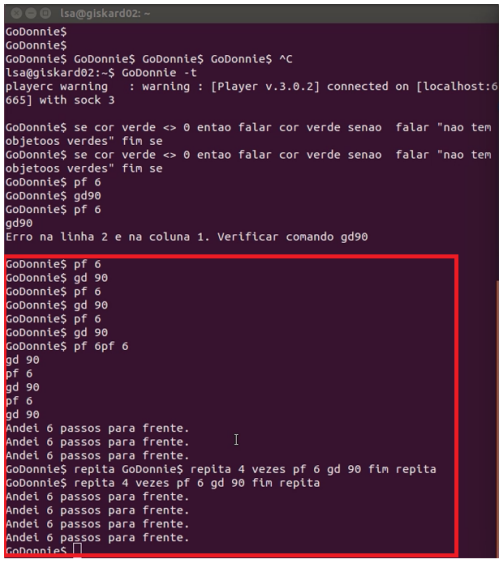


(b) cenário com a posição atual do robô

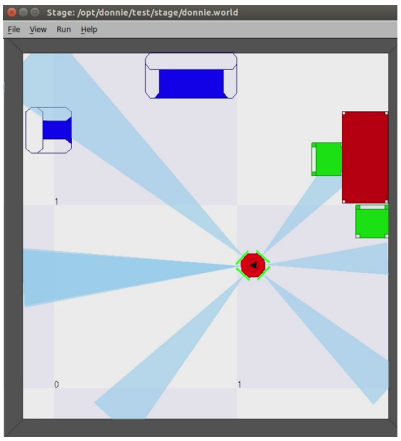
Figura 8.14: Atividade 5 - Resposta P2

Fonte: a autora

Na atividade 6, foi solicitado que o participante P2 fizesse um quadrado de tamanho de lado 6. O participante utilizou uma sequência dos comandos PF e GD ao invés de utilizar o comando REPITA, como fez o participante P1 que não tinha experiência em programação. Após, ao ser questionado se existia outra forma para fazer o quadrado, comentou que poderiam ser utilizados comandos de repetição. Consultou no manual como utilizar o comando REPITA e refez o quadrado com esse comando. A Figura 8.15a representa os comandos utilizados pelo P2, enquanto a Figura 8.15 representa o cenário com a posição atual do robô.



(a) comandos utilizados pelo P2



(b) cenário com a posição atual do robô

Figura 8.15: Atividade 6 - Resposta P2

Fonte: a autora

Conforme já descrito, havia aplicação de partes do questionário intercalada à realização das atividades. Segue uma análise das respostas ao questionário.

8.2.1 Facilidade de uso

O participante P2 concordou totalmente que a sintaxe e semântica da *GoDonnie*, e que a interface sonora e o cenário virtual são de fácil entendimento. Além disso, concordou totalmente que é fácil realizar as atividades no ambiente. Também concordou que os comandos da *GoDonnie* permitem resolver um problema de diferentes formas. Quanto às mensagens sonoras, o participante concordou totalmente que são de fácil entendimento.

8.2.2 Utilidade

O participante P2 concordou totalmente que a linguagem pode ser utilizada para resolver diferentes atividades. Além disso, concordou totalmente que o uso da *GoDonnie* permite compreender objetos em um ambiente virtual e compreender a relação desses objetos com esse ambiente. Para este participante (que possui baixa visão) havia questões relacionadas à interface gráfica. O participante concordou totalmente que o módulo gráfico do ambiente auxilia no entendimento desse ambiente e sobre onde está o robô. Sugeriu que fosse informado o tamanho dos obstáculos.

8.2.3 Prevenção e Tratamento de erros

O participante P2 discordou parcialmente que a *GoDonnie* evita que erros aconteçam, pois informou que o ambiente avisa somente sobre o erro depois de cometido. Por outro lado, considera que concorda totalmente que as regras de prevenção de erro, como a de interrupção de deslocamento quando pode haver uma colisão, são adequadas. O participante concordou totalmente que a *GoDonnie* permite que o usuário compreenda quando ocorre erro. Além disso, concordou parcialmente que a *GoDonnie* permite que o usuário compreenda o tipo de erro que ocorreu e que corrija, de fato, o erro. No entanto, salientou que é informado onde é o erro, mas a compreensão depende do conhecimento sobre a linguagem. Ainda, concordou totalmente que a *GoDonnie* permite que o usuário corrija os erros.

8.2.4 Ajuda e Documentação

Quanto ao manual da *GoDonnie*, o participante concordou totalmente que o mesmo é adequado ao aprendizado da linguagem e que os exemplos são úteis para o aprendizado dos comandos. Salientou que seria interessante ter mais exemplos no manual, conforme já destacado pelos participantes anteriores. Além disso, concordou totalmente que o manual é de fácil leitura. O participante não concordou e nem discordou que o manual possui uma ordem lógica de apresentação, mas comentou que está bem estruturado. Justificou, salientando que a ordem lógica vai depender do tipo de usuário que for utilizar esse material.

8.2.5 Satisfação de uso

O participante P2 concordou totalmente que é agradável utilizar a *GoDonnie* para aprender programação, destacando o uso dos sons icônicos. Também concordou totalmente que o uso da *GoDonnie* é interessante para compreender um espaço real, e sugeriu a realização de um curso de programação, além da criação de cenários que pudessem representar ambientes reais, como uma sala de aula, por exemplo. Por fim, concordou totalmente que o uso da *GoDonnie* é prazeroso.

8.2.6 Interface sonora

Quanto à interface sonora, o participante P2 concordou totalmente que os sons icônicos (passos, giros, colisão) e as mensagens faladas são intuitivas. Concordou parcialmente que os sons icônicos são necessários para compreensão da execução dos comandos da *GoDonnie*, destacando que são um complemento. Quanto às mensagens faladas, o participante concorda totalmente que são necessárias para compreensão da execução dos comandos da *GoDonnie*, indo além dos sons icônicos. Posteriormente, o participante concordou totalmente que as mensagens icônicas auxiliam o entendimento das mensagens faladas, assim como as mensagens faladas são necessárias para um melhor entendimento dos sons icônicos. Também concordou totalmente que os sons icônicos e as mensagens faladas são agradáveis. Quanto à velocidade das mensagens faladas e sons icônicos, o participante concordou totalmente que estão adequados, mas sugeriu que a velocidade fosse configurada de acordo com a velocidade do *Orca*, pois usuários mais avançados no uso de leitores de tela podem preferir dessa forma.

8.2.7 Orientação e Mobilidade

O participante P2 concordou totalmente que a *GoDonnie* é um recurso que pode ser utilizado previamente pelo usuário para explorar um ambiente. Ainda, concordou totalmente que a *GoDonnie* permite compreender a relação dos objetos com o robô, onde está o robô no ambiente e a direção do robô. Além disso, o P2 concordou totalmente que a *GoDonnie* permite que o usuário externalize seu plano de ação mental para deslocar o robô de um ponto a outro do ambiente.

8.2.8 Programação

O participante P2 concordou totalmente que a *GoDonnie* é adequada para ser utilizada por pessoas com DV. Também concordou totalmente que a *GoDonnie* é adequada para o ensino de programação para iniciantes em programação e para programação de robô. Além disso, concordou totalmente que a *GoDonnie* não requer muitas ações para realização de problemas simples, assim como permite diferentes formas de resolução de um mesmo problema.

8.2.9 Decisões de design

Salientou-se para o usuário as decisões de design tomadas, que podem trazer limitações ao uso da *GoDonnie*. Essas estão centradas em duas decisões principais:

- Utilizar somente dados inteiros
- Não possuir um comando para entrada de dados via teclado (*input*)

Para avaliar essas questões junto aos usuários finais, foram elaboradas duas questões semiabertas, que são discutidas na sequência.

A primeira questão foi sobre a escolha de design de manipular dados somente do tipo números inteiros, não sendo possível trabalhar com números com vírgula e nem com letras. O P2 selecionou intermediário, pois acredita que poderia ser adicionado o tipo alfabético. Destacou que o tipo *Float* não seria necessário, pois seria complicado andar meio passo. O uso de caracteres do tipo alfabético também foi destacado pelos professores Prof2, Prof5 e Prof6, não tendo sido considerado pelo participante P1, que é cego.

A segunda questão foi sobre a entrada de dados que é realizada pela captura de informações pelo robô, diferente de outras linguagens de programação em que o usuário pode informar dados de entrada. O participante P2 indicou que tal questão seria de baixa prioridade, pois acredita que a *GoDonnie* não necessita da entrada de dados via teclado. Os professores Prof1, Prof2, Prof4, Prof5, Prof6 e Prof7 consideraram que a entrada de dados via teclado poderia ser um acréscimo importante à *GoDonnie*, como também foi destacado pelo participante P1.

8.2.10 Questões gerais

Para avaliar a *GoDonnie* de forma geral, foram elaboradas quatro questões abertas, que são discutidas a seguir:

Na primeira pergunta foi questionado: "A programação de robôs vem sendo utilizada como estratégia para a aprendizagem de programação por usuários iniciantes. Você considera que a *GoDonnie* pode ser utilizada por pessoas com deficiência visual que sejam iniciantes em programação?". O participante P2 destacou: *Sim, pode porque é fácil. Os comandos são simples, coisas fáceis de pegar para questão de aprendizado de conceitos. Acho bem válido, bem interessante. Não precisa do aprendizado da linguagem em si, só comando natural, para frente, para trás, repete n vezes, é fácil.* Esse depoimento indica que, mesmo o participante tendo consultado o manual para ajudar a relembrar os comandos, a linguagem seria de fácil aprendizado.

Na segunda pergunta foi questionado: "Você recomendaria o uso da *GoDonnie* para outras pessoas que desejam aprender a programar robô?". O participante P2 destacou que: *Sim, porque é fácil. Pelas facilidades dos comandos fica fácil e rápido de aprender a movimentação do robô e a explorar o espaço.*

Na terceira pergunta foi questionado: "Você gostaria de continuar utilizando a *GoDonnie*?"O participante P2 destacou que: *Sim, mas eu queria ver o robô <físico> andando. Até pra estruturar mais a linguagem, jeito de programar e tentar fazer uns exercício mais reais, tipo aquele que eu te comentei antes, de mapear o ambiente para pessoas DV ou baixa visão (...). Por exemplo, mapear essa sala mesmo, que nós estamos. Seria um programa interessante, a exploração do ambiente guiado assim, por alguém vidente, seria interessante.*

Na quarta pergunta foi questionado: "Se desejar, faça outros comentários sobre a *GoDonnie*". Como comentário geral, o participante acrescentou que: *Acho que tu conseguiu mapear tudo direitinho, e é bem fácil de usar. Essa é grande vantagem, ainda mais para usuário iniciante, <que requer> a aquisição do conceito de programação em si, sem ter muitos comandos diferentes da linguagem natural.*

8.3 Discussão dos resultados

Os resultados encontrados na presente avaliação sugerem que o manual da *GoDonnie* deve conter mais exemplos de utilização dos comandos (P1), e o comando COR deve ser alterado para informar a localização (P1) e tamanho dos objetos (P2). Também foi identificada a necessidade de ser incluído o tipo de dado alfabético (P2) e a entrada de dados via teclado (P1 e P2) na *GoDonnie*. Além disso, houve uma preferência do participante P1 pela voz do *TTS* que é da *Google*. Houve a sugestão por parte do participante P2, que a velocidade das mensagens faladas pelo *TTS* seja configurada de acordo com a utilizada no *Orca*. O P2 também sugeriu a realização de um curso de programação para pessoas com DV com a linguagem *GoDonnie*. Nesta avaliação ficou evidente a importância do manual da linguagem de programação para auxílio à realização de atividades de programação, assim como do mapa tátil e do instrumento de conferir graus, que foram utilizados para melhor compreender o cenário virtual. Além disso, por meio dessa avaliação, foi verificado que os usuários com DV, participantes da pesquisa, conseguiram criar mapas mentais do cenário virtual por meio da linguagem de programação, e do uso do módulo gráfico, o que indicou que foi possível compreender um espaço por meio da linguagem. Ainda, conforme [47], as habilidades de O&M estão relacionadas a cinco fases. No ambiente de programação *Donnie*, essas podem ser, resumidamente, assim percebidas:

- Percepção: foi possível demonstrar que os usuários conseguiram capturar informações sobre o robô e cenário por meio de comandos como ESPIAR, POS, DISTÂNCIA e COR.
- Análise: foi possível verificar que o usuário conseguiu perceber o cenário e onde está o robô com a criação do mapa tátil.
- Seleção: os usuários conseguiram identificar quais comandos seriam necessários para a realização das atividades de programação.
- Planejamento: os usuários conseguiram elaborar o programa reunindo os comandos para que o robô pudesse se deslocar.
- Execução: os usuários conseguiram executar o programa e compreender se o robô alcançou o objetivo previamente definido.

9. CONSIDERAÇÕES FINAIS

Esta pesquisa teve como objetivo geral definir e avaliar uma linguagem de programação para estimular as habilidades de orientação e mobilidade por pessoas com deficiência visual a partir do uso de programação de robô. Para tal, foi definida uma linguagem de programação chamada *GoDonnie* que foi baseada na linguagem Logo. A linguagem é executada no ambiente de programação *Donnie*, que contém um cenário com robô virtual.

Acredita-se que o objetivo geral delimitado nesta dissertação foi alcançado por meio dos objetivos específicos, conforme a descrição que segue:

- **Elaborar uma linguagem de programação para comandar um robô que possa ser utilizada por usuários que possuem DV na resolução de problemas de lógica e de O&M:** o Capítulo 5 descreve a linguagem de programação *GoDonnie*, que é detalhada no Apêndice A. A linguagem possui comandos que permitem a resolução de problemas de lógica, assim como de O&M. Desta forma, existem comandos para percepção de onde está o robô (POS, ESTADO), para a localização dos objetos (ESPIAR), para identificação da quantidade de objetos por cor (COR) e para a verificação da distância entre os objetos e o robô (DISTÂNCIA). A partir disso, o usuário pode criar um plano de ação para ir até o local desejado utilizando comandos de movimentação (PF e PT) e rotação (GD e GE). Existem ainda comandos comuns a outras linguagens que permitem a resolução de problemas mais sofisticados como CRIAR, SE, REPITA, ENQUANTO, PARA e APRENDER. Todos os comandos fornecem *feedbacks* sonoros para que o usuário compreenda a execução. Além disso, existe ainda o comando FALAR que pode ser usado em combinação com os demais comandos já citados para expressar uma informação de maneira sonora.
- **Avaliar a linguagem de programação que comanda o robô junto a pessoas que possuem deficiência visual, no que se refere:**
  - **à usabilidade:** no Capítulo 6 são apresentadas as avaliações realizadas utilizando a linguagem *GoDonnie* sem o ambiente de programação. Foram realizadas avaliações em 4 etapas:
    - a primeira etapa foi realizada com 4 pessoas videntes com experiência em programação, no qual foram validados o manual da *GoDonnie* e os tipos de atividades que seriam aplicados aos usuários com DV.
    - a segunda etapa foi realizada com uma pessoa cega sem conhecimento em programação, no qual foi verificado que a linguagem é fácil de aprender por programadores iniciantes e, conforme sugestões do participante, os materiais concretos foram adaptados para que se tornassem ainda mais fáceis entender.

    * a terceira etapa foi realizada com 2 participantes que possuem deficiência visual e tem experiência em programação, no qual foi verificado que a *GoDonnie* pode ser utilizada como recurso para resolução de problemas na área de programação de robô.
    * a quarta etapa foi realizada com 7 professores de programação, no qual foi verificado que os comandos da *GoDonnie* são práticos e de fácil utilização na programação, segundo os Critérios de Usabilidade: Consistência, Facilidade de Uso, Relação entre o sistema e o mundo real, e Ajuda e Documentação. Foram sugeridas algumas modificações no manual, como a inserção do contexto da *GoDonnie* e links de navegação, além de uma melhor organização dos comandos. Também foi sugerido a inclusão dos tipos de dados *Float*, alfabéticos e alfanuméricos. Outra sugestão foi a indicação da necessidade da inclusão da entrada de dados via teclado.

  Desta forma, sugere-se que o processo de avaliação de uma linguagem de programação que manipula um robô, seja virtual ou físico, inclua: materiais concretos para explicar conceitos de programação e de robô; atividades de programação guiadas e também com níveis de dificuldade como fácil, intermediário e difícil; manual da linguagem com links de navegação. É interessante a aplicação de questionários ao final das atividades, como os propostos neste trabalho.

- **ao apoio do desenvolvimento de habilidades de orientação e mobilidade:** no Capítulo 8 são descritas as avaliações realizadas com a *GoDonnie* no Ambiente de Programação *Donnie*, que foi realizada em duas etapas: uma com usuário cego sem conhecimento em programação (P1) e outra com um usuário com baixa visão com conhecimento em programação (P2). Como resultado, foi verificado que os comandos da *GoDonnie* possuem uma boa usabilidade com relação à facilidade de uso, utilidade, prevenção e tratamento de erros, e os usuários ficaram satisfeitos com o uso. Além disso, possui uma interface sonora agradável. Também foi verificado que o uso da *GoDonnie* auxilia no desenvolvimento de O&M. Deste modo, os participantes conseguiram fazer um mapa mental do cenário e dos objetos relacionados a ele. Esse mapa foi externalizado com o uso de um mapa tátil, em que os participantes distribuíram os objetos representando o cenário virtual.

- **Verificar como a programação de robôs pode ser utilizada como um recurso para apoiar a solução de problemas relacionados às habilidades de orientação e mobilidade de pessoas que possuem deficiência visual:** Com as avaliações realizadas, foram identificadas algumas sugestões de bom uso para o ambiente para que ele apoie a solução de problemas relacionados a O&M, conforme descritas a seguir:

- Utilizar um mapa tátil com um material fácil de apalpar, como por exemplo EVA, e que esse material seja marcado como uma "matriz"em que cada quadrinho represente um passo do robô. Isso é interessante para que sejam passados conceitos iniciais de programação e execução dos comandos, e para que o participante compreenda melhor o cenário.
- Utilizar um instrumento para conferência de ângulos. Isso é interessante para o caso de os participantes não estarem familiarizados com o uso de graus.
- Usar objetos com formas diferentes no mapa tátil para representar objetos diferentes do cenário virtual.
- Usar objetos com formas diferentes no cenário virtual para representar objetos diferentes.
- Representar os objetos diferentes com cores diferentes no cenário virtual.
- Criar o cenário virtual em que os objetos não estejam sobrepostos.
- Representar o robô ocupando um passo (quadro) no mapa tátil.
- Colocar uma marcação palpável para identificar a frente do robô.
- Usar objetos em que a pessoa possa tatear sem tira-los do lugar no mapa tátil.

Além disso, algumas dicas gerais para o bom uso do ambiente de programação foram identificadas:

- Oferecer fones de ouvido durante a realização de atividades de programação no ambiente.
- Fornecer o manual da linguagem em braile e/ou no computador de acordo com a preferência do usuário.
- Oferecer treinamento para os usuários que não tenham experiência prévia no uso de leitor de telas.
- Oferecer treinamento para os usuários que não tenham experiência prévia no uso de teclado.

Essa pesquisa contribuiu no sentido de definir uma linguagem de programação com usabilidade e acessibilidade para pessoas com DV. Além disso, propôs uma metodologia de avaliação para linguagens de programação baseadas em robótica quanto a facilidade de aprendizado e O&M, e também uma metodologia de avaliação para ambientes de programação com uso de robótica para desenvolvimento de programação e O&M. Todos os instrumentos utilizados encontram-se nos Apêndices deste trabalho. Por fim, foram disponibilizados o protocolo e o resultado da Revisão Sistemática da Literatura sobre ensino de programação com uso de robótica que pode ser utilizada como apoio a outros projetos.

9.1 Limitações

Entre as limitações do presente estudo, pode-se citar:

- Houve restrição no número de participantes com DV na avaliação junto ao ambiente de programação *Donnie*.
- Houve limitações de design do comando APRENDER, para que o mesmo não tivesse colchetes na chamada. Essa limitação deve-se à uma restrição no desenvolvimento que utiliza esse símbolo como delimitador entre a chamada do procedimento e a passagem de parâmetros.
- Houve problemas na detecção dos objetos (tamanho x cor) fazendo com que o comando ESPIAR e COR, por vezes, não trouxessem as informações exatas.
- Houve a falta do robô físico para que se pudesse verificar o impacto motivacional desse recurso.
- Houve o uso do *Linux* e do leitor de telas *Orca*. A saber, o ambiente *Windows* e leitores de telas como *NVDA* são mais utilizados e eram os ambientes de uso dos participantes com DV.

9.2 Trabalhos futuros

Identificou-se os seguintes trabalhos futuros advindos das avaliações realizadas e de algumas limitações detectadas:

- Aprimorar a linguagem com a inserção/revisão de:
  - entrada de dados via teclado (*input*) e dos tipos de dados *Float*, alfabético e alfanumérico na *GoDonnie*.
  - funcionalidade e atributos do comando COR, conforme sinalizado na avaliação com participante cego sem conhecimento em programação (Capítulo 8).
  - alterar o comando APRENDER para que este não tenha colchetes na chamada do comando.
- Buscar solução para:
  - o som de colisão fornecido pelo ambiente para que esse seja mais intuitivo.
  - as diferenças de qualidade de som fornecida via *TTS* da *Google* e dos leitores de telas (*Orca* e NVDA).

  - a velocidade das mensagens faladas pelo *TTS*, para que essas sejam configuradas de acordo com a utilizada no leitor de telas.
  - a possibilidade de o ambiente de programação *GoDonnie* também poder ser utilizado no sistema operacional *Windows*.

- Realizar a avaliação do Robô e do cenário físico junto ao usuários com DV, verificando se esse pode ser um fator motivacional para o uso do ambiente.
- Realizar um curso de programação com o uso do ambiente de programação *Donnie*, explorando o uso de práticas de programação colaborativa como em duplas e grupos.

9.3 Lições aprendidas

Muitas foram as lições aprendidas neste trabalho e, dentre elas, a principal foi projetar uma linguagem de programação com foco na O&M para pessoas com DV. Para este objetivo, foi visto a importância de entender as necessidades do público-alvo e compreender como as diferentes interfaces podem afetar a esses usuários. Além disso, compreender quais materiais concretos podem ajudar as pessoas com DV a entender melhor ambientes. Foi visto a importância de transmitir as informações do ambiente por meio de *feedback*, esses sendo sons icônicos e mensagens faladas, um não excluindo o outro.

Identificou-se nas avaliações a importância de realizar diferentes tipos de atividades para compreender o poder que a linguagem tem, para entender o que a linguagem atende e, caso contrário, ver o que é importante que atenda. Destaca-se, também, o processo de elaboração e condução da avaliação com o usuário final, pois, por mais que existisse um conhecimento prévio a esta pesquisa, sempre existem desafios e aprendizados ao trabalhar com usuários com necessidades especiais.

Ressalta-se, ainda, a importância de se trabalhar com ambiente de programação que pode carregar diferentes cenários e de a linguagem permitir ser continuada. Além disso, realizar a construção de referencial teórico baseado em RSL é um trabalho criterioso, que demanda mais tempo de dedicação, mas que torna o trabalho mais rico, pois é possível identificar todas as evidências disponíveis sobre o tema.

Outra lição inerente a este trabalho foi gerir um grupo interdisciplinar, que possuía diferentes equipes com agendas próprias. Neste sentido, ressalta-se a importância de compreender todas as partes do projeto e, de certa forma, motivar a equipe para que o objetivo final seja alcançado. Além disso, destaca-se a importância de sempre realizar testes no ambiente antes da avaliação com os usuários. Desta forma, a versão testada com os usuários será mais estável.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Aho, A. V. “Ubiquity symposium: Computation and computational thinking”, *Ubiquity*, vol. 2011–January, Jan 2011, pp. 1–8.

[2] Al-Ratta, N. M.; Al-Khalifa, H. S. “Teaching programming for blinds: A review”. In: Fourth International Conference on Information and Communication Technology and Accessibility, 2013, pp. 1–5.

[3] Altadmri, A.; Brown, N. C. “37 million compilations: Investigating novice programming mistakes in large-scale student data”. In: 46th ACM Technical Symposium on Computer Science Education, 2015, pp. 522–527.

[4] Barbosa, S.; Silva, B. “Interação humano-computador”. Elsevier Brasil, 2010, 406p.

[5] Barros, R. P.; Burlamaqui, A. M. F.; de Azevedo, S. O.; de Lima Sa, S. T.; Goncalves, L. M. G.; da Silva, A. A. R. S.; et al.. “Cardbot-assistive technology for visually impaired in educational robotics: Experiments and results”, *IEEE Latin America Transactions*, vol. 15–3, Mar 2017, pp. 517–527.

[6] Barros, R. P.; Torres, V. P.; Burlamaqui, A. M. F.; Natal, R. “Cardbot: Tecnologias assistivas para imersao de deficientes visuais na robótica educacional”. In: V Workshop de Robótica Educacional, 2014, pp. 11–16.

[7] Benitti, F. B. V.; Vahldick, A.; Urban, D. L.; Krueger, M. L.; Halma, A. “Experimentação com robótica educativa no ensino médio: ambiente, atividades e resultados”. In: XV Workshop de Informática na Escola, 2009, pp. 1811–1820.

[8] Benyon, D. “Designing interactive systems: a comprehensive guide to HCI and interaction design”. Person education limited, 2010, 678p.

[9] Berland, M.; Lee, V. “Collaborative strategic board games as a site for distributed computational thinking”, *International Journal of Game-Based Learning*, vol. 1–2, 2011, pp. 65–81.

[10] Brereton, P.; Kitchenham, B. A.; Budgen, D.; Turner, M.; Khalil, M. “Lessons from applying the systematic literature review process within the software engineering domain”, *Journal of Systems and Software*, vol. 80–4, Abr 2007, pp. 571–583.

[11] Cambruzzi, E.; de Souza, R. M. “Robótica educativa na aprendizagem de lógica de programação: Aplicação e análise.” In: XXI Workshop de Informática na Escola, 2015, pp. 21–28.

[12] Campos, M. d. B.; Sanchez, J.; Oliveira, J. D.; Inácio, T. “Usability evaluation of a mobile navigation application for blind users”. In: International Conference on Universal Access in Human-Computer Interaction, 2015, pp. 117–128.

[13] Carvalho, A. d. L.; Ponce de Leon, F. d.; et al.. “Grandes desafios da pesquisa em computação no brasil–2006–2016”, *Sociedade Brasileira de Computação*, Maio 2006.

[14] da Silva, C. F.; Ferreira, S. B. L.; Ramos, J. a. F. M. “Whatsapp accessibility from the perspective of visually impaired people”. In: 15th Brazilian Symposium on Human Factors in Computer Systems, 2016.

[15] de Almeida, L. C. F.; da Silva, J. S. D. M.; do Amaral, H. J. C. “Robótica educacional: Uma possibilidade para o ensino e aprendizagem”, *Revista da Escola Regional de Informática*, vol. 2–2, Nov 2013, pp. 178–184.

[16] Demo, P. “Educar pela pesquisa”. Autores Associados LTDA, 2011, 121p.

[17] Dutra, C. P. “Orientação e Mobilidade: Conhecimentos básicos para a inclusão do deficiente visual”. MEC/SEESP, 2003, 167p.

[18] Engers, M. E. A. “O professor alfabetizador eficaz: análise de fatores influentes da eficácia do ensino”, Tese de Doutorado, Universidade Federal do Rio Grande do Sul, 1987, 291p.

[19] Farias, H.; Bonifácio, B.; Ferreira, R. “Avaliando o uso da ferramenta scratch para ensino de programação através de análise quantitativa e qualitativa”. In: Simpósio Brasileiro de Informática na Educação, 2015, pp. 947–956.

[20] Fonseca, J. J. S. “Metodologia da pesquisa científica”, 2002, (Apostila).

[21] Garcia, J. C. D.; Galvão Filho, T. A. “Pesquisa nacional de tecnologia assistiva”, *Instituto de Tecnologia Social Brasil*, Fev 2012, pp. 8–66.

[22] Gil, A. C. “Como elaborar projetos de pesquisa”. Editora Atlas S.A, 2002, 173p.

[23] Green, T.; Petre, M. “Usability analysis of visual programming environments: A ‘cognitive dimensions’ framework”, *Journal of Visual Languages Computing*, vol. 7–2, Jun 1996, pp. 131 – 174.

[24] Green, T. R. G. “Cognitive dimensions of notations”. In: V People and Computers, 1989, pp. 443–460.

[25] Hoc, J.-M.; Nguyen-Xuan, A. “Chapter 2.3 - language semantics, mental models and analogy”. In: *Psychology of Programming*, Academic Press, 1990, pp. 139 – 156.

[26] Howard, A. M.; Park, C. H.; Remy, S. “Using haptic and auditory interaction tools to engage students with visual impairments in robot programming activities”, *IEEE Transactions on Learning Technologies*, vol. 5–1, Dez 2012, pp. 87–95.

[27] Jacobson, W. H. “The art and science of teaching orientation and mobility to persons with visual impairments”. American Foundation for the Blind, 1993, 200p.

[28] Jašková, L.; Kaliaková, M. “Programming microworlds for visually impaired pupils”. In: 3rd International Constructionism Conference, 2014, pp. 1–10.

[29] Júnior, N. M. F.; Vasques, C. K.; Francisco, T. H. A. “Robótica educacional e a produção científica na base de dados da capes”, *Revista Electrónica de Investigación y Docencia (REID)*, vol. 35, Jul 2010, pp. 53.

[30] Kakehashi, S.; Motoyoshi, T.; Koyanagi, K.; Ohshima, T.; Kawakami, H. “P-cube: Block type programming tool for visual impairments”. In: Conference on Technologies and Applications of Artificial Intelligence, 2013, pp. 294–299.

[31] Kakehashi, S.; Motoyoshi, T.; Koyanagi, K.; Oshima, T.; Masuta, H.; Kawakami, H. “Improvement of p-cube: Algorithm education tool for visually impaired persons”. In: IEEE Symposium on Robotic Intelligence In Informationally Structured Space, 2014, pp. 1–6.

[32] Kane, S. K.; Bigham, J. P. “Tracking stemxcomet: teaching programming to blind students via 3d printing, crisis management, and twitter”. In: ACM Technical Symposium on Computer Science Education, 2014, pp. 247–252.

[33] Kazimoglu, C.; Kiernan, M.; Bacon, L.; Mackinnon, L. “A serious game for developing computational thinking and learning introductory computer programming”, *Procedia-Social and Behavioral Sciences*, vol. 47, Ago 2012, pp. 1991–1999.

[34] Kiernan, M.; Kazimoglu, C.; Bacon, L.; Mackinnon, L. “Developing an educational game to support cognitive learning”, *Compass: Journal of Learning and Teaching*, vol. 5–9, Jun 2014.

[35] Kjeldskov, J.; Skov, M. B.; Stage, J. “Instant data analysis: Conducting usability evaluations in a day”. In: Third Nordic Conference on Human-computer Interaction, 2004, pp. 233–240.

[36] Kurtev, S.; Christensen, T. A.; Thomsen, B. “Discount method for programming language evaluation”. In: 7th International Workshop on Evaluation and Usability of Programming Languages and Tools, 2016, pp. 1–8.

[37] Lahav, O.; Mioduser, D. “A blind person’s cognitive mapping of new spaces using a haptic virtual environment”, *Journal of Research in Special Educational Needs*, vol. 3–3, Nov 2003, pp. 172–177.

[38] Lahav, O.; Schloerb, D. W.; Kumar, S.; Srinivasan, M. A. “Blindaid: A learning environment for enabling people who are blind to explore and navigate through unknown real spaces”. In: Virtual Rehabilitation, 2008, pp. 193–197.

[39] Lazar, J.; Feng, J. H.; Hochheiser, H. “Research methods in human-computer interaction”. Morgan Kaufmann, 2010, 419p.

[40] Ludi, S.; Abadi, M.; Fujiki, Y.; Sankaran, P.; Herzberg, S. “Jbrick: Accessible lego mindstorm programming tool for users who are visually impaired”. In: 12th International ACM SIGACCESS Conference on Computers and Accessibility, 2010, pp. 271–272.

[41] Ludi, S.; Reichlmayr, T. “The use of robotics to promote computing to pre-college students with visual impairments”, *Trans. Comput. Educ.*, vol. 11–3, Out 2011, pp. 20:1–20:20.

[42] Ludi, S. A.; Reichlmayr, T. “Developing inclusive outreach activities for students with visual impairments”. In: 39th SIGCSE Technical Symposium on Computer Science Education, 2008, pp. 439–443.

[43] Ludi, S. L.; Ellis, L.; Jordan, S. “An accessible robotics programming environment for visually impaired users”. In: 16th International ACM SIGACCESS Conference on Computers and Accessibility, 2014, pp. 237–238.

[44] Machado, E. V.; et al.. “Orientação e mobilidade: conhecimentos básicos para a inclusão do deficiente visual”. Brasília: MEC, SEESP, 2003.

[45] Marques, A. M. d. M. “Utilização pedagógica de mapas mentais e de mapas conceptuais”, Dissertação de Mestrado, Universidade Aberta, Lisboa, Portugal, 2008, 153p.

[46] Massi, I.; et al.. “Orientação e mobilidade: conhecimentos básicos para a inclusão do deficiente visual”. Brasília: MEC, SEESP, 2003.

[47] Mazzaro, J. L.; et al.. “Orientação e mobilidade: conhecimentos básicos para a inclusão do deficiente visual”. Brasília: MEC, SEESP, 2003.

[48] Mealin, S.; Murphy-Hill, E. “An exploratory study of blind software developers”. In: IEEE Symposium on Visual Languages and Human-Centric Computing, 2012, pp. 71–74.

[49] Meira, M. C.; Lima, M. S. S.; Borges, M. A. F. “Torneios baseados em robocode para incentivar jovens a aprender programação”. In: XXXVI Congresso da Sociedade Brasileira de Computação, 2016, pp. 2403–2412.

[50] Melo, A. M.; Baranauskas, M. C. C.; Bonilha, F. F. G. “Avaliação de acessibilidade na web com a participação do usuário: um estudo de caso”. In: VI Simpósio sobre Fatores Humanos em Sistema Computacionais, 2004, pp. 17–20.

[51] Motoyoshi, T.; Kakehashi, S.; Masuta, H.; Koyanagi, K.; Oshima, T.; Kawakami, H. “The usefulness of p-cube as a programming education tool for programming beginners”. In: 24th IEEE International Symposium on Robot and Human Interactive Communication, 2015, pp. 297–300.

[52] Nielsen, J. “Usability inspection methods”. In: Conference Companion on Human Factors in Computing Systems, 1994, pp. 413–414.

[53] Oliveira, J. D.; de Borba C., M.; de Morais A., A.; Harb M., I. “Teaching robot programming activities for visually impaired students: A systematic review”. In: International Conference on Universal Access in Human-Computer Interaction, 2017, pp. 155–167.

[54] Pane, J. F. “A programming system for children that is designed for usability”, Tese de Doutorado, Pittsburgh, PA, USA, 2002, 415p.

[55] Papert, S. “The children’s machine: Rethinking School In The Age Of The Computer”. Basic Books, 1994, 256p.

[56] Park, C. H.; Howard, A. “Engaging students with visual impairments in engineering and computer science through robotic game programming (research-to-practice)”. In: ASEE Annual Conference and Exposition, 2013, pp. 1–14.

[57] Prates, R. O.; de Souza, C. S.; Barbosa, S. D. “Methods and tools: a method for evaluating the communicability of user interfaces”, *Interactions*, vol. 7–1, Jan 2000, pp. 31–38.

[58] Rogers, Y.; Sharp, H.; Preece, J. “Design de interação: além da interação humano-computador”. Bookman, 2013, 569p.

[59] Sadowski, C.; Kurniawan, S. “Heuristic evaluation of programming language features: Two parallel programming case studies”. In: 3rd ACM SIGPLAN Workshop on Evaluation and Usability of Programming Languages and Tools, 2011, pp. 9–14.

[60] Sampieri, R. H.; Collado, C. F.; Lucio, P. B.; Pérez, M. d. l. L. C. “Metodología de la investigación”. McGraw-hill México, 1998, 599p.

[61] Sánchez, J.; Alarcón, P. “dmc: Una herramienta digital para aprender con mapas conceptuales”. In: IX Taller Internacional de Software Educativo, 2004, pp. 102–111.

[62] Sánchez, J.; Alarcón, P. “Construcción de conocimiento con editores de mapas conceptuales”. In: Nuevas Ideas en Informática Educativa, 2005, pp. 202–211.

[63] Sánchez, J.; Flores, H. “Concept mapping for virtual rehabilitation and training of the blind”, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 18–2, Abr 2010, pp. 210–219.

[64] Solomon, C. J.; Papert, S. “A case study of a young child doing turtle graphics in logo”. In: National Computer Conference and Exposition, 1976, pp. 1049–1056.

[65] Stefik, A.; Siebert, S. “An empirical investigation into programming language syntax”, *ACM Transactions on Computing Education*, vol. 13–4, Nov 2013, pp. 19.

[66] Stevens, R. D. “Principles for the design of auditory interfaces to present complex information to blind people”, Tese de Doutorado, University of York, 1996, 286p.

[67] Takagi, H.; Saito, S.; Fukuda, K.; Asakawa, C. “Analysis of navigability of web applications for improving blind usability”, *ACM Transactions on Computer-Human Interaction*, vol. 14–3, Set 2007, pp. 13.

[68] Tran, D.; Haines, P.; Ma, W.; Sharma, D. “Text-to-speech technology-based programming tool”. In: 7th WSEAS International Conference on Signal, Speech and Image Processing, 2007, pp. 173–176.

[69] Vahldick, A.; Benitti, F. B. V.; Urban, D. L.; Krueger, M. L.; Halma, A. “O uso do lego mindstorms no apoio ao ensino de programação de computadores”. In: XV Workshop de Educação em Computação, 2009, pp. 523–526.

[70] Vahldick, A.; Mendes, A. J.; Marcelino, M. J.; Farah, P. R. “Pensamento computacional praticado com um jogo casual sério no ensino superior”. In: XXXVI Congresso da Sociedade Brasileira de Computação, 2016, pp. 2303–2312.

[71] Valente, J. A. “O papel do facilitador no ambiente logo”. In: *O papel do facilitador no ambiente Logo: formação e atuação*, UNICAMP/NIED, 1996, cap. 1, pp. 1–34.

[72] Valente, J. A. “Diferentes usos do computador na educação”, *Em aberto*, vol. 12–57, Mar 2008, pp. 1–14.

[73] Wing, J. M. “Computational thinking”, *Communications of the ACM*, vol. 49–3, Mar 2006, pp. 33–35.

APÊNDICE A – MANUAL DA LINGUAGEM *GODONNIE*

Esta versão do manual explica 23 comandos, que estão organizados em 12 seções.

Menu

1. Contexto da GoDonnie

A *GoDonnie* é uma linguagem de programação, que comanda um robô chamado *Donnie* por um cenário. Este robô funciona em um ambiente próprio. Para tal, estando em um terminal Linux, é necessário digitar **donnie_player** e pressionar a tecla ENTER. Após é necessário iniciar o módulo em que é adicionado os comandos da *GoDonnie*, para isso, deve-se digitar *GoDonnie –t*. Essas expressões podem ser digitadas em minúsculo ou maiúsculo.

No modo de linha de comando, após ter sido digitado *GoDonnie -t*, o usuário digita os comandos desejados. Pode ser digitado um comando por linha ou vários comandos em uma mesma linha. Após cada linha, o usuário deve pressionar a tecla ENTER. Para que o robô execute os comandos, é preciso pressionar a tecla ESC. Desta forma, o usuário pode digitar um comando e pressionar ENTER e ESC, ou escrever um comando por linha, pressionando a tecla ENTER para mudar de linha, e somente pressionar a tecla ESC quando desejar executar os comandos.

A *GoDonnie* não diferencia letras minúsculas de maiúsculas, assim como os comandos podem ser digitados com acentuação ou sem. O robô sempre parte da posição 0,0 virado para o norte. Esta posição fica no canto inferior à esquerda.

Seção 1: sair do terminal de programação
Comando: SAIR
Explicação: fecha o ambiente de programação. Só pode ser usado no terminal.
Exemplo: sair

[Próximo] [Seção Anterior][Menu][1.contexto da GoDonnie]
[2.sair do terminal de programação][3. declaração de variáveis]
[4. comandos de áudio][5. operadores]
[6. comandos de movimentação][7. comandos de rotação]
[8. comandos de visualização do ambiente][9. comandos de informações (ambiente)]
[10. comandos de condição][11. comandos de repetição]
[12. declaração de procedimentos][13. comandos variados][14. Exemplos de aplicação]

Seção 2: Declaração de variáveis
Variável é um objeto que guarda um valor. Essa variável só poderá receber valores inteiros.

Comando: CRIAR x
Argumentos: x corresponde a variável que será criada.
Explicação: as variáveis guardam somente os últimos valores recebidos.
As variáveis guardam somente valores inteiros. Desta forma, se houver um resultado com vírgula, esse será descartado e somente a parte inteira será armazenada na variável.
Existem 5 formas de criar uma variável:
1) criar uma variável sem um valor inicial e adicionar posteriormente o valor.
Exemplo: CRIAR A
Cria uma variável chamada A.
Tendo sido criada a variável, pode atribuir um valor diretamente.
Exemplo: A = 2
A variável chamada A armazena 2.
2) criar uma variável que recebe um valor inicial diretamente.
Exemplo: CRIAR B =5
Cria uma variável chamada B, que armazena o valor 5
3) criar uma variável que recebe uma expressão.
Exemplo: CRIAR C = A + B
Cria uma variável chamada C, que recebe o valor da variável A somado ao valor da variável chamada B. O resultado da variável C é 7.
Lembrar que ao adicionar um novo valor a uma variável já existente, irá sobrescrever o valor anterior.
C = 1
Altera o valor da variável C e armazena o valor 1, perdendo o valor anterior.
4) criar uma variável dentro do comando de repetição PARA (o comando para será visto na seção 10).
Exemplo: PARA CRIAR d = 0; d < 5; d = d + 1 FAÇA PF 1 FIM PARA
O robô se deslocará 5 passos para frente.
5) criar uma variável que receberá o valor de outro comando, como: COR (será visto na seção 8), DISTÂNCIA (será visto na seção 8) e POS (será visto na seção 8).
Exemplo: CRIAR d = DISTÂNCIA F
A variável d armazenará o valor da distância frontal do robô em relação ao objeto.
Exemplo: CRIAR c = COR VERDE
A variável c armazenará a quantidade de cores verdes.
Exemplo: CRIAR px = POS X
A variável px armazenará a posição atual do robô no eixo x.
Exemplo: F = 5
Retornará erro porque a variável F ainda não foi criada.

Existem regras para o nome das variáveis:
- Não há diferença entre letras maiúsculas e minúsculas. Desta forma, CRIAR A (maiúsculo) será o mesmo que CRIAR a (minúsculo).
- Não podem ter caracteres especiais. Exemplo: *, @, #, +
- Não podem iniciar com número. Exemplo: CRIAR 52abc está errado.

[Próximo] [Seção Anterior][Menu][1.contexto da GoDonnie]
[2.sair do terminal de programação][3. declaração de variáveis]
[4. comandos de áudio][5. operadores]
[6. comandos de movimentação][7. comandos de rotação]
[8. comandos de visualização do ambiente][9. comandos de informações (ambiente)]
[10. comandos de condição][11. comandos de repetição]

[12. declaração de procedimentos][13. comandos variados][14. Exemplos de aplicação]

Seção 3: Comandos de áudio
Comandos para manipulação e retorno de áudio.

Comando: FALAR "x"
Argumentos: x é uma palavra ou frase, que deve vir entre aspas duplas.
Explicação: fala a palavra ou frase contida em x, que deve vir entre aspas duplas.
Exemplo: FALAR "oi"
Será falado oi

Comando: FALAR x
Argumentos: x é uma variável que deve ter sido criada anteriormente.
Explicação: Fala o conteúdo da variável.
Exemplo: CRIAR a = 5
FALAR a
Será falado 5

Comando: SOM ligado ou **SOM** desligado
Explicação: Comando que liga ou desliga as mensagens faladas do sistema, somente os sons icônicos continuam executando.
Exemplo: SOM LIGAR
SOM DESLIGAR

[Próximo] [Seção Anterior][Menu][1.contexto da GoDonnie]
[2.sair do terminal de programação][3. declaração de variáveis]
[4. comandos de áudio][5. operadores]
[6. comandos de movimentação][7. comandos de rotação]
[8. comandos de visualização do ambiente][9. comandos de informações (ambiente)]
[10. comandos de condição][11. comandos de repetição]
[12. declaração de procedimentos][13. comandos variados][14. Exemplos de aplicação]

Seção 4 - Operadores
São operadores que fornecem suporte a expressões matemáticas e lógicas.

Operadores matemáticos
Explicação: operadores servem para comparar valores ou expressões.
+ soma
- subtração
* multiplicação
/ divisão
Operadores de comparação:
Explicação: operadores servem para comparar valores ou expressões.
<> diferente
== igual comparação
< menor
> maior
<= menor ou igual
>= maior ou igual
Operador de atribuição:
Explicação: Define ou redefine o valor armazenado em uma variável.
= atribuição

Exemplos:
Para realizar uma soma.

Criar a = 2
cria a variável chamada a e atribui o valor de 2.
Criar b = 1
Cria a variável chamada b e atribui o valor de 1.
Criar soma
Cria a variável chamada soma
soma = a + b
atribui a variavel soma o valor da soma da variável a e b.
Falar soma
Será falado: 3
Para realizar uma divisão.
Criar c = 2
cria a variável chamada c e atribui o valor de 2.
Criar d = 2
cria a variável chamada d e atribui o valor de 2.
Criar divisão
Cria a variável divisão
divisão = c / d
Atribui o valor da divisão dos conteúdos das variáveis c e d.
Falar divisão
Será falado: 1

[Próximo] [Seção Anterior][Menu][1.contexto da GoDonnie]
[2.sair do terminal de programação][3. declaração de variáveis]
[4. comandos de áudio][5. operadores]
[6. comandos de movimentação][7. comandos de rotação]
[8. comandos de visualização do ambiente][9. comandos de informações (ambiente)]
[10. comandos de condição][11. comandos de repetição]
[12. declaração de procedimentos][13. comandos variados][14. Exemplos de aplicação]

Seção 5: comandos de movimentação
São comandos que movimentam o robô no ambiente.

comando: PF n ou para frente n
Argumentos: n é o número de passos.
Explicação: o robô anda n passos para frente. Este comando aceita somente números inteiros, ou variáveis que armazenam números inteiros, ou expressões matemáticas que resultem em números inteiros.
Exemplo: PF 5
O robô andará 5 passos para frente. Supondo que o robô está na posição 0, 0 e virado para o norte, o comando PF 5 colocará o robô na posição 5, 0, mantendo a direção para o norte.
CRIAR A=10
CRIAR B=20
PF A+B
Fará com que o robô ande 30 passos para frente.

comando: PT n ou para trás n
Argumentos: n é o número de passos. Este comando aceita somente números inteiros e positivos, ou variáveis que armazenam números inteiros, ou expressões matemáticas que resultem em números inteiros.
Explicação: move n passos para trás. Este comando aceita somente números inteiros, ou variáveis que armazenam números inteiros, ou expressões matemáticas que resultem em números inteiros.
Exemplo: PF 5

O robô andará 5 passos para frente. Supondo que o robô está na posição 0, 0 e virado para o norte, o comando PF 5 colocará o robô na posição 5, 0, mantendo a direção para o norte.
CRIAR A = 10
PF A
Fará com que o robô ande 10 passos para frente.
CRIAR A=10
CRIAR B=20
PF A+B
Fará com que o robô ande 30 passos para frente.
Se o robô colidir em algo antes de completar a quantidade de passos solicitados. Será informado ao usuário: "Andei somente X passos para frente. Encontrei obstáculo".
Se for digitado o comando com um número negativo como abaixo:
PF -5
Será informado ao usuário que o robô andou 0 passos.

[Próximo] [Seção Anterior][Menu][1.contexto da GoDonnie]
[2.sair do terminal de programação][3. declaração de variáveis]
[4. comandos de áudio][5. operadores]
[6. comandos de movimentação][7. comandos de rotação]
[8. comandos de visualização do ambiente][9. comandos de informações (ambiente)]
[10. comandos de condição][11. comandos de repetição]
[12. declaração de procedimentos][13. comandos variados][14. Exemplos de aplicação]

Seção 6: comandos de rotação
Rotação sem movimento do robô

comando: GD n
Argumentos: n é número de graus. Este comando aceita somente números inteiros positivos e negativos, ou variáveis que armazenam números inteiros, ou expressões matemáticas que resultem em números inteiros.
Explicação: Gira n graus para direita. Não há deslocamento do robô.
Exemplo: GD 90
O robô irá girar 90 graus para direita. Supondo que o robô está virado para o norte, o comando GD 90 irá girar o robô 90 graus para a direita, mantendo-o na direção leste.
CRIAR A = 45
GD A
Fará com que o robô gire 45 graus para a direita.
CRIAR A=80
CRIAR B=10
GD A+B
Fará com que o robô gire 90 graus para a direita.

GD -90
O robô gira para o lado esquerdo 90 graus.

comando: GE n
Argumentos: n é número de graus. Este comando aceita somente números inteiros positivos e negativos, ou variáveis que armazenam números inteiros, ou expressões matemáticas que resultem em números inteiros.
Explicação: o robô gira n graus para esquerda. Não há deslocamento do robô.
Exemplo: GE 90
O robô irá girar 90 graus para esquerda. Supondo que o robô está virado para o leste, o comando GE 90 irá girar o robô 90 graus para a esquerda, mantendo-o na direção norte.
CRIAR A = 45
GE A

Fará com que o robô gire 45 graus para a esquerda.
CRIAR A=80
CRIAR B=10
GE A+B
Fará com que o robô gire 90 graus para a esquerda.
GE -90
O robô gira para o lado direito 90 graus.

[Próximo] [Seção Anterior][Menu][1.contexto da GoDonnie]
[2.sair do terminal de programação][3. declaração de variáveis]
[4. comandos de áudio][5. operadores]
[6. comandos de movimentação][7. comandos de rotação]
[8. comandos de visualização do ambiente][9. comandos de informações (ambiente)]
[10. comandos de condição][11. comandos de repetição]
[12. declaração de procedimentos][13. comandos variados][14. Exemplos de aplicação]

Seção 7: comandos de visualização do ambiente
São comandos para obter informações sobre o ambiente em que o robô está. Não é possível armazenar o retorno desses comandos em variáveis.

comando: ESPIAR
Explicação: retorna à identificação do objeto, um ângulo aproximado e a distância aproximada de colisão entre o robô e o objeto identificado. O rastreamento para identificação dos objetos ocorre a 90 graus a esquerda e a direita da frente do robô.
Exemplo: Supondo que o robô está na posição 2,3, virado para o norte, e que há um obstáculo verde na posição 0,5 e outro obstáculo vermelho na posição 6,3.
ESPIAR
Será falado:
A 40 graus a esquerda: 1 objeto de cor verde a 2 passos.
A 90 graus a direita: 1 objeto da cor vermelha a 4 passos.
No caso de dois objetos no mesmo ângulo será informado:
a 30 graus a esquerda: dois objetos de cores verde, vermelho a 17 passos.

comando: ESTADO
Explicação: Retorna à posição no eixo X, Y e o ângulo do robô e informa o último comando digitado de rotação ou de deslocamento, anterior ao comando ESTADO.
Exemplo: PF 3 ESTADO
Supondo que o robô estava em 0,0. O robô andará 3 passos para frente e informará "Comando 1 foi PF 3, andou 3, não bateu, posição [3,0,0]. O 3 corresponde ao eixo x, o primeiro 0 ao eixo y e o último 0 ao ângulo do robô.
Caso o robô tenha colidido em algo completando apenas 2 passos com sucesso, o ESTADO retornará:
"Comando 1 foi PF 3, andou 2, bateu, posição [2,0,0]". O 2 corresponde ao eixo x, o primeiro 0 ao eixo y e o último 0 ao ângulo do robô.
Não havendo comandos digitados anteriormente, retornará:
Nenhum comando executado, Posição [0, 0, 0].

[Próximo] [Seção Anterior][Menu][1.contexto da GoDonnie]
[2.sair do terminal de programação][3. declaração de variáveis]
[4. comandos de áudio][5. operadores]
[6. comandos de movimentação][7. comandos de rotação]
[8. comandos de visualização do ambiente][9. comandos de informações (ambiente)]
[10. comandos de condição][11. comandos de repetição]
[12. declaração de procedimentos][13. comandos variados][14. Exemplos de aplicação]

Seção 8: comandos de posição e percepção do ambiente
São comandos para obter informações sobre o ambiente em que o robô está. É possível armazenar o retorno desses comandos em variáveis.

comando: DISTÂNCIA d
Argumentos: d é a direção do sensor do robô (f - frontal; fd - frontal direita; fe -frontal esquerda; td - traseiro direito; t - traseiro; te - traseiro esquerda)
Explicação: retorna à quantidade de passos do sensor do robô até um obstáculo, de acordo com a direção escolhida.
Exemplo:
Sensor da frente: **DISTÂNCIA F**
Sensor da frente direita: **DISTÂNCIA FD**
Sensor da frente esquerda: **DISTÂNCIA FE**
Sensor de traseira: **DISTÂNCIA T**
Sensor de traseira esquerda: **DISTÂNCIA TE**
Sensor de traseira direita: **DISTÂNCIA TD**

Há três formas de se utilizar o comando DISTÂNCIA:
1) Se o usuário desejar escutar o retorno, deve utilizar o comando FALAR na frente do comando DISTÂNCIA. Supondo que o robô está na posição 0,0, virado para o norte e há obstáculos nas seguintes posições, o resultado será:
Obstáculo em 0, 3:
Exemplo: FALAR DISTÂNCIA F
Será falado: 3 passos
2) Se deseja armazenar em uma variável.
Exemplo: CRIAR d = DISTÂNCIA T
Armazena na variável d a distância traseira do robô até o obstáculo que está diretamente atrás dele. Supondo que o Robô está na posição 0,3 virado para o norte e existe um obstáculo em 0,0. O valor armazenado em d será 3.
3) Se deseja usar diretamente dentro de outro comando, por exemplo: SE (será visto na seção 9), PARA (será visto na seção 10), REPITA (será visto na seção 10) ou ENQUANTO (será visto na seção 10).
Exemplo: SE DISTÂNCIA F > 0 ENTÃO
PF 1
SENÃO
FALAR "não é possível andar para frente"
FIM SE
No exemplo acima, se a distância frontal do robô for maior que 0, o robô andará 1 passo para frente. Se for igual ou menor a 0, irá falar "não é possível andar para frente".

ENQUANTO DISTÂNCIA F>0
FAÇA
PF 1
FIM ENQUANTO
No exemplo acima, enquanto a distância frontal do robô em relação ao objeto for maior que 0, andará 1 passo para frente.

comando: POS k
Argumentos: k é um eixo do plano cartesiano (X ou Y) ou ângulo (A), é possível utilizar o comando sem argumento, dessa forma retornará o X, Y e A ao mesmo tempo.
Explicação: retorna à posição atual do robô no eixo X ou no eixo Y ou o ângulo atual do robô.
Há três formas de se utilizar o comando POS k:
1) Se o usuário deseja escutar o retorno, deve utilizar o comando FALAR à frente do comando POS x, POS y ou POS a.
Exemplo: Supondo que o robô está na posição 0,0 virado para o norte:
FALAR POS x

será falado 0
FALAR POS y
será falado 0
FALAR POS a
Será falado 0
2) Se deseja somente armazenar em uma variável.
Exemplo: CRIAR x = POS x
A variável x possui a posição do robô no eixo x.
CRIAR y = POS y
A variável y contém a posição do robô no eixo y.
CRIAR a = POS a
A variável a contém o ângulo do robô.
3) Se deseja usar diretamente dentro de outro comando, por exemplo: SE (será visto na seção 9), PARA (será visto na seção 10), REPITA (será visto na seção 10) ou ENQUANTO (será visto na seção 10).
Exemplo: SE POS y > 0 ENTÃO
PF 5
SENÃO
PT 5
FIM SE

comando: COR c
Argumentos: c é a cor desejada (azul; vermelho; verde)
Explicação: Verifica quantos objetos de determinada cor o robô consegue identificar num ângulo de 180 graus a sua frente. Retorna um inteiro indicando o número de objetos a frente do robô da cor azul, vermelha ou verde.
Há três formas de se utilizar o comando COR:
1) Se o usuário desejar escutar o retorno, deve utilizar o comando FALAR a frente do comando COR.
Exemplo: Supondo que há 1 objeto verde e 2 azuis
FALAR COR azul
será falado: 2
FALAR COR verde
será falado: 1
2) Se deseja somente armazenar em uma variável, declarando-a anteriormente.
Exemplo: CRIAR A = COR AZUL
A variável A possui a quantidade de objetos azuis
CRIAR V = COR VERDE
A variável V contém a quantidade de objetos verdes.
3) Se deseja usar diretamente dentro de outro comando, por exemplo: SE (será visto na seção 9), PARA (será visto na seção 10), REPITA (será visto na seção 10) ou ENQUANTO (será visto na seção 10).
Exemplo: SE COR AZUL > 0 ENTÃO
FALAR "Número de objetos azuis"
FALAR COR AZUL
SENÃO
FALAR "Não encontrei objetos azuis"
FIM SE

SE COR VERDE > 0 ENTÃO
FALAR "Número de objetos verdes"
FALAR COR VERDE
SENÃO
FALAR "Não encontrei objetos verdes"
FIM SE

[Próximo] [Seção Anterior][Menu][1.contexto da GoDonnie]

[2.sair do terminal de programação][3. declaração de variáveis]
[4. comandos de áudio][5. operadores]
[6. comandos de movimentação][7. comandos de rotação]
[8. comandos de visualização do ambiente][9. comandos de informações (ambiente)]
[10. comandos de condição][11. comandos de repetição]
[12. declaração de procedimentos][13. comandos variados][14. Exemplos de aplicação]

Seção 9: comandos de condição
São comandos condicionais que permitem ao programa fazer a escolha do que executar, de acordo com uma condição estipulada.

Comando: SE expressão operador lógico expressão **ENTÃO** comandos **FIM SE**
expressão = variável ou expressão.
Argumentos: expressão é igual a uma variável ou expressão
Explicação: Testa se uma condição é verdadeira e, em caso afirmativo, executa os primeiros comandos.
Exemplo: CRIAR a = 0
SE a<4
ENTÃO PF 5
FIM SE
Se a variável "a" tiver um valor menor do que 4 então o robô andará 5 passos para frente.

Comando: SE expressão-comparador-expressão **ENTÃO** comandos **SENÃO** comandos **FIM SE**
Argumentos: expressão = variável ou expressão.
Explicação: Testa se uma condição é verdadeira e, em caso afirmativo, executa os primeiros comandos. Caso contrário, executa os comandos da expressão SENÃO.
Exemplo:
Supondo que, se a variável a for menor do que 4 o robô tenha que andar para frente 5 passos e caso contrário tenha que girar 45 graus para esquerda:
CRIAR a = 0
SE a<4
  ENTÃO PF 5
  SENÃO GE 45
FIM SE

[Próximo] [Seção Anterior][Menu][1.contexto da GoDonnie]
[2.sair do terminal de programação][3. declaração de variáveis]
[4. comandos de áudio][5. operadores]
[6. comandos de movimentação][7. comandos de rotação]
[8. comandos de visualização do ambiente][9. comandos de informações (ambiente)]
[10. comandos de condição][11. comandos de repetição]
[12. declaração de procedimentos][13. comandos variados][14. Exemplos de aplicação]

Seção 10: Comandos de repetição
São comandos de repetição que permitem uma ou mais instruções serem executadas um determinado número de vezes.

comando: PARA inicialização**;** expressão operador lógico expressão**;** incremento ou decremento **FAÇA** comandos **FIM PARA**
Argumentos: Inicialização: variável = algum valor inteiro
variável ou Expressão operador lógico variável ou expressão:
variável ou expressão - operador lógico - variável ou expressão
Incremento: variável + constante ou variável + variável

Decremento: variável - constante ou variável - variável
Explicação: repete a sequência de **comandos** um determinado número de vezes.
Exemplo:
O exemplo faz com que o robô precise andar em direção a um obstáculo que está a sua frente e a cada passo fale "oi".
CRIAR obstaculo = DISTÂNCIA F
PARA CRIAR x=1; x<=obstaculo; x=x+1
FAÇA
PF 1
FALAR "oi"
FIM PARA
A variável "x" começará com o valor 1 e o robô andará um passo para frente e falará "oi", enquanto seu valor for menor ou igual a linha do obstáculo que está à sua frente.

comando: REPITA n **VEZES** comandos **FIM REPITA**
Argumentos: n é o número de vezes que os comandos serão repetidos.
Explicação: Repete os comandos n vezes
Exemplo:
REPITA 4 VEZES
GD 90
PF 2
FIM REPITA
Supondo que o robô comece na posição 0,0. Os comandos PF 3 GD 90 serão repetidos 4 vezes. Ao final, o robô terá feito um trajeto similar a um quadrado e finalizará na posição 0,0 virado para o norte.

comando: ENQUANTO expressão operador lógico expressão **FAÇA** comandos **FIM ENQUANTO**
Argumentos: variável ou Expressão operador lógico variável ou expressão:
Variável ou expressão - operador lógico - variável ou expressão
Explicação: Repete os comandos enquanto a Expressão-operador lógico-expressão for verdadeira.
Exemplo: O exemplo faz com que o robô precise andar em direção a um obstáculo que está a sua frente e a cada passo fale "estou chegando".
ENQUANTO DISTÂNCIA F >3
FAÇA
PF 1
FALAR "estou chegando"
FIM ENQUANTO
Enquanto a distância da frente do robô em relação ao objeto for maior que 3, o robô andará um passo para frente e falará "estou chegando".

[Próximo] [Seção Anterior][Menu][1.contexto da GoDonnie]
[2.sair do terminal de programação][3. declaração de variáveis]
[4. comandos de áudio][5. operadores]
[6. comandos de movimentação][7. comandos de rotação]
[8. comandos de visualização do ambiente][9. comandos de informações (ambiente)]
[10. comandos de condição][11. comandos de repetição]
[12. declaração de procedimentos][13. comandos variados][14. Exemplos de aplicação]

Seção 11: Declaração de procedimentos
Procedimento é um programa menor (subprograma) que permite decompor e resolver um problema mais complexo em um mais simples. Pode ser chamado em outras partes do programa.

comando: APRENDER nome: variável1, variável2, variável3, **... FAÇA comandos FIM APRENDER**
Argumentos: nome é o nome do subprograma e variavel1, variavel2, variavel3 são variáveis que só existiram no contexto deste subprograma.
Explicação: Serve para criar um subprograma. Este comando somente funciona via arquivo.
Exemplo: O robô precisa caminhar simulando um retângulo. Esse retângulo pode ter tamanhos diferentes, conforme a atividade. Por isso, pode ser utilizado o comando APRENDER para criar um procedimento único chamado RETÂNGULO que receberia duas variáveis, uma para o tamanho da altura e a outra para o tamanho da base. Assim, esse procedimento poderia ser utilizado para fazer retângulos de tamanhos diferentes.

```
APRENDER RETÂNGULO: base, altura
FAÇA
PF base GD 90
PF altura GD 90
PF base GD 90
PF altura GD 90
FIM APRENDER
```

Ou

```
APRENDER RETÂNGULO: base, altura
FAÇA
REPITA 2 VEZES
PF base GD 90
PF altura GD 90
FIM REPITA
FIM APRENDER
```

```
chamada do subprograma:
RETÂNGULO [5,3]
RETÂNGULO [8,4]
RETÂNGULO [9,5]
```

Seção 12: comandos variados

Comando: ESPERAR t
Argumento: t é o tempo em segundos
Explicação: Espera t segundos para executar o próximo comando.
Exemplo: Se o robô deve andar para frente 2 passos, esperar 3 segundos e andar mais 4 passos:

```
PF 2
ESPERAR 3
PF 4
```

Comando: --
Após esse símbolo -- tudo que for escrito na linha que possui -- não será executado. São lembretes sobre o código.
Exemplo: -- Isto é um comentário.

[Próximo] [Seção Anterior][Menu][1.contexto da GoDonnie]
[2.sair do terminal de programação][3. declaração de variáveis]
[4. comandos de áudio][5. operadores]
[6. comandos de movimentação][7. comandos de rotação]

[8. comandos de visualização do ambiente][9. comandos de informações (ambiente)]
[10. comandos de condição][11. comandos de repetição]
[12. declaração de procedimentos][13. comandos variados][14. Exemplos de aplicação]

Exemplos de aplicação

Aqui serão descritos problemas mais complexos

1. O robô está em um cenário que possui alguns objetos. Faça um programa para verificar se existem objetos da cor azul, se houver informe a quantidade, se não houver falar "Não tem objetos com essa cor".
Sugestão de solução:
Criar azul = cor azul
Se azul>0 então
falar azul
senão falar "Não tem objetos com essa cor"
fim se

2. Faça um programa para que o robô se desloque até um objeto que está a sua frente e fale a quantidade de passos que faltam para atingir o objetivo que é chegar ao objeto.
Sugestão de solução:
Enquanto distância f > 0 faça
falar distância f
pf 1
fim enquanto

3. O robô está localizado na posição de eixos x=0 e y=0. Faça um programa para que o robô caminhe em forma de ziguezague.

Sugestão de Solução:
Aprender ziguezague: passos faça
Repita passos vezes
Gd 90
Pf 1
Gd 90
Pf 1
Gd 90
Pf 1
Gd 90
Pf 1
Fim repita
Fim aprender

Ziguezague [5]

4. Faça com que o Robô simule a subida em uma escada com 6 degraus de tamanhos iguais. O robô deve ficar parado 2 segundos entre a construção de cada degrau.
Sugestão de solução:
Repita 4 vezes
Pf 1
Gd 90
Pf 1
Ge 90
Esperar 2

Fim repita

[Próximo][Seção Anterior][Menu][1.contexto da GoDonnie]
[2.sair do terminal de programação][3. declaração de variáveis]
[4. comandos de áudio][5. operadores]
[6. comandos de movimentação][7. comandos de rotação]
[8. comandos de visualização do ambiente][9. comandos de informações (ambiente)]
[10. comandos de condição][11. comandos de repetição]
[12. declaração de procedimentos][13. comandos variados][14. Exemplos de aplicação]

APÊNDICE B – MODELOS DE TCLES

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Prezado(a) Participante:_____

Informação ao sujeito da pesquisa:

Você está sendo convidado(a) a participar do trabalho de pesquisa que trata da concepção, modelagem, e validação de uma linguagem de programação para comandar um robô por usuários que são cegos bem como sobre uma proposta de utilização dessa linguagem, de responsabilidade da mestranda Juliana Damasio Oliveira sob a orientação da Profa. Dra. Márcia de Borba Campos, ambas do PPGCC/PUCRS.

Sua participação envolve realizar atividades de programação. Além disso, será necessário responder a uma entrevista. Para auxílio a nossa análise de dados será necessário registrar a sua participação através de fotos, vídeo e áudio. As informações obtidas através dessa pesquisa serão confidenciais e asseguramos o sigilo sobre sua participação. Assim, os dados não serão divulgados de forma a possibilitar sua identificação.

Sua participação nesse estudo é voluntária e se você decidir não participar, ou quiser cancelar sua participação em qualquer momento, tem absoluta liberdade de fazê-lo. Mesmo não tendo benefícios diretos em participar, indiretamente você estará contribuindo para a compreensão do fenômeno estudado e para a produção de conhecimento científico.

Quaisquer dúvidas relativas à pesquisa poderão ser esclarecidas pelos pesquisadores pelos e-mails juliana.damasio@acad.pucrs.br e marcia.campos@pucrs.br.

DECLARAÇÃO DE CONSENTIMENTO DO SUJEITO DA PESQUISA:

Concordo em participar deste estudo e declaro que eu li os detalhes descritos neste documento. Entendo que eu sou livre para aceitar ou recusar, e que posso interromper a minha participação a qualquer momento sem dar uma razão. Eu concordo que os dados coletados para o estudo sejam usados para o propósito acima descrito. Eu entendi a informação apresentada neste TERMO DE CONSENTIMENTO. Eu tive a oportunidade para fazer perguntas e todas as minhas perguntas foram respondidas. Eu receberei uma cópia assinada e datada deste Documento de CONSENTIMENTO LIVRE E ESCLARECIDO.

_____ _____

Assinatura do participante Local e data

Assinatura dos Pesquisadores:

_____

Juliana Damasio Oliveira
Matrícula: 15290089-0

_____

Márcia de Borba Campos

Agradecemos sua participação nesta pesquisa.

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Prezado(a) Participante:_____

Informação ao sujeito da pesquisa:

Você está sendo convidado(a) a participar do trabalho de pesquisa que trata da concepção, modelagem, e validação de uma linguagem de programação para comandar um robô por usuários que são cegos bem como sobre uma proposta de utilização dessa linguagem, de responsabilidade da mestranda Juliana Damasio Oliveira sob a orientação da Profa. Dra. Márcia de Borba Campos, ambas do PPGCC/PUCRS.

Sua participação envolve realizar atividades guiadas presenciais e a distância. Além disso, será necessário responder a uma entrevista. Para auxílio a nossa análise de dados será necessário registrar a sua participação através de fotos, vídeo e áudio. As informações obtidas através dessa pesquisa serão confidenciais e asseguramos o sigilo sobre sua participação. Assim, os dados não serão divulgados de forma a possibilitar sua identificação.

Sua participação nesse estudo é voluntária e se você decidir não participar, ou quiser cancelar sua participação em qualquer momento, tem absoluta liberdade de fazê-lo. Mesmo não tendo benefícios diretos em participar, indiretamente você estará contribuindo para a compreensão do fenômeno estudado e para a produção de conhecimento científico.

Quaisquer dúvidas relativas à pesquisa poderão ser esclarecidas pelos pesquisadores pelos e-mails juliana.damasio@acad.pucrs.br e marcia.campos@pucrs.br.

DECLARAÇÃO DE CONSENTIMENTO DO SUJEITO DA PESQUISA:

Concordo em participar deste estudo e declaro que eu li os detalhes descritos neste documento. Entendo que eu sou livre para aceitar ou recusar, e que posso interromper a minha participação a qualquer momento sem dar uma razão. Eu concordo que os dados coletados para o estudo sejam usados para o propósito acima descrito. Eu entendi a informação apresentada neste TERMO DE CONSENTIMENTO. Eu tive a oportunidade para fazer perguntas e todas as minhas perguntas foram respondidas. Eu receberei uma cópia assinada e datada deste Documento de CONSENTIMENTO LIVRE E ESCLARECIDO.

_____ _____

Assinatura do participante Local e data

Assinatura dos Pesquisadores:

_____

Juliana Damasio Oliveira
Matrícula: 15290089-0

_____

Márcia de Borba Campos

Agradecemos sua participação nesta pesquisa.

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Prezado(a)
Participante:_____

Informação ao sujeito da pesquisa:

Você está sendo convidado(a) a participar do trabalho de pesquisa que trata da concepção, modelagem, e validação de uma linguagem de programação para comandar um robô por usuários que que possuem deficiência visual bem como sobre uma proposta de utilização dessa linguagem, de responsabilidade da mestranda Juliana Damasio Oliveira sob a orientação da Profa. Dra. Márcia de Borba Campos, ambas do PPGCC/PUCRS.

Sua participação envolve conhecer a linguagem e o ambiente de programação *GoDonnie* que está sendo proposto e realizar atividades de programação. Além disso, será necessário responder a um questionário. Para auxílio a nossa análise de dados será necessário registrar a sua participação por meio de áudio, vídeo e fotos. As informações obtidas por meio dessa pesquisa serão confidenciais e asseguramos o sigilo sobre sua participação. Assim, os dados não serão divulgados de forma a possibilitar sua identificação.

Sua participação nesse estudo é voluntária e se você decidir não participar, ou quiser cancelar sua participação em qualquer momento, tem absoluta liberdade de fazê-lo. Mesmo não tendo benefícios diretos em participar, indiretamente você estará contribuindo para a compreensão do fenômeno estudado e para a produção de conhecimento científico.

Quaisquer dúvidas relativas à pesquisa poderão ser esclarecidas pelos pesquisadores pelos e-mails juliana.damasio@acad.pucrs.br e marcia.campos@pucrs.br.

DECLARAÇÃO DE CONSENTIMENTO DO SUJEITO DA PESQUISA:

Concordo em participar deste estudo e declaro que eu li os detalhes descritos neste documento. Entendo que eu sou livre para aceitar ou recusar, e que posso interromper a minha participação a qualquer momento sem dar uma razão. Eu concordo que os dados coletados para o estudo sejam usados para o propósito acima descrito. Eu entendi a informação apresentada neste TERMO DE CONSENTIMENTO. Eu tive a oportunidade para fazer perguntas e todas as minhas perguntas foram respondidas. Eu receberei uma cópia assinada e datada deste Documento de CONSENTIMENTO LIVRE E ESCLARECIDO.

_____ _____
Assinatura do participante Local e data

Assinatura dos Pesquisadores:

_____
Juliana Damasio Oliveira
Matrícula: 15290089-0

_____
Márcia de Borba Campos

Agradecemos sua participação nesta pesquisa.

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Prezado(a)
Participante:_____

Informação ao sujeito da pesquisa:

Você está sendo convidado(a) a participar do trabalho de pesquisa que trata da concepção, modelagem, e validação de uma linguagem de programação para comandar um robô por usuários que são cegos bem como sobre uma proposta de utilização dessa linguagem, de responsabilidade da mestranda Juliana Damasio Oliveira sob a orientação da Profa. Dra. Márcia de Borba Campos, ambas do PPGCC/PUCRS.

Sua participação envolve conhecer a linguagem de programação *GoDonnie* que está sendo proposta. Além disso, será necessário responder a um questionário. Para auxílio a nossa análise de dados será necessário registrar a sua participação por meio de áudio. As informações obtidas por meio dessa pesquisa serão confidenciais e asseguramos o sigilo sobre sua participação. Assim, os dados não serão divulgados de forma a possibilitar sua identificação.

Sua participação nesse estudo é voluntária e se você decidir não participar, ou quiser cancelar sua participação em qualquer momento, tem absoluta liberdade de fazê-lo. Mesmo não tendo benefícios diretos em participar, indiretamente você estará contribuindo para a compreensão do fenômeno estudado e para a produção de conhecimento científico.

Quaisquer dúvidas relativas à pesquisa poderão ser esclarecidas pelos pesquisadores pelos e-mails juliana.damasio@acad.pucrs.br e marcia.campos@pucrs.br.

DECLARAÇÃO DE CONSENTIMENTO DO SUJEITO DA PESQUISA:

Concordo em participar deste estudo e declaro que eu li os detalhes descritos neste documento. Entendo que eu sou livre para aceitar ou recusar, e que posso interromper a minha participação a qualquer momento sem dar uma razão. Eu concordo que os dados coletados para o estudo sejam usados para o propósito acima descrito. Eu entendi a informação apresentada neste TERMO DE CONSENTIMENTO. Eu tive a oportunidade para fazer perguntas e todas as minhas perguntas foram respondidas. Eu receberei uma cópia assinada e datada deste Documento de CONSENTIMENTO LIVRE E ESCLARECIDO.

_____ _____
Assinatura do participante Local e data

Assinatura dos Pesquisadores:

_____
Juliana Damasio Oliveira
Matrícula: 15290089-0

_____
Márcia de Borba Campos

Agradecemos sua participação nesta pesquisa.

APÊNDICE C – INSTRUMENTOS DE AVALIAÇÃO COM USUÁRIOS VIDENTES

Pré-teste

1. Nome:
2. Idade:
3. Gênero: ( ) Feminino ( ) Masculino ( ) Outro
4. Formação:
5. Conhece linguagens de programação? ( )Sim ( )Não
6. Se sim, quais?
7. Qual o nível de conhecimento nas linguagens descritas?
8. Possui experiência com robótica? ( )Sim ( )Não. Se sim, descreva.
9. Qual o nível de conhecimento em robótica descritas?
10. Já utilizou leitor de telas? ( ) Sim ( ) Não

Atividades presenciais de programação

Atividades guiadas

Id	Atividade	Discussão
1	Fazer o robô se deslocar e retornar a posição original. Posição inicial: • O robô está na posição de eixos x=0 e y=0. • O robô está virado para o norte. Sugestão de comandos: • Deslocar o robô para frente, andando 3 passos. • Girar para direita 180 graus • Deslocar o robô para frente, andando 3 passos.	Quais são as outras possibilidades? • Experimente realizar o desafio utilizando o comando de giro para esquerda. • Experimente realizar o desafio sem utilizar comandos de giro.
2	Fazer o robô se deslocar numa trajetória no formato de um quadrado de tamanho de lado 4. Posição inicial: • O robô está na posição de eixo x=5 e eixo y=6. • O robô está virado para o norte. Sugestão de comandos: • Deslocar o robô para frente, andando 4 passos. • Girar para a esquerda 90 graus. • Deslocar o robô para frente, andando 4 passos. • Girar para a esquerda 90 graus. • Deslocar o robô para frente, andando 4 passos. • Girar para a esquerda 90 graus. • Deslocar o robô para frente, andando 4 passos. • Girar para a esquerda 90 graus.	Quais são as outras possibilidades? • Experimente realizar o desafio utilizando o comando de repetição.
3	Fazer com que o robô reconheça os objetos a sua	Experimente utilizar o comando

	frente e faça um bip quando encontrar um objeto verde. Dica: existem 3 objetos a frente do robô e as cores podem ser: verde, vermelho ou azul. Posição inicial: ● O robô está na posição de eixo x=4 e eixo y=0. ● O robô está virado para o norte. Sugestão de comandos: - Criar uma variável e armazenar nela a quantidade de objetos verdes. - SE existirem objetos verdes armazenados na variável então faça um bip para a quantidade de objetos verdes. (Dica: comando SE, REPITA e BIP) - Se não existirem objetos verdes falar a mensagem "Não existem objetos dessa cor". (Dica: comando FALAR)	PARA no lugar de REPITA. Experimente não utilizar variáveis.
4	Fazer com que o robô se desloque até um objeto que está a sua frente e fale a quantidade de passos que faltam para atingir o objetivo que é chegar ao objeto. Posição inicial: ● O robô está na posição de eixo x=4 e eixo y=0. ● O robô está virado para o norte. Sugestão de comandos: - ENQUANTO a distância a frente do robô for maior que zero, falar a quantidade de passos. (Dica: ENQUANTO, DISTÂNCIA e FALAR) - Quando chegar no objeto falar "Cheguei" (Dica: FALAR)	Experimente fazer o mesmo exercício criando uma variável que armazene a distância.

5	Fazer com que o robô caminhe em forma de zigue-zague.
	
	Posição inicial:
	O robô está localizado na posição de eixos x=0 e y=0.
	
	Sugestão de solução:
	- Faça com que o robô APRENDA o movimento de zigue-zague, para isso ele deve receber a quantidade de passos. (Dica: APRENDER)
	- Deve REPETIR a quantidade de vezes: (Dica: REPITA)
	  - Girar para direita 90 graus
	  - Andar para frente 1 passo
		Discussão: Quais são as outras possibilidades?

	- Girar para a esquerda 90 graus - Andar para frente 1 passo - Girar para esquerda 90 graus - Andar para frente 1 passo - Girar para direita 90 graus - Andar para frente 1 passo	

Atividade com nível de dificuldade fácil

Id	Tipo de atividade	Desafio
6	Caminho	Faça com que o Robô simule a subida em uma escada com 4 degraus de tamanhos iguais. O robô deve ficar parado 2 segundos entre a construção de cada degrau. Escolha a posição inicial da escada.

Atividade com nível de dificuldade intermediário

Id	Atividade	Desafio
7	Labirinto	Dado um labirinto que contém 3 objetos. O robô parte da posição de eixos x=0 e y=0 e está virado para norte. O robô deve se deslocar até o eixos x=8 e y=14 sem bater em objetos. Discussão: Quais são as outras possibilidades de resolução?

Atividade com nível de dificuldade Avançado

Id	Atividade	Desafio
8	Procurar Objeto	O robô deve explorar livremente o cenário e informar quantos objetos existem da cor vermelha, verde e azul. O robô inicia na posição de eixos x=0 e y=0, virado para o norte. Dica: existem 5 objetos. Discussão: Quais são as outras possibilidades de resolução?

Pós-teste

Perguntas para entrevista estruturada

1. O manual auxiliou na execução das atividades? Teria alguma sugestão?
2. Os comandos foram fáceis de entender? Explique.
3. Teve alguma dificuldade na execução das atividades?
4. Teve algum comando que necessitou e não estava no manual?
5. Tem alguma sugestão quanto a atividade ou manual?
6. Os comandos estão de acordo com as suas funções?
7. O nível de dificuldade das atividades estavam de acordo com o indicado?

APÊNDICE D – INSTRUMENTOS DE AVALIAÇÃO COM USUÁRIO CEGO SEM CONHECIMENTO EM PROGRAMAÇÃO

Pré-teste

1. Nome:
2. Idade:
3. Gênero**:** ( ) Feminino ( ) Masculino ( ) Outro
4. Nível de ensino:
5. Qual curso?
6. Qual leitor de telas utiliza?
7. Sabe braille? ( ) Sim ( ) Não. Com quantos anos aprendeu?
8. Nível de acuidade visual:
   ( ) Baixa visão - acuidade de 0,3 à 0,05
   ( ) Cegueira - acuidade igual ou menor que 0,05.
9. Sua acuidade visual é: ( ) congênita ou ( ) adquirida
10. Tem conhecimento sobre graus? ( ) Sim ( ) Não.
11. Qual a forma que prefere se orientar: ( ) graus ou ( ) horas ( ) Outro_____
12. Conhece linguagens de programação? ( )Sim ( )Não. Se sim, quais?
13. Qual o nível de conhecimento nas linguagens descritas? (básico, intermediário, avançado)
14. Possui experiência com robótica? ( )Sim ( )Não. Se sim, descreva.
15. Qual o nível de conhecimento em robótica descritas? (básico, intermediário, avançado)
16. Nesse momento, qual seu nível de interesse em aprender programação?
    a. ( ) nenhum
    b. ( ) baixo
    c. ( ) médio
    d. ( ) alto
    e. ( ) nenhum

17. Nesse momento, qual seu nível de interesse em aprender sobre robótica?
    a. ( ) nenhum
    b. ( ) baixo
    c. ( ) médio
    d. ( ) alto
    e. ( ) nenhum

Instrumento de validação dos comandos GoDonnie

primeiro dia					segundo dia						terceiro dia						Obs.
Comandos	**Fácil de Aprender?**			**Quantas vezes expliquei?**	**Fácil de lembrar?**				**Quantos dias depois**	**Quantas vezes expliquei**	**Fácil de lembrar?**				**Quantos dias depois**	**Quantas vezes expliquei**	
					Sintaxe		**Semântica**				**Sintaxe**		**Semântica**				
1 dia	Sim	Não	Neutro		Sim	Não	Sim	Não			Sim	Não	Sim	Não			
PF																	
PT																	
GD																	
GE																	
ESPERAR																	
FALAR “x”																	
SOM																	
ESPIAR																	
DISTÂNCIA																	
POS																	
CORES																	
2 dia																	
ESTADO																	
HISTÓRICO																	
SE ENTÃO																	
SE ENTÃO SENÃO																	
REPITA																	
3 dia																	
CRIAR																	
PARA																	

4 dia																	
ENQUANTO																	
FALAR X																	
APRENDER																	

Instrumento de validação dos materiais concretos

Materiais concretos	**Fácil de aprender**			**Quantas vezes expliquei?**	**Observações**
1 dia	Sim	Não	Neutro		
Ângulos					
Mapa tátil					
2 dia					
ângulos					
Mapa tátil					
3 dia					
ângulos					
Mapa tátil					

Pós-teste

1. Qual seu nível de interesse em aprender programação?
   a. ( ) Nenhum
   b. ( ) baixo
   c. ( ) médio
   d. ( ) alto

2. Qual seu nível de interesse em aprender sobre robótica?
   a. ( ) nenhum
   b. ( ) baixo
   c. ( ) médio
   d. ( ) alto

Atividades à distância de programação

LISTA 1

NÍVEL FÁCIL

1. Faça um programa para comandar o robô para andar para frente 10 passos.
2. Faça um programa para comandar o robô para andar para trás 5 passos.

3. Faça o robô girar para direita 90 graus.
4. Faça o robô andar para frente 15 passos, girar para direita 90 graus, andar mais 15 passos e falar "Cheguei".
5. Faça um programa para o robô para andar para frente 20 passos e voltar ao ponto inicial.

NÍVEL MÉDIO

1. Faça um programa para o robô fazer um quadrado de tamanho de lado 20.
2. Faça um programa para o robô fazer um quadrado de tamanho de lado 40.
3. Faça um programa para o robô fazer um retângulo de tamanhos de lado 10 e 15.
4. Faça um programa para ler a idade de uma pessoa. Falar se a pessoa já pode votar ou se ainda não pode.
5. Faça um programa para ler um número. Falar se o número lido é maior que 100 ou menor que 100.
6. Faça um programa para ler um número. Falar se o número é positivo ou negativo. Lembre que número positivo é maior que zero e número negativo é menor do que zero.

NÍVEL AVANÇADO

1. Faça o robô andar para frente 20 passos. A cada 2 passos, executar um bip.
2. Faça um programa para o robô simular uma escada com 10 degraus.

Perguntas de usabilidade e satisfação de uso

1. Você achou fácil fazer os exercícios?
2. Nós classificamos os exercícios em nível fácil, médio e difícil. Você acha que os exercícios estavam no nível correto?
3. Qual exercício demorou mais para fazer?
4. Quanto tempo precisou para fazer os exercícios? Você fez tudo de uma vez? Ou fez um pouco em um momento e continuou depois?
5. Você precisou pedir ajuda para resolver os exercícios?

LISTA 2

NÍVEL FÁCIL

1. Faça um programa para ler um número. Falar se o número lido é maior que 50 ou menor que 50
2. Faça um programa para o robô para andar para trás 20 passos e voltar ao ponto inicial.
3. O robô está com a cabeça virado para o norte. Ele precisa andar para frente 20 passos e voltar ao ponto inicial, mantendo a cabeça para o norte.
4. Quais dos programas abaixo estão corretos? Pode ter mais de uma resposta correta.
    a. Programa A: PF 20 GD 180 PF 20
    b. Programa B: PF 20 GD 180 PF 20 GD 180
    c. Programa C: PF 20 PT 20
5. O robô precisa fazer um quadrado de tamanho de lado 5.
6. Qual dos programas abaixo está correto? Pode ter mais de uma resposta correta.
    a. Programa A: REPITA 2 VEZES PF 5 GD 90 PF 5 GD 90 FIM REPITA
    b. Programa B: REPITA 4 VEZES PF 5 GE 90 FIM REPITA
    c. Programa C: REPITA 4 VEZES PF 90 GD 5 FIM REPITA

NÍVEL INTERMEDIÁRIO

1. Qual dos programas abaixo desenha uma escada com 10 degraus no qual os degraus tem tamanho de 5 passos?
    a. Programa A: REPITA 10 VEZES PF 5 FIM REPITA
    b. Programa B: REPITA 10 VEZES PF 5 GD 90 FIM REPITA

c. Programa C: REPITA 10 VEZES PF 5 GD 90 PF 5 GE 90 FIM REPITA

2. Analise as linhas abaixo e informe se estão corretas ou erradas. Se tiver errada, corrija. Observação: as linhas de comando não estão organizadas em um programa. São linhas que não se relacionam. É para ver se estão escritas corretamente ou não. Se tiverem com erro ou se estiverem incompletas, é para corrigir.
   a. ESPERAR
   b. PF 40
   c. GE 90
   d. GD 180
   e. SOM DESLIGAR
   f. FALAR OI TUDO BEM
   g. SE IDADE MAIOR DO QUE 18 ENTÃO FALAR "É MAIOR”
   h. DISTÂNCIA f
   i. POS x
   j. CRIAR NOTA DO ALUNO
   k. SE NOTA >= 7 ENTÃO FALAR "FÉRIAS”
   l. SE NOTA >= 7 SENÃO FALAR "ESTUDAR MAIS”
   m. SE NOTA >= 7 ENTÃO FALAR "FÉRIAS" SENÃO FALAR "ESTUDAR MAIS”

NÍVEL AVANÇADO

1. Leia a quantidade de faltas de um aluno e informe se ele ainda pode faltar ou se já estourou a quantidade de faltas. Se ele ainda puder faltar, dê um bip curto. Se ele não puder mais faltar, dê um bip mais longo. Ele pode ter até 20 faltas.

LISTA 3

---

1. João e Luís são amigos. Neste programa, joão é o amigo 1 que tem idade1. O luís é o amigo 2 que tem idade2. O programa abaixo deve ler a idade dos amigos e mostrar o nome de quem é o mais novo.
2. João e Luís são amigos. João é o amigo 1 que tem idade1. O Luís é o amigo 2 que tem idade2. Faça um programa para mostrar o nome de quem é mais velho.
3. Maria e Ana estão trabalhando. Maria tem o salario1 e Ana tem o salario2. Elas querem viajar e pensaram em juntar os salários. Faça um programa para calcular e mostrar quanto de dinheiro elas poderão ter no total.
4. Maria tem 5 bananas e Ana tem 10 bananas. Faça um programa para calcular e mostrar o total de bananas.
5. Você vai viajar e colocou combustível no seu carro. Você encheu o tanque com 30 litros. Ao chegar de viagem, viu que havia gasto 10 litros. Faça um programa para calcular e mostrar quantos litros de combustível ainda tem no carro?
6. Fazer um programa para ler a nota da prova 1 e a nota da prova 2.
7. Fazer um programa para ler a nota da prova 1, a nota da prova 2 e a nota do trabalho da disciplina de técnicas da avaliação da personalidade. Mostrar a nota final de G1.

APÊNDICE E – INSTRUMENTOS DE AVALIAÇÃO COM DEFICIENTES VISUAIS COM CONHECIMENTO EM PROGRAMAÇÃO

Pré-teste

1. Nome:
2. Idade:
3. Gênero**:** ( ) Feminino () Masculino ( ) Outro
4. Nível de ensino:
5. (Se tiver graduação) Qual curso de graduação e semestre?
6. Qual leitor de telas utiliza?
7. Sabe braille? ( ) Sim ( ) Não. Com quantos anos aprendeu?
8. Nível de acuidade visual:
   ( ) Baixa visão - acuidade de 0,3 à 0,05
   ( ) Cegueira - acuidade igual ou menor que 0,05.
9. Sua acuidade visual é: ( ) congênita ou ( ) adquirida.
10. Você possui algum treinamento de orientação e mobilidade? Se sim, descreva.
11. Você tem autonomia para se deslocar em espaços ou necessita de ajuda? (ambiente familiar ou não familiar , ambiente aberto e fechado)
12. Tem conhecimento sobre graus? ( ) Sim ( ) Não.
13. Você prefere se orientar por: ( ) graus ou ( ) horas ( ) Outro
14. Você conhece linguagens de programação?( )Sim ( )Não. Se sim, quais?
    a. Qual seu nível de conhecimento nas linguagens mencionadas? (básico, intermediário, avançado)
    b. Conhece estruturas de repetição? ( )Sim ( )Não. Quais? Pode escrever a sintaxe da estrutura?
    c. Conhece estrutura de seleção? ( )Sim ( )Não. Quais? Pode escrever a sintaxe da estrutura?
    d. Sua rotina de programação é:
        i. ( ) todos os dias
        ii. ( ) até 3 vezes na semana
        iii. ( ) de 1 a 2 vezes na semana
        iv. ( ) quinzenal
        v. ( ) mensal
        vi. ( ) não tenho como precisar
        vii. ( ) não tenho hábito de programar
    e. Você gosta de programar?
        i. ( ) sim, muito
        ii. ( ) sim, pouco
        iii. ( ) não
        iv. ( ) neutro
15. Possui experiência com robótica? ( )Sim ( )Não. Se sim, descreva.
    a. Qual o nível de conhecimento em robótica descritas? (básico, intermediário, avançado)
16. Nesse momento, qual seu nível de interesse em aprender sobre robótica?
    a. ( ) nenhum
    b. ( ) baixo
    c. ( ) médio
    d. ( ) alto

ATIVIDADES PRESENCIAIS GUIADAS DE PROGRAMAÇÃO

Existem 5 atividades guiadas, são tarefas diretas, de aplicação dos comandos, mas já orientadas a problemas. Após cada atividade existe uma discussão sobre a atividade.

1. Fazer o robô se deslocar e retornar a posição original.

   Posição inicial:
   - O robô está na posição de eixos x=0 e y=0.
   - O robô está virado para o norte.

   Sugestão de comandos:
   - Deslocar o robô para frente, andando 3 passos.
   - Girar para direita 180 graus
   - Deslocar o robô para frente, andando 3 passos.

   Discussão:
   Quais são as outras possibilidades?
   - Experimente realizar o desafio utilizando o comando de giro para esquerda.
   - Experimente realizar o desafio sem utilizar comandos de giro.
2. Fazer o robô se deslocar numa trajetória no formato de um quadrado de tamanho de lado 4.
   Posição inicial:
   - O robô está na posição de eixo x=5 e eixo y=6.
   - O robô está virado para o norte.

   Sugestão de comandos:
   - Deslocar o robô para frente, andando 4 passos.
   - Girar para a esquerda 90 graus.
   - Deslocar o robô para frente, andando 4 passos.
   - Girar para a esquerda 90 graus.
   - Deslocar o robô para frente, andando 4 passos.
   - Girar para a esquerda 90 graus.
   - Deslocar o robô para frente, andando 4 passos.
   - Girar para a esquerda 90 graus.

   Discussão:
   Quais são as outras possibilidades?
   - Experimente realizar o desafio utilizando o comando de giro para direita.
3. Fazer com que o robô reconheça os objetos a sua frente e faça um bip quando encontrar um objeto verde.

   Dica: existem 3 objetos a frente do robô e as cores podem ser: verde, vermelho ou azul.

   Posição inicial:
   - O robô está na posição de eixo x=4 e eixo y=0.
   - O robô está virado para o norte.

   Sugestão de comandos:
   - Criar uma variável e armazenar nela a quantidade de objetos verdes.

- SE existirem objetos verdes armazenados na variável então falar a quantidade de objetos verdes. (Dica: comando SE e FALAR)
- Se não existirem objetos verdes falar a mensagem "Não existem objetos desta cor". (Dica: comando FALAR)

Discussão: Experimente utilizar o comando PARA no lugar de REPITA. Experimente não utilizar variáveis.

4. Fazer com que o robô se desloque até um objeto que está a sua frente e fale a quantidade de passos que faltam para atingir o objetivo que é chegar ao objeto.

   Posição inicial:
   - O robô está na posição de eixo x=4 e eixo y=0.
   - O robô está virado para o norte.

   Sugestão de comandos:
   - ENQUANTO a distância a frente do robô for maior que zero, falar a quantidade de passos. (Dica: ENQUANTO, DISTÂNCIA e FALAR)
   - Quando chegar no objeto falar "Cheguei" (Dica: FALAR)

   Discussão: Experimente fazer o mesmo exercício criando uma variável que armazene a distância.

5. Fazer com que o robô caminhe em forma de zigue-zague.

   Posição inicial:
   O robô está localizado na posição de eixos x=0 e y=0.

   Sugestão de solução:
   - Faça com que o robô APRENDA o movimento de zigue-zague, para isso ele deve receber a quantidade de passos. (Dica: APRENDER)

   - Deve REPETIR a quantidade de vezes: (Dica: REPITA)
     - Girar para direita 90 graus
     - Andar para frente 1 passo
     - Girar para a esquerda 90 graus
     - Andar para frente 1 passo
     - Girar para esquerda 90 graus
     - Andar para frente 1 passo
     - Girar para direita 90 graus
     - Andar para frente 1 passo

   Discussão: Quais são as outras possibilidades?

Pós-teste

Perguntas para entrevista estruturada

1. Os comandos foram fáceis de entender? Explique.
2. Teve alguma dificuldade na execução das atividades?
3. Teve algum comando que necessitou e não existe na linguagem?
4. Tem alguma sugestão quanto às atividades?

5. Os comandos estão de acordo com as suas funções?

ATIVIDADES À DISTÂNCIA

Este material contém uma lista com 19 questões de programação, que estão organizadas em blocos de comandos. Após cada bloco, há perguntas relacionadas à usabilidade da linguagem de programação e sobre a satisfação de uso do participante.

Os exercícios de 1 até 5 compreendem, principalmente, os comandos de **movimentação** e **rotação** do robô. Estão descritos no Manual, nas seções 3 e 4, nas páginas 2 e 3.

1. Faça um programa para comandar o robô para andar para frente 10 passos.
2. Faça um programa para comandar o robô para andar para trás 5 passos.
3. Faça um programa para comandar o robô a girar para direita 90 graus.
4. Faça um programa para comandar o robô a girar para esquerda 180 graus.
5. Faça um programa para o robô para andar para frente 20 passos e voltar ao ponto inicial.

Questões de 1 a 7 são relacionadas a usabilidade e satisfação de uso:

1. Os comandos de deslocamento são fáceis de aprender. Marcar um X(xis) na opção desejada.

( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

2. Os comandos de rotação são fáceis de aprender. Marcar um X(xis) na opção desejada.

( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

3. Os comandos de deslocamento são fáceis de usar. Marcar um X(xis) na opção desejada.

( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

4. Os comandos de rotação são fáceis de usar. Marcar um X(xis) na opção desejada.

( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

5. Os comandos de deslocamento são adequados para movimentar o robô. Marcar um X(xis) na opção desejada.

( ) Concordo totalmente

( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

6. Os comandos de rotação são adequados para girar o robô. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

7. Indique se há sugestões de melhoria:
- Para a sintaxe dos comandos de deslocamento:
- Para a sintaxe dos comandos de rotação:
- Para a descrição desses comandos no manual:

Os exercícios de 6 até 7 compreendem, principalmente, os comandos de criação de **variáveis** e **operadores**. Estão descritos no Manual, nas seções 1 e 2, nas páginas 1 e 2.

6. Maria tem 5 bananas e Ana tem 10 bananas. Faça um programa para calcular e mostrar o total de bananas.
7. Faça um programa para ler a nota de 2 alunos e calcular a média.

Questões de 1 a 7 são relacionadas a usabilidade e satisfação de uso:
1. O comando de criação de variável é fácil de aprender. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

2. Os operadores são fáceis de aprender. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

3. O comando de criação de variável é fácil de usar. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

4. Os operadores são fáceis de usar. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente

( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

5. O comando de criação de variável é adequado. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

6. Os operadores são adequados como suporte a expressões matemáticas e comparações lógicas. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

7. Indique se há sugestões de melhoria:
- Para a sintaxe do comando para criação de variável:
- Para a sintaxe dos operadores:
- Para a descrição desses comandos no manual:

Os exercícios de 8 até 10 compreendem, principalmente, os comandos de **áudio** do robô. Estão descritos no Manual, na seção 5, na página 3.

8. Faça o robô falar alguma frase.
9. O som do robô está desligado e você deseja ligar, qual seria a forma correta?
10. O robô possui um valor guardado na variável X. Faça o robô falar o valor da variável X.

Questões de 1 a 7 são relacionadas a usabilidade e satisfação de uso:
1. O comando SOM é fácil de aprender. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

2. O comando de FALAR é fácil de aprender. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

3. O comando SOM é fácil de usar. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente

( ) Discordo totalmente

4. O comando de FALAR é fácil de usar. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

5. O comando SOM é adequado para ligar ou desligar o áudio o robô. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

6. O comando de FALAR é adequado para reproduzir uma frase/palavra/variável do robô. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

7. Indique se há sugestões de melhoria:
- Para a sintaxe do comando som:
- Para a sintaxe do comando de falar:
- Para a descrição desses comandos no manual:

Os exercícios de 11 até 13 compreendem, principalmente, os comandos de informação/percepção do ambiente. Estão descritos no Manual, na seção 6, das páginas 3 até 6.

11. O robô deve explorar livremente o cenário e informar quantos objetos existem da cor vermelha, verde e azul. O robô inicia na posição de eixos x=0 e y=0, virado para o norte. (Dica: utilizar o comando COR)
12. Faça um programa para armazenar a quantidade de passos até um objeto que está a alguns passos à frente direita do robô. (dica: comando distância)
13. O robô iniciou na posição 0,0 virado para o norte, ele caminhou para frente 6 passos. Você deseja saber a posição no eixo x (xis) e no eixo y(ípsilon) que o robô se encontra, qual comando utilizar?

Questões de 1 a 10 são relacionadas a usabilidade e satisfação de uso:
1. O comando COR é fácil de aprender. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

2. O comando DISTÂNCIA é fácil de aprender. Marcar um X(xis) na opção desejada.

( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

3. O comando POS é fácil de aprender. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

4. O comando de COR é fácil de usar. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

5. O comando de DISTÂNCIA é fácil de usar. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

6. O comando de POS é fácil de usar. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

7. O comando COR é adequado para verificar a cor de objetos. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

8. O comando DISTÂNCIA é adequado para verificar a quantidade de passos até um objeto. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

9. O comando POS é adequado para verificar a posição atual do robô nos eixos x (xis) e y (ípsilon) e o ângulo. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

10. Indique se há sugestões de melhoria:
    - Para a sintaxe dos comandos de informação/percepção:
    - Para a descrição desses comandos no manual:

Os exercícios de 14 até 15 compreendem, principalmente, os comandos de **condição/seleção** do robô. Estão descritos no Manual, na seção 7, na página 6.

14. Faça um programa para ler um número. Falar se o número lido é maior que 50 ou menor que 50.
15. Faça um programa para ler a idade de uma pessoa. Falar se a pessoa já pode votar ou se ainda não pode.

Questões de 1 a 4 são relacionadas a usabilidade e satisfação de uso:
1. O comando de seleção é fácil de aprender. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

2. O comando de seleção é fácil de usar. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

3. O comando seleção é adequado para o robô fazer a escolha do que deve executar, de acordo com uma condição estipulada**.** Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

4. Indique se há sugestões de melhoria:
    - Para a sintaxe do comando SE:
    - Para a descrição desse comando no manual:

O exercício 16 compreende, principalmente, os comandos de **repetição** do robô. Estão descritos no Manual, na seção 8, nas páginas 6 e 7.

16. Faça um programa para o robô fazer um quadrado de tamanho de lado 5. Faça o mesmo exercício utilizando o comando REPITA, ENQUANTO e PARA.

Questões de 1 a 10 são relacionadas a usabilidade e satisfação de uso:
1. O comando PARA é fácil de aprender. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

2. O comando REPITA é fácil de aprender. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

3. O comando ENQUANTO é fácil de aprender. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

4. O comando PARA é fácil de usar. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

5. O comando REPITA é fácil de usar. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

6. O comando ENQUANTO é fácil de usar. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

7. O comando ENQUANTO é adequado para repetição de comandos do robô. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

8. O comando REPITA é adequado para repetição de comandos do robô. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

9. O comando PARA é adequado para repetição de comandos do robô. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

10. Indique se há sugestões de melhoria:
- Para a sintaxe dos comandos de repetição:
- Para a descrição desses comandos no manual:

Os exercícios de 17 até 18 compreendem, principalmente, o comando de **procedimento** do robô. Estão descritos no Manual, na seção 9, nas páginas 7 e 8 .

17. Faça um programa para ensinar o robô a fazer a média entre dois números.
18. Faça um programa para ensinar o robô a fazer um retângulo.

Questões relacionadas a usabilidade e satisfação de uso:
1. O comando de declaração de procedimentos é fácil de aprender. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

2. O comando de declaração de procedimentos é fácil de usar. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

3. O comando de procedimento é adequado para declarar procedimentos. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

4. Indique se há sugestões de melhoria:
   - Para a sintaxe do comando aprender:
   - Para a descrição desse comando no manual:

O exercício 19 compreende, principalmente, comandos variados do robô. Estão descritos no Manual, na seção 10, na página 8.

19. Faça com que o Robô simule a subida em uma escada com 4 degraus de tamanhos iguais. O robô deve ficar parado 2 segundos entre a construção de cada degrau. Escolha a posição inicial da escada.

Questões de 1 a 4 são relacionadas a usabilidade e satisfação de uso:
1. O comando ESPERAR é fácil de aprender. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

2. O comando ESPERAR é fácil de usar. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

3. O comando ESPERAR é adequado. Marcar um X(xis) na opção desejada.
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Não sei opinar
( ) Discordo parcialmente
( ) Discordo totalmente

4. Indique se há sugestões de melhoria:
   - Para a sintaxe dos comandos variados:
   - Para a descrição desses comandos no manual:

APÊNDICE F – INSTRUMENTOS DE AVALIAÇÃO COM PROFESSORES DE PROGRAMAÇÃO

Pré-teste

1. Nome:
2. Faixa-etária:
   ( ) de 20 a 30 anos
   ( ) de 31 a 40 anos
   ( ) de 41 a 50 anos
   ( ) de 51 a 60 anos
   ( ) de 61 a 70 anos
   ( ) acima de 70 anos
3. Há quanto tempo ensina programação?
4. Quais as linguagens de programação que ensina?
5. Possui conhecimento em ensino de programação com uso de robótica? Quanto tempo?
6. Já trabalhou com alunos com deficiência visual em disciplina de programação?

Questionário de usabilidade

1. Consistência					
Perguntas	**Concordo totalmente**	**concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
1.1. Os comandos da GoDonnie possuem significados consistentes?					
1.2. Há coerência entre os comandos da GoDonnie e o que eles fazem?					
Observações					
2. Facilidade de uso					
Perguntas	**Concordo totalmente**	**concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
2.1. A sintaxe da linguagem GoDonnie é de fácil entendimento?					
2.2. A semântica da linguagem					

GoDonnie é de fácil entendimento?					
2.3. A sintaxe da GoDonnie possui um conjunto de comandos que permite a resolução de problemas computacionais?					
2.4. A semântica da GoDonnie permite melhor utilizar o conjunto de comandos para resolução de problemas computacionais?					
2.5. É fácil realizar as atividades com a GoDonnie?					
2.6. Os comandos da GoDonnie são simples?					
2.7. Os comandos da GoDonnie são concisos?					
2.8. Os comandos da GoDonnie são fáceis de lembrar como usar?					
2.9. Os comandos da GoDonnie permitem que um problema possa ser resolvido de diferentes formas?					
Observações					
3. Relação entre o sistema e o mundo real					
Perguntas	**Concordo totalmente**	**concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
3.1. A linguagem					

GoDonnie é clara.					
3.2. Os conceitos utilizados nos comandos da GoDonnie são compreensíveis?					
3.3. Existe correspondência entre os comandos da GoDonnie com linguagem similar (linguagem de programação ou com Português, por exemplo)?					
Observações					
4. Ajuda e documentação					
Perguntas	**Concordo totalmente**	**concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
4.1. O manual da GoDonnie é adequado ao aprendizado da mesma?					
4.2. Os problemas apresentados nas atividades do manual da GoDonnie estão adequados para apresentar exemplos de soluções?					
4.3. O manual da GoDonnie é de fácil leitura?					
4.4. O manual da GoDonnie possui uma ordem lógica na apresentação dos comandos?					

Observações					
5. Facilidade de Programação					
Perguntas	**Concordo totalmente**	**concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
5.1. A GoDonnie está adequada para ser utilizada por pessoas com deficiência visual?					
5.2. A GoDonnie está adequada para o ensino de programação para usuários com deficiência visual iniciantes em programação?					
5.3. A GoDonnie está adequada para programação de robô?					
5.4. A GoDonnie não requer muitas ações para resolução de problemas simples?					
Observações					

A versão atual da Godonnie trabalha somente com o tipo inteiro. Desta forma, não podem ser atribuídos valores fracionários (float), alfanuméricos ou alfabéticos. Sabedor de que a GoDonnie é uma linguagem de programação para manipular um robô, qual o impacto dessas decisões de design?

( ) Alto, pois a GoDonnie **DEVERIA** manipular:
- ( ) Dados do tipo float
- ( ) Dados do tipo alfanumérico
- ( ) Dados do tipo alfabético

( ) Intermediário, pois a GoDonnie **PODERIA** manipular:
- ( ) Dados do tipo float

( ) Dados do tipo alfanumérico
( ) Dados do tipo alfabético

( ) Baixo, pois a GoDonnie **NÃO NECESSITA** manipular:
( ) Dados do tipo float
( ) Dados do tipo alfanumérico
( ) Dados do tipo alfabético

( ) Não há impacto.

Na versão atual da GoDonnie, a entrada de dados é realizada pelo robô, que captura as informações sobre ele (localização, ângulo de referência relacionado ao cenário, distância para obstáculos, por exemplo) e sobre o cenário (quantidade, cor e posição de objetos, por exemplo). Desta forma, não há entrada de dados via teclado. Qual o impacto dessa decisão de design?

( ) Alto, pois a GoDonnie **DEVERIA** permitir a entrada de dados via teclado.
( ) Intermediário, pois a GoDonnie **PODERIA** permitir a entrada de dados via teclado.
( ) Baixo, pois a GoDonnie **NÃO NECESSITA** permitir a entrada de dados via teclado.
( ) Não há impacto.

Se desejar, faça comentários gerais sobre a GoDonnie.

APÊNDICE G – INSTRUMENTOS DE AVALIAÇÃO DO AMBIENTE DE PROGRAMAÇÃO COM DEFICIENTES VISUAIS

Atividades presenciais de programação

1 - O robô está em um cenário que possui tamanho de 34 passos na vertical e 34 passos na horizontal. Esse cenário possui alguns objetos que estão distribuídos no espaço. Você tem o desafio de conseguir identificá-los e informar onde estão.
Comentário: Mostrar no mapa tátil.

A. O robô está na posição x= 10, y=0 e virado norte, do lado esquerdo do cenário. Os objetos estão localizados na parte superior do cenário. Precisamos colocar o robô mais para o meio do cenário (lembra que o cenário possui 34 passos de largura). Como fazer isso?
B. Ok, agora estamos no meio do cenário, precisamos saber quantos objetos existem no cenário. Precisamos que o robô esteja virado para o norte, pois os objetos estão na parte superior do cenário. O que devemos fazer? e precisamos saber quantos objetos existem. Como fazer isso?
C. E quantos objetos tem com a cor azul?
D. E com a cor verde?
E. E com cor vermelho?
F. Agora, vamos ver como poderíamos representar esse cenário no mapa tátil. Aqui está o mapa e esses objetos representam os objetos do cenário. Podes distribuí-los? Se precisar espiar novamente o cenário, fique à vontade para usar o computador.

2- Em um curso de programação sobre a GoDonnie, foi ensinado que o comando DISTÂNCIA informa a distância do robô até um objeto. Há um objeto na frente do robô. Verifique a distância do robô para este objeto. Lembre-se que é necessário informar ao robô que deve falar a informação.

A. Desloque o robô até este objeto, sem colidir.
B. Se o robô andasse mais para frente, o que aconteceria? Experimente.
C. Qual a posição do robô no eixo x, y e para onde está virado?

Aplicar as perguntas do questionário:

Categoria do questionário	**Questões**
Facilidade de uso	1.6
Utilidade	2.2 até 2.5
Prevenção e tratamento de erros	3.1 e 3.6
Interface sonora	4.1 até 4.12
Orientação e mobilidade	5.1 até 5.5
Satisfação de uso	8.2

3- Nos exercícios 1 e 2 foram utilizados os comandos ESPIAR, FALAR, PF, GD, GE, POS, DISTÂNCIA. Verifique a explicação desses comandos no manual.
Aplicar as perguntas do questionário:

Categoria do questionário	**Questões**
Ajuda e documentação	7.1 até 7.4

4- Supondo que foi oferecido um curso para ensinar programação usando a GoDonnie. Durante esse curso, um aluno digitou alguns comando. Eu vou ler os comandos para que você digite. Existem erros nessa programação?

A. PF 5 PF3. Existe erro. Qual seria?
B. OK, vamos digitar e ouvir se a explicação sobre o erro.
C. Corrija o programa

Aplicar as perguntas do questionário:

Categoria do questionário	**Questões**
Prevenção e tratamento de erros	3.2 até 3.5

5- No cenário, há objetos nas cores vermelha, verde e azul. Informe se há objetos na cor vermelha, informe a quantidade. Se não houver, informe que não há objetos nessa cor.

6- O robô fez um quadrado de tamanho 3. Mas ficou pequeno. Faça um programa para que o robô faça um quadrado maior, sendo que o tamanho máximo do lado deve ser 6.

A. Tem outra forma de fazer esse quadrado?

Aplicar as perguntas do questionário:

Categoria do questionário	**Questões**
Facilidade de uso	1.7
Utilidade	2.1
Programação	6.5

Aplicar restante do questionário.

1. Facilidade de uso					
Perguntas	**Concordo totalmente**	**concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
1.1. A interface sonora do ambiente é de fácil entendimento?					
1.2. A sintaxe da linguagem GoDonnie é					

de fácil entendimento?					
1.3. A semântica da linguagem GoDonnie é de fácil entendimento?					
1.4. O uso da GoDonnie no ambiente virtual é de fácil entendimento?					
1.5. É fácil realizar as atividades com a GoDonnie no ambiente?					
1.6. As mensagens sonora são de fácil entendimento.					
1.7. Os comandos da GoDonnie permitem que um problema possa ser resolvida de diferentes formas.					
Observações					
2. Utilidade					
Perguntas	**Concordo totalmente**	**concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
2.1. Os comandos da GoDonnie podem ser utilizados para resolver diferentes atividades?					
2.2. O uso da GoDonnie permite compreender objetos em um ambiente virtual?					
2.3. O uso da GoDonnie permite compreender a relação dos objetos com o ambiente virtual?					
2.4. O módulo gráfico do ambiente auxilia no entendimento do ambiente? (usuários com baixa visão)					
2.5. O módulo gráfico do ambiente auxilia no entendimento de onde está o robô? (usuários com baixa visão)					

Observações					
3. Prevenção e tratamento de erros					
Perguntas	**Concordo totalmente**	**concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
3.1. A GoDonnie evita que erros aconteçam?					
3.2. A GoDonnie permite que o usuário compreenda quando ocorre um erro?					
3.3. A GoDonnie permite que o usuário compreenda o tipo de erro que ocorreu?					
3.4. A GoDonnie permite que o usuário compreenda como corrigir erros?					
3.5. A GoDonnie permite que o usuário corrija os erros?					
3.6. As regras de prevenção de erro, por exemplo, de interrupção de deslocamento quando haveria uma colisão futura, são adequadas?					
Observações					
4. Interface sonora					
Perguntas	**Concordo totalmente**	**concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
4.1. Os sons icônicos (passos, rotação e colisão) são intuitivos?					
4.2. Os sons icônicos (passos, rotação e colisão) são necessários para compreensão da execução dos comandos da GoDonnie?					
4.3. Os sons icônicos					

(passos, giros, colisão) são agradáveis?					
4.4. A velocidade dos sons icônicos (passos, giros, colisão) é adequada?					
4.5. As mensagens faladas referentes à execução dos comandos são intuitivas?					
4.6. As mensagens faladas referentes à execução dos comandos são necessárias para compreensão da execução dos comandos da GoDonnie?					
4.7. As mensagens faladas referentes à execução dos comandos são agradáveis?					
4.8. A velocidade das mensagens faladas referentes à execução dos comandos é adequada?					
4.9. As mensagens icônicas auxiliam o entendimento das mensagens faladas?					
4.10. As mensagens icônicas são necessárias para um melhor entendimento das mensagens faladas?					
4.11. As mensagens faladas auxiliam o entendimento das mensagens icônicas?					
4.12. As mensagens faladas são necessárias para um melhor entendimento das mensagens icônicas?					
Observações					
5. Orientação e Mobilidade					

Perguntas	**Concordo totalmente**	**concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
5.1. A GoDonnie é um recurso que pode ser utilizado previamente pelo usuário para explorar um ambiente?					
5.2. A GoDonnie permite compreender a relação dos objetos com o robô?					
5.3. A GoDonnie permite compreender onde está o robô no ambiente?					
5.4. A GoDonnie permite que o usuário compreenda a direção onde está o robô no ambiente?					
5.5. A GoDonnie permite que o usuário externalize seu plano de ação mental para deslocar o robô de um ponto a outro do ambiente?					
Observações					
6. Programação					
Perguntas	**Concordo totalmente**	**concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
6.1. A GoDonnie é adequada para ser utilizada por pessoas com deficiência visual?					
6.2. A GoDonnie é adequada para o ensino de programação para iniciantes em programação?					
6.3. A GoDonnie é adequada para programação de robô?					
6.4. A GoDonnie não					

requer muitas ações para realização de problemas simples?					
6.5. A GoDonnie permite diferentes formas de resolução de um mesmo problema?					
Observações					
7. Ajuda e documentação					
Perguntas	**Concordo totalmente**	**concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
7.1. O manual da GoDonnie é adequado ao aprendizado da mesma?					
7.2. Os exemplos de atividades do manual da GoDonnie são úteis ao aprendizado dos comandos?					
7.3. O manual da GoDonnie é de fácil leitura?					
7.4. O manual da GoDonnie possui uma ordem lógica na apresentação dos comandos?					
Observações.					
8. Satisfação de uso					
Perguntas	**Concordo totalmente**	**concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
8.1. É agradável utilizar a GoDonnie para aprender a programação?					
8.2. O uso da GoDonnie é interessante para compreender um espaço real?					
8.3. O uso da GoDonnie é prazeroso?					

Observações.

Sobre Decisões de design

1. **A versão atual da Godonnie manipula somente números inteiros. Desta forma, não trabalha com números com vírgula e nem com letras. Por exemplo, não é possível mandar o robô andar para frente 2 passos e meio, e nem solicitar que seja armazenado um nome em uma variável. Você considera que o impacto desse funcionamento é:**

( ) Alto, pois a GoDonnie **DEVERIA** manipular:
- ( ) Dados do tipo float (números com vírgula)
- ( ) Dados do tipo alfanumérico (letras com números)
- ( ) Dados do tipo alfabético (letras)

( ) Intermediário, pois a GoDonnie **PODERIA** manipular:
- ( ) Dados do tipo float (números com vírgula)
- ( ) Dados do tipo alfanumérico (letras com números)
- ( ) Dados do tipo alfabético (letras)

( ) Baixo, pois a GoDonnie **NÃO NECESSITA** manipular:
- ( ) Dados do tipo float (números com vírgula)
- ( ) Dados do tipo alfanumérico (letras com números)
- ( ) Dados do tipo alfabético (letras)

( ) Não há impacto.

2. Na versão atual da GoDonnie, o robô identifica sua localização, ângulos, distância para objetos, cores dos objetos, dentre outras informações. Desta forma, o usuário não informa esses dados para o robô porque o robô captura essas informações do cenário. Diferente da GoDonnie, há linguagens de programação em que o usuário pode informar dados de entrada. Por exemplo, o programador pode fazer um programa que pergunte a idade de uma pessoa e, quando a pessoa digitar a idade pelo teclado, o robô pode andar a mesma quantidade de passos. Com a GoDonnie, isso não é possível de ser feito. Qual o impacto dessa restrição?

( ) Alto, pois a GoDonnie **DEVERIA** permitir a entrada de dados via teclado.
( ) Intermediário, pois a GoDonnie **PODERIA** permitir a entrada de dados via teclado.
( ) Baixo, pois a GoDonnie **NÃO NECESSITA** permitir a entrada de dados via teclado.
( ) Não há impacto.

Sobre a GoDonnie

1. A programação de robôs vem sendo utilizada como estratégia para a aprendizagem de programação por usuários iniciantes. Você considera que a GoDonnie pode ser utilizada por pessoas com deficiência visual, que sejam iniciantes em programação? Comente sobre isso.

2. Você recomendaria o uso da GoDonnie para outras pessoas que desejam aprender a programar robô? Comente sobre isso.

3. Você gostaria de continuar utilizando a GoDonnie? Comente sobre isso.

Se desejar, faça outros comentários sobre a GoDonnie.

APÊNDICE H – PROCESSO DE TREINAMENTO DO UBUNTU COM O LEITOR ORCA COM PESSOAS COM DEFICIÊNCIA VISUAL

1 dia (16/01/2017):

- Expliquei aos usuários P1 e P2 que o projeto utiliza o sistema operacional Ubuntu, que é um pouco diferente a forma de utilização dele em relação ao Windows. Informei que o Orca é o leitor de telas do ubuntu, e que teríamos que aprender a usá-lo e configurá-lo de acordo com a preferência deles.
- Mostrei para eles o notebook do projeto e o leitor de telas Orca, mostrei executando. O P1 nunca tinha usado o Ubuntu, enquanto que o P2 já estava mais familiarizado com o ambiente. P1 relatou no dia, que achou mais complexo do que o Windows, na primeira impressão. Os dois usuários informaram que o melhor seria que o Orca iniciasse já na abertura do Ubuntu, para que eles não precisassem de ajuda. Hoje existe o seguinte procedimento para ligar o Orca:

1. Ligar o computador
2. Logar
3. abrir o terminal
4. Digitar orca --replace
5. O Terminal tem que ficar aberto, senão o Orca para de funcionar.
6. Para configurar a linguagem e velocidade do Orca, deve-se clicar em insert + espaço. Na aba Voice tem essas configurações.

- Os participantes preferiram a linguagem em português, quanto a velocidade relataram estar tudo bem.
- Pedi aos participantes para Pesquisarem as teclas de atalho do leitor de telas Orca e do sistema operacional Ubuntu (Linux). como por exemplo:
    - Como abrir editor de textos no ubuntu. colocar as teclas de atalhos utilizadas
    - Como salvar o arquivo no editor de textos.
    - Como realizar uma pesquisa no ubuntu no editor de textos.
    - Qualquer outra atividade que vocês costumam realizar no Windows, pesquisar como realizar no Ubuntu.
    - Por favor, pesquisem e façam um manual para vocês.
    - Pesquisem como abrir arquivo no Ubuntu com o terminal.
    - Tragam as suas pesquisas para a próxima reunião.

2 dia (19/01/2017):

- Inicializei o notebook e o leitor de telas para os participantes.
- Pedi que eles treinassem no notebook os atalhos que foram pesquisados por eles. Os usuários treinaram durante 1 hora em conjunto.

3 dia (20/01/2017):

- Somente P1 estava presente, pedi que ele treinasse a utilização de editor de textos no Ubuntu.
- Pedi que ele realizasse as mesmas listas de atividades que ele já havia feito no Windows. Para que se ambientasse a escrita e leitura no Editor do Ubuntu.
- Ele teve algumas dúvidas na utilização, alguns sites em que ele pesquisou informam algumas teclas de atalho que não funcionam no Ubuntu. Pedi que pesquisasse mais a respeito e que testasse mais.

3 dia (23/01/2017):

- Pedi que o P1 continuasse realizando a lista de atividades**.**
- Neste dia o P2 participou, pedi que ele auxiliasse o P1 e que eles treinassem juntos.
- Pedi que eles começassem a ver como abrir um arquivo no Terminal. E vissem como funciona o Terminal.

4 dia (24/01/2017):

- Pedi que o P1 continuasse realizando a lista de atividades e treinando a utilização do ubuntu**.**
- P1 informou que estava melhor adaptado ao Ubuntu e leitor Orca. Informou que as teclas de atalho que utiliza no Editor do Windows são bem parecidas no Editor do Ubuntu.
- Informou que ainda tem um pouco de dificuldade de encontrar tutoriais na internet com mais informações, pois a maioria utiliza figuras para demonstração. Mas de toda forma, está se virando ao utilizar o Ubuntu e Editor.

5 dia (26/01/2017):

- Solicitei ao P1 que continuasse realizando a lista de atividades e treinando a utilização do ubuntu**.**
- P1 está mais seguro na utilização do Ubuntu.

6 dia (27/01/2017):

- Solicitei que o P1 continuasse treinando.
- Encaminhei por e-mail para os dois usuários o questionário abaixo.

1. Qual foi a primeira impressão sobre o sistema operacional Ubuntu e sobre o leitor de telas Orca? Descreva.
2. E depois de utilizar uns dias, qual a impressão? Descreva.
3. O que vocês aprenderam até agora sobre o ubuntu e Orca? Descreva.
4. Quais foram as dificuldades enfrentadas? Descreva.
5. Na sua opinião, qual a maior diferença entre ubuntu e Windows? Você acha que são parecidos? Descreva o que acha parecido e o que é diferente.
6. Na sua opinião, qual a maior diferença entre Orca e NVDA? Você acha que são parecidos? Descreva o que acha parecido e o que é diferente.
7. Você se sente apto a utilizar o editor de textos no Ubuntu? Por exemplo, vocês acham que precisam treinar mais um pouco ou já se sentem aptos a utilizar o editor de textos?
8. Você se sente apto a utilizar o Terminal do Ubuntu? Por exemplo, vocês acham que precisam treinar mais um pouco ou já se sentem aptos a utilizar o terminal?
9. Vocês se sentem ambientados ao Ubuntu e Orca? Por exemplo, vocês acham que precisam treinar mais um pouco ou já se sentem aptos a utilizar o ubuntu e orca?

Respostas do P1:

1. Qual foi a primeira impressão sobre o sistema operacional Ubuntu e sobre o leitor de telas Orca? Descreva.
Sobre esse sistema operacional ubuntu, imaginei que seria fácil de lidar, como os demais sistemas que já conheço como por exemplo, o windows parecia ser fácil somente no início depois teve alguns conflitos que ao passar do tempo começou a ser resolvidos lidando bastante com as práticas, mas a primeira impressão é complicada mas com o tempo vai se acostumando com a máquina, já sobre o orca parecia ser parecia com o nvda no início com poucas diferenças o único defeito é que às vezes para por isso, é preciso reiniciar o computador, mas a impressão é boa também.
As minhas impressões também pensei que seria um sistema fácil de lidar com atalhos que as vezes não se bate com os atalhos da net, por isso, as vezes me irrita essa coisa por não obedecer as vezes os atalhos solicitados.

2. E depois de utilizar uns dias, qual a impressão? Descreva.
Claro depois se vai conhecendo a máquina a gente vai se acostumando com ela com suas limitações e depois de bastante práticas por aqui é fácil de lidar nisso, é claro no início parece que vai ser difícil mas com o tempo se vai pegando o jeito.

Pois o que ajudou também as manhas que já sei do word isso que me ajudou nas escritas no editor do ubuntu, já o ledor de tela do orca é também se acostuma com esse tipo de voz.

3. O que vocês aprenderam até agora sobre o ubuntu e Orca? Descreva.
Escrever os comandos do manual e dos exercícios anteriores das listas 1 a 7 Aprendi o atalhos para achar arquivos salvar arquivos, entrar no terminal, andar pelo terminal e Lidar entre as plataformas do sistema. Isso tudo com horas e horas de práticas.

4. Quais foram as dificuldades enfrentadas? Descreva.
Primeiramente a lidar com esse novo sistema como escrever, como salvar as coisas que são solicitados no geral, o mais complicado é ficar horas pesquisando para lidar aqui e depois acha vem aplicar aqui verifica que nada dá certo que muitas coisas a gente se aprende por nós mesmos, e sem dizer que se acha na net na grande maioria não ajuda muito é só um suporte. Mas minhas dificuldades são lidar aqui no terminal mesmo para transferir os arquivos para seu devido lugar certo.

5. Na sua opinião, qual a maior diferença entre ubuntu e Windows? Você acha que são parecidos? Descreva o que acha parecido e o que é diferente.
Não acho nada parecido claro algumas igualdades mas poucas, nas igualdades é bem parecido mesmo nos comandos e tal para escrever algumas semelhanças com o word, a grande diferença é que o ledor de tela fala na grande maioria das suas funções ele fala em inglês.

6. Na sua opinião, qual a maior diferença entre Orca e NVDA? Você acha que são parecidos? Descreva o que acha parecido e o que é diferente.
No orca é mais eletrônico ele fala e m inglês os comandos o nvda é também tem as mesmas funções como o do orca, no orca ele não dá certos comandos como no nvda de como dar um sinal quando é letra minúscula e maiúscula algumas teclas o orca não fala se fala fala em inglês. Em certos atalhos não fala o início da linha e tal. O nvda fala mais mas em português

7. Você se sente apto a utilizar o editor de textos no Ubuntu? Por exemplo, vocês acham que precisam treinar mais um pouco ou já se sentem aptos a utilizar o editor de textos?
Acredito que sim

8. Você se sente apto a utilizar o Terminal do Ubuntu? Por exemplo, vocês acham que precisam treinar mais um pouco ou já se sentem aptos a utilizar o terminal?
Acredito que também sim e acredito também que a pesquisadora se sentando ao meu lado para me explicar vai render muito mais ainda.

9. Vocês se sentem ambientados ao Ubuntu e Orca? Por exemplo, vocês acham que precisam treinar mais um pouco ou já se sentem aptos a utilizar o ubuntu e orca?
Também sim já podemos começar a utilizar

P1 enviou mais alguns relatos no dia (30/01/2017)

Comecei a lidar no sistema do ubuntu e orca, primeiramente devo procurar um arquivo para poder trabalhar, por exemplo o que comecei a trabalhar sobre que deveria digitar os exercícios das listas já trabalhadas em atividades anteriores, ou seja, digitei os exercícios de acordo com o que eu lembrava, é claro que alguns eu não lembrei de início precisei dar uma revisada mas em alguns o erro que cometi em certos casos foi erro bem banal sem muito significados. Em alguns fiz

certas modificações para não ficar bem igual aos anteriores como por exemplo os casos de criar variáveis entre outros. No meu caso foi muito bom relembrar esses exercícios para ver o que eu lembrava pois esses exercícios são bons de fazer quando se sabe.

Depois dessas atividades comecei a lidar como é criar um arquivo nesse sistema como como vou explicar a seguir:
Aperto no menu, eu nesse caso vou chama-lo de menu mas no original não é esse nome do botão nesse sistema.
Obs.: **é a terceira tecla do teclado da esquerda para a direita é a terceira tecla certo que vou chamar de menu.**
Ou também a segunda tecla da tecla de espaço ou seja da direita para a esquerda respectivamente

Logo após isso devo **apertar no botão tab e assim respectivamente para achar o arquivo que eu necessito.**
Logo que acho devo clicar nele e ele abre instantaneamente e comecei a explorar mais essa ferramenta dentro do sistema ubuntu, logo que comecei a lidar no início a**chei que seria difícil lidar nisso mas que nada**, foi até bem fácil é claro com a pesquisadora ao meu lado para tirar minhas possíveis dúvidas desse novo sistema para mim. Também gostaria de apontar que, que durante muitos dias eu pesquisei na internet algum arquivo ou site que pudesse me ajudar nisso, como por acaso me dar sugestões de atalhos dentro do sistema terminal ou qualquer outro coisa do tipo, nisso apareceu alguns resultados e pesquisei à questão que em casos não foram bem acessíveis mas foi bem difícil achar algum fácil de ler para uma leitura bem tranquila, o mais fácil que encontrei foi um de um site que era: **ubuntu para iniciantes**, **acredito que era um blog**, **mesmo assim não me ajudou muito.**

Agora vou começar a desenvolver como faz para abrir o famoso terminal é somente clicar em **control mais anti mais t essas 3 teclas é possível abri-lo respectivamente.**
Depois posso andar por ele, e também posso apagar algo do terminal e escrever por cima dele o que quero fazer por ali.

Lembra das teclas que citei anteriormente sobre o menu e tab, dando esse comando também posso digitar os iniciais do que desejo para aparecer o que quero, ou seja, o arquivo que desejo abrir para trabalhar respectivamente.
Com esse arquivo posso o em casos elaborar, repetir e recordar atividades ou revisões para melhor práticas para lidar nesse sistema.

Comecei a melhor praticar no editor de texto desse sistema, para começar gostaria de lembrar que para abrir o sistema para **abrir o campo de busca na procura de um arquivo devo apertar no menu somente um clique se abre ali o campo, e a partir daí posso escrever o** arquivo que quero que abra.
Obs.: só um clique rápido

Ou seja, agora durante as minhas escritas nesse sistema, ou melhor no editor de texto do ubuntu, observei que estou escrevendo meus comandos que quero que o robô faça, e partir daqui notei que **quando escrevo certas palavras com acento o ledor de tela não fala o acento somente fala é letra em questão.**
Como por exemplo:
Você, época
E o orca fala somente e
Se eu voltar com a seta ele fala ê
Para o é, é a mesma coisa, ou seja, ele fala respectivamente é se voltar com a seta para trás

Agora no caso de água e coração.
Em á ele não fala o nome do acento somente fala a quando volto com a seta para trás e aí sim ele fala o nome da letra e seu nome do acento.
Para a próxima palavra é a mesma coisa primeiramente fala no orca só a letra e se volto aí sim ele fala a letra e o seu nome.

Por fim outra observação é que, o orca também não fala quando estou digitando alguma palavra com ç, por exemplo escrevi faça digitei ele lê, o ledor de tela bem assim: "f" "a" "a"
Isso que cliquei no ç e ele não leu, como já expliquei anteriormente ele lê somente se eu voltar para trás com a seta certo.

Só achei melhor anotar essa questão pois acredito que muitas pessoas que vão usar com ledor de tela vão perceber essa falha por mim é tranquilo, só levantei esta questão por causa das pessoas que estiverem usando e serem recém alfabetizadas entre outros casos.
claro amanhã vou dar mais umas melhoradas nas escritas, desculpa qualquer coisa e qualquer dúvida me pergunta certo.

Resposta do P2:

1. Qual foi a primeira impressão sobre o sistema operacional Ubuntu e sobre o leitor de telas Orca? Descreva.
Em um primeiro momento o sistema parece bem diferente do que eu estou acostumado como usuário do Windows, entretanto como usuário avançado me pareceu bastante fácil a utilização do sistema como um todo, sendo a parte um pouco mais difícil a nomenclatura dos comandos e apps, mas isto é facilmente contornado caso seja fornecido um manual para o usuário.
Em relação ao app ORCA, foi um pouco complicado visto que só foi o meu primeiro contato com este app.
A maior dificuldade acontece quando ainda é necessário ajustar alguma configuração básica deste app,, em caso contrário é até bem similar a utilização do NVDA,. A não ser pelo fato que de eu, particularmente, achei o ORCA um pouco mais lento que o seu similar para windows (me refiro ao tempo de resposta)

2. E depois de utilizar uns dias, qual a impressão? Descreva.
Após utilizar o sistema e o leitor de tela por algumas sessões o usuário acaba facilmente se acostumando com os comandos e a nomenclatura do linux

3. O que vocês aprenderam até agora sobre o ubuntu e Orca? Descreva.
Ubuntu
- logar no sistema
- carregar o ORCA
- carregar o terminal
- carregar o editor de texto
- digitar textos
- gravar arquivos texto
- procurar por arquivos
- editar um arquivo existente
ORCA
- carregar o orca
- atualizar as configurações
-- Neste caso ainda é necessário um pouco mais de prática

4. Quais foram as dificuldades enfrentadas? Descreva.
A princípio algumas diferenças operacionais nativas do sistema, mas nada intransponível. A maior dificuldade é quando se tem que alterar alguma configuração do ORCA, se o computador já viesse com este app configurado seria bem mais fácil.

5. Na sua opinião, qual a maior diferença entre ubuntu e Windows? Você acha que são parecidos? Descreva o que acha parecido e o que é diferente.
Os sistemas operacionais são bem semelhantes, o que mais diferencia é que no ubuntu não temos a mesma fluência por apenas não estarmos tão acostumados a utilizar. O que se faz de diferente é apenas a chamada dos programas que no Ubuntu o fizemos por comandos digitados (no meu caso).

6. Na sua opinião, qual a maior diferença entre Orca e NVDA? Você acha que são parecidos? Descreva o que acha parecido e o que é diferente.
A funcionalidade dos dois é a mesma, o que eu achei neste primeiro contato é que o tempo de resposta do ORCA é um pouco mais lento do que o NVDA.
Devido a pouca utilização do orca ainda acho um pouco complicado realizar as configurações básicas bem como carrega-lo, Caso ele esteja previamente instalado e inicializar junto com o sistema seria uma facilidade a mais para o usuário final.

7. Você se sente apto a utilizar o editor de textos no Ubuntu? Por exemplo, vocês acham que precisam treinar mais
um pouco ou já se sentem aptos a utilizar o editor de textos?
Sim, estou apto a usar, seria melhor "dar uma treinadinha" antes para garantir, mas é super simples.

8. Você se sente apto a utilizar o Terminal do Ubuntu? Por exemplo, vocês acham que precisam treinar mais um pouco ou já se sentem aptos a utilizar o terminal?
Aparentemente sim, acho que mais um sessão de 4 horas seria suficiente para ter a fluência necessária.

9. Vocês se sentem ambientados ao Ubuntu e Orca? Por exemplo, vocês acham que precisam treinar mais um pouco ou já se sentem aptos a utilizar o ubuntu e orca?
-Resposta igual a pergunta anterior.

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria Acadêmica
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: proacad@pucrs.br
Site: www.pucrs.br/proacad