

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TANIA SANAI SHIMABUKURO

**IDENTIFICAÇÃO DE CARACTERÍSTICAS DO AMBIENTE NO
*SOFTWARE LOGIKID***

BAURU
Dezembro/2020

TANIA SANAI SHIMABUKURO

**IDENTIFICAÇÃO DE CARACTERÍSTICAS DO AMBIENTE NO
*SOFTWARE LOGIKID***

Trabalho de Conclusão de Curso do Curso
de Ciência da Computação da Universidade
Estadual Paulista “Júlio de Mesquita Filho”,
Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Dr. Renê Pegoraro

BAURU
Dezembro/2020

Tania Sanai Shimabukuro Identificação de Características do Ambiente
no *Software LOGIKID*/ Tania Sanai Shimabukuro. – Bauru, Dezembro/2020-
46 p. : il. (algumas color.) ; 30 cm.
Orientador: Prof. Dr. Renê Pegoraro
Trabalho de Conclusão de Curso – Universidade Estadual Paulista “Júlio de
Mesquita Filho”
Faculdade de Ciências
Ciência da Computação, Dezembro/2020.
1. AEDROMO 2. LOGIKID 3. Robótica Educacional

Tania Sanai Shimabukuro

Identificação de Características do Ambiente no *Software LOGIKID*

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. Renê Pegoraro
Orientador
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Prof. Dr. Simone Prado
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Prof. Dr. Wilson Massashiro Yonezawa
Departamento de Computação
Faculdade de Ciências
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Bauru, 15 de Dezembro de 2020.

Dedico este trabalho à minha irmã que me incentivou e amparou durante toda a minha jornada. Dedico também aos meus avós que não tiveram o orgulho de me verem concluindo a graduação, mas que sempre estiveram em meu coração e pensamento durante toda a minha vida.

Agradecimentos

Agradeço minha família que sempre me apoiou e me incentivou à chegar até aqui. Agradeço aos meus pais pela oportunidade de estudar e fazerem o máximo deles para que eu pudesse terminar minha graduação. Agradeço especialmente à minha irmã, Mariana Akemi Shimabukuro, por ser uma grande inspiração e exemplo a ser seguido, que mesmo longe se fez presente durante todos esses anos, me ajudando, tirando dúvidas e ouvindo meus desabafos.

Agradeço ao Professor Dr Renê Pegoraro por ter sido um orientador incrível e me ensinado tanto nos últimos quatro anos, por ter me auxiliado e explicado repetidas vezes para que eu não ficasse com dúvida alguma.

Agradeço também à todos os professores do Departamento de Computação e do Departamento de Matemática que se dedicaram para que eu pudesse ter uma educação de qualidade.

Agradeço aos meus amigos, tanto os de sala quanto os outros, por deixarem meus dias mais leves, me ajudarem nos tempos de crise e por terem feito parte dessa etapa tão importante da minha vida. Agradeço, especialmente, ao Pedro Barros por ter sido o melhor companheiro de turma que eu pude ter ao meu lado durante treze anos, por ter me ajudado sempre que fiquei doente e por ter se preocupado e cuidado tanto de mim. Outro agradecimento especial à Giulia Ito por ser uma amiga incrível, que cuida tanto de mim, da minha saúde mental, física e vocal.

“Educar verdadeiramente não é ensinar fatos novos ou enumerar fórmulas prontas, mas sim preparar a mente para pensar.” (Albert Einstein)

Resumo

O AEDROMO (Ambiente Experimental Didático com Robôs Móveis) é um ambiente versátil capaz de suportar diversos experimentos. O LOGIKID é um software, que com o auxílio do AEDROMO, facilita o ensino da lógica de programação para crianças e adolescentes. Este projeto de pesquisa visou desenvolver melhorias no *software* LOGIKID, como passagem de parâmetro por referência e manuseio de vetores e matrizes em relação à linguagem do *software* e exibição do comando sendo criado e exibição de apenas uma pergunta por vez relacionado à interface gráfica com o usuário. Essas melhorias foram implementadas a fim de aumentar o potencial educacional do mesmo.

Palavras-chave: AEDROMO; LOGIKID; Robótica Educacional.

Abstract

AEDROMO (Ambiente Experimental Didático com Robôs Móveis) is a versatile environment designed to support a wide variety of experiments. The LOGIKID is a software application that when combined with the AEDROMO facilitates the programming logic and concepts to children and teenagers. This research project has implemented several improvements to the LOGIKID including the implementation of passing arguments to a function by reference and the manipulation of multidimensional arrays related to LOGIKID's language and live updates on the commands being executed and display one question at a time related to the user interface. These improvements were implemented in order to further expand AEDROMO's versatility.

Keywords: AEDROMO; LOGIKID; Educational Robotics.

Lista de figuras

Figura 1 – Blocos do SCRATCH e fragmentos de algoritmo.	16
Figura 2 – Jogo LIGHTBOT.	17
Figura 3 – As peças de um kit LEGO MINDSTORMS EV3.	19
Figura 4 – Página inicial do LEGO MINDSTORMS EV3 <i>Home Edition</i> , onde exibe algumas possibilidades de robôs projetáveis do Sistema LEGO.	19
Figura 5 – Página de programação do LEGO MINDSTORMS EV3 <i>Home Edition</i> .	20
Figura 6 – Interface de programação do ambiente de desenvolvimento LEGAL.	21
Figura 7 – Ambiente do AEDROMO.	22
Figura 8 – Estrutura da linguagem do software LOGIKID.	25
Figura 9 – Função <i>Soma</i> em Java.	26
Figura 10 – Função <i>Soma</i> na linguagem do LOGIKID.	26
Figura 11 – Função <i>testeSoma</i> responsável por chamar a função <i>Soma</i> na linguagem do LOGIKID.	27
Figura 12 – Exemplo da definição de uma função chamada <i>deslocaRobo</i> definida na linguagem do LOGIKID que chama as funções <i>Estado</i> e <i>irPara</i> .	27
Figura 13 – Função com recursividade.	28
Figura 14 – Diagrama do interpretador da linguagem do LOGIKID.	29
Figura 15 – Diagrama do interpretador da linguagem do LOGIKID - Comando <i>se</i> e <i>repete</i> .	30
Figura 16 – Interface gráfica com o usuário - LOGIKID.	31
Figura 17 – Janela para completar as informações da instrução a ser adicionada.	32
Figura 18 – O robô digitalizado - versão final.	33
Figura 19 – Interface do LOGIKID desenvolvida no Processing.	33
Figura 20 – Caixa de diálogo exibida para confirmar a instrução que será adicionada ao programa.	35
Figura 21 – Tela do menu inicial do LOGIKID.	36
Figura 22 – Tela exibida após o menu inicial.	37
Figura 23 – Tela exibida após a seleção da opção "Ir Para".	38
Figura 24 – Tela exibida após a seleção da opção "Ponto (x, y)".	39
Figura 25 – Tela exibida após a inserção dos valores de X e Y.	40
Figura 26 – Tela exibida após a confirmação da instrução.	41
Figura 27 – Simulador do AEDROMO.	41

Lista de abreviaturas e siglas

AEDROMO	Ambiente Experimental e Didático com Robôs Móveis
LISDI	aboratório de Integração de Sistemas e Dispositivos Inteligentes
MIT	<i>Massachusetts Institute of Technology</i>
UDP/IP	<i>User Datagram Protocol over Internet Protocol</i>
UNESP	Universidade Estadual Paulista "Júlio de Mesquita Filho"

Sumário

1	INTRODUÇÃO	12
1.1	Objetivos	13
1.1.1	Objetivos Específicos	13
1.2	Estrutura da Monografia	13
2	FUNDAMENTAÇÃO TEÓRICA	15
2.1	Software LOGO	15
2.1.1	Pesquisas com <i>software</i> LOGO	15
2.2	Software SCRATCH	16
2.3	Software LIGHTBOT	17
2.4	LEGO MINDSTORMS	18
2.5	LEGAL	20
2.6	AEDROMO	21
3	DESENVOLVIMENTO DO PROJETO	24
3.1	Materiais	24
3.2	Linguagem do <i>software</i> LOGIKID	24
3.3	Interface com o usuário	31
3.3.1	Reestruturação do LOGIKID	32
3.4	Exemplo de utilização	36
4	CONTRIBUIÇÕES	42
5	CONCLUSÃO	44
	REFERÊNCIAS	45

1 Introdução

A tecnologia se desenvolve mais a cada dia e está continuamente presente no cotidiano de toda a população. Ademais, diversas pesquisas indicam o crescimento do mercado de robôs, inclusive os destinados à educação, e essa é uma tendência que deve continuar nos próximos anos (BENITTI, 2012).

A Robótica Educacional pode ser definida como “um conjunto de recursos que visa o aprendizado científico e tecnológico integrado às demais áreas do conhecimento” (LOPES, 2008, p. 41).

O AEDROMO é um Ambiente Experimental e Didático com Robôs Móveis, formado por uma área de trabalho, objetos passivos, dois robôs (ou mais), dois computadores, uma câmera global do tipo *webcam* e um transmissor. A flexibilidade de adaptação, uso por várias disciplinas, e o baixo custo são as premissas na concepção e desenvolvimento deste ambiente. Os experimentos, neste ambiente, podem ser motivados para fins de pesquisa, aprendizagem ou, entretenimento (FILHO et al., 2006).

Sabendo disso, uma importante ferramenta para a área da robótica educacional é o AEDROMO. A sua capacidade de reconhecer pelo menos dois robôs e vários objetos coloridos permite que atividades com sistemas complementares, isto é programas cliente, sejam realizadas, algo que ocorre com o *software* LOGIKID.

O Trabalho de Conclusão de Curso de Lisboa (2015), o LOGIKID, é um *software* que se baseia em controlar um robô por meio de uma linguagem simples, não precisando ter conhecimento de uma linguagem de programação. Dessa forma, é facilitado o uso do mesmo por crianças e adolescentes. Ele pode ser executado no ambiente físico, AEDROMO, ou no virtual (um simulador). Essa ferramenta possibilita que o usuário veja a execução por completo de todos os comandos dados em seu programa, permitindo maior compreensão da lógica de programação.

Ao utilizar robôs físicos, o LOGIKID se aproxima à projetos como o do LEGO MINDSTORMS, inspirado por Seymour Papert e desenvolvido em uma parceria entre o Media Lab do *Massachusetts Institute of Technology* (MIT) e o Lego Group (FRIEDRICH et al., 2012), que se baseia no uso de blocos para configurar os comandos a serem realizados por seu robô, e do LEGAL¹, desenvolvido pela empresa PETE², que é uma linguagem de programação em português e agrupa conceitos de valor social, como a utilização de “por favor” e “obrigado” para iniciar e finalizar seus códigos, respectivamente.

O *software* LOGIKID se comunica com o usuário através de perguntas que têm alguns

¹ Disponível em: <<https://www.pete.com.br/legal/>>. Acesso em: 02 dez. 2020.

² Disponível em: <<https://www.pete.com.br/>>. Acesso em: 02 dez. 2020.

botões para sua resposta. Assim, o usuário escolhe a opção desejada a ser realizada pelo robô. Essa interação é dada por textos.

A primeira versão do LOGIKID não permitia que o robô percebesse os obstáculos e as características ao seu redor no ambiente em que estava posicionado. Dessa forma, ele era incapaz de identificar colisões ou ações transformadoras neste ambiente, fazendo os experimentos serem puramente deliberativos.

Este trabalho atualizou a linguagem do software de modo que as características do ambiente possam ser identificadas, expandindo as possibilidades e os limites a serem explorados através do LOGIKID. Um exemplo de aplicação a partir dessa atualização é a capacidade do robô executar uma determinada função, como andar, até que uma parede seja encontrada e, então, ele mudar sua direção para evitar colisões. Além disso, a interface gráfica com o usuário foi reconstruída para ser mais intuitiva, dinâmica e cativante e para capacitá-la ao uso das novas particularidades da linguagem.

1.1 Objetivos

O objetivo deste trabalho é atualizar o software de ensino de lógica de programação para crianças e adolescentes chamado LOGIKID, a fim de possibilitar que o mesmo identifique as características de seu ambiente.

1.1.1 Objetivos Específicos

- Ajustes nas funcionalidades existentes no *software*;
- Implementar novas funcionalidades de identificação de características do ambiente à linguagem de programação do LOGIKID;
- Reestruturar a interface com o usuário para possibilitar o uso das novas características da linguagem, como manipulação de vetores e matrizes, por exemplo.

1.2 Estrutura da Monografia

Esta monografia está estruturada da seguinte forma:

- Fundamentação Teórica: apresenta sínteses de trabalhos, utilizando alguns conceitos de robótica e educação, cujos objetivos englobaram não apenas o ensino de computação, mas também o ensino multidisciplinar por intermédio da computação;
- Desenvolvimento do projeto: relata sobre a reestruturação do software que é utilizado como técnica complementar no ensino da lógica de programação para crianças e adolescentes;

- Contribuições: apresenta as contribuições realizadas ao ambiente LOGIKID a partir do desenvolvimento do projeto;
- Conclusões e considerações finais: apresenta as conclusões e considerações finais da pesquisa.

2 Fundamentação Teórica

Nesta seção estão apresentados alguns projetos correlatos, dos quais os objetivos compreendem o ensino de conceitos multidisciplinares através da computação, além de conceitos da computação em si. Dessa forma, apresenta-se a seguir um levantamento bibliográfico de alguns softwares amplamente utilizados no ambiente educacional como LOGO, SCRATCH e LIGHTBOT.

2.1 *Software* LOGO

O LOGO foi desenvolvido no Instituto de Tecnologia de Massachussets (MIT) por Seymour Papert, por volta de 1960. O diferencial do LOGO é o fato de possuir objetivo indeterminado, o que permite sua utilização em inúmeras atividades diferentes ([FERRUZZI, 2001](#)).

Essa linguagem de programação tem como público alvo crianças a partir dos 5 (cinco) anos de idade. Há uma tartaruga como personagem para executar as instruções criadas. Esta forma de comunicação entre o computador e o seu usuário se diferencia das outras por ter sido desenvolvida para que crianças pudessem aprender por meio dela, além de apresentar uma filosofia de educação não diretiva, inspirada por Piaget, na qual o usuário aprende ao explorar o seu ambiente e cria suas próprias regras ([LOGO, 2009](#)).

2.1.1 Pesquisas com *software* LOGO

[Souza, Reis e Pereira \(2012\)](#) desenvolveram um projeto de extensão realizado em duas escolas públicas de São João del-Rei. Eles constataram que após a aplicação do projeto, utilizando o *software* Kturtle nas aulas de linguagem LOGO, houve uma nítida melhora sobre os conceitos de informática, além dos alunos demonstrarem estar mais interessados e motivados em sala de aula. Os pesquisadores concluíram, então, que a robótica educacional contribuiu de maneira significativa na consolidação de conceitos relativos à informática, às figuras geométricas e às operações matemáticas.

[Ferruzzi \(2001\)](#) realizou uma pesquisa no Instituto de Educação Infanto Juvenil de Londrina, na qual observou que o uso do LOGO no contexto ensino-aprendizagem auxilia em noções de mensuração, uma vez que quando os alunos trabalham com o deslocamento da tartaruga, compreendem relações mensuráveis como relações entre maior e menor. Ele pode notar a possibilidade de construção de processos de ensino-aprendizagem diferentes do estilo tradicional.

2.2 Software SCRATCH

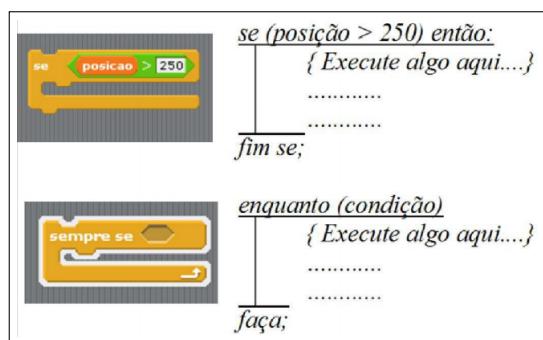
O SCRATCH é gratuito e foi projetado pelo grupo Lifelong Kindegarten no Media Lab do MIT ([SCRATCH, 2020](#)). Ele é ideal para iniciantes da área de programação, já que para utilizá-lo não é necessário nenhum conhecimento prévio de qualquer linguagem de programação. Seu público alvo consiste em pessoas com mais de 8 (oito) anos de idade e objetiva auxiliar no aprendizado de conceitos matemáticos e computacionais. Essa linguagem permite que histórias animadas, jogos e outros programas interativos sejam criados. Além do SCRATCH permitir criar suas próprias histórias interativas, jogos e animações, ele permite o compartilhamento dessas criações com membros da comunidade online.

Ao utilizar o SCRATCH, os usuários desenvolvem habilidades significativas como trabalhar de forma colaborativa, refletir de maneira sistemática, pensar de forma criativa, fluidez tecnológica, gestão de informação, resolução de problemas, tomada de decisão, entre outros ([SCRATCH, 2020; EDUSCRATCH, 2011](#)). Sua programação é feita através da manipulação de blocos de comandos, além disso o *software* é considerado por especialistas um programa inovador, pois fornece condições ao educando de ser projetista-criador da aprendizagem e do conhecimento no momento em que disponibiliza ferramentas de programação simplificadas ([FRANÇA; AMARAL, 2013; MICHELON, 2012](#)).

[Neto \(2013\)](#) acredita que por não utilizar linhas de código, como é utilizado em linguagens como Java¹ e Python², por exemplo, é possível realizar a criação de programas de maneira simplificada e dinâmica, além de permitir visualização gráfica da execução do programa criado.

Um programa do SCRATCH, mesmo não possuindo uma estrutura tradicional, apresenta certa similaridade. Isso pode ser notado na forma como os blocos são identados. Na Figura 1 há blocos de "se... então" e "sempre se..." do SCRATCH juntamente com sua versão em algoritmo ("português estruturado") para que as semelhanças e diferenças sejam notáveis.

Figura 1 – Blocos do SCRATCH e fragmentos de algoritmo.



Fonte: [Neto \(2013\)](#).

¹ Disponível em: <<http://www.oracle.com/technetwork/java/>>. Acesso em: 23 nov. 2020.

² Disponível em: <<https://www.python.org/>>. Acesso em: 23 nov. 2020.

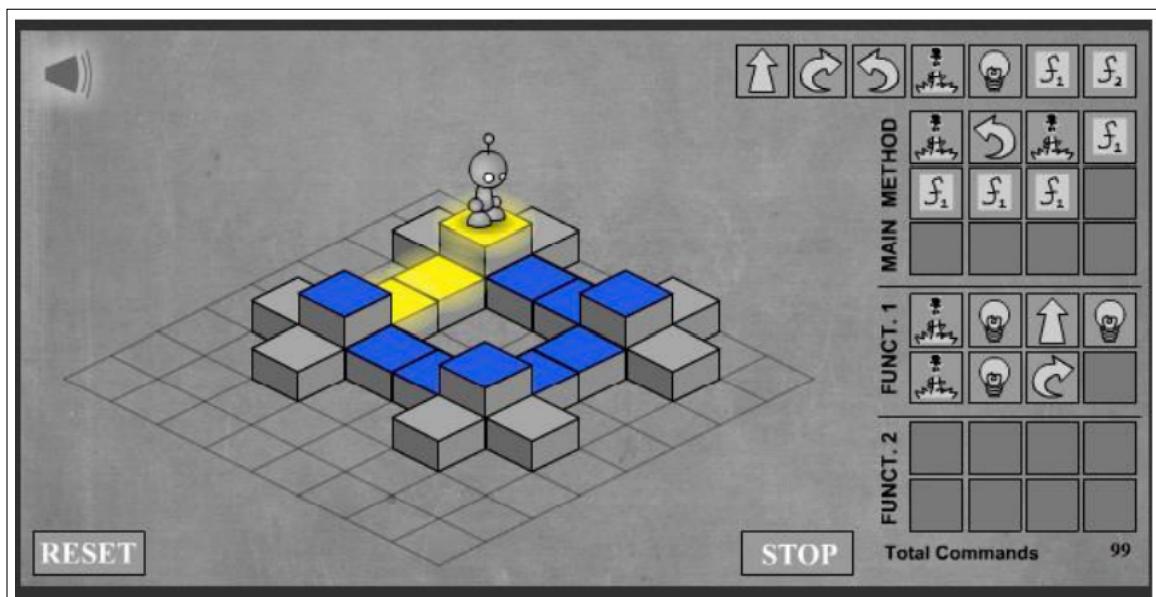
2.3 Software LIGHTBOT

O software LIGHTBOT foi criado em 2008 por Daniel Yaroslavski, estudante de graduação da Universidade de Waterloo no Canadá. Posteriormente, reconstruiu-se uma nova versão para iOS³ e Android⁴ que é uma maneira fácil de auxiliar as crianças a aprenderem conceitos como *loops* (comandos de repetição). O software começou como uma espécie de jogo de Q-Bert⁵, o qual permite que um pequeno robô seja conduzido para cima e para baixo em uma série de blocos. Mais tarde, Yaroslavski acrescentou aspectos mais educacionais, incluindo a necessidade de usar conceitos de programação para resolver quebra-cabeças (BIGGS, 2013).

O objetivo deste jogo é mover um robô por um tabuleiro e "ligar" certas casas deste tabuleiro. Ao ligar todas as casas marcadas o usuário avança de nível. Para isso o jogador deve criar um *script* (arquivo que serve como roteiro, contendo as instruções criadas no software) utilizando uma quantidade restrita de ações, de um conjunto de comandos disponíveis. Pode-se usar uma sequência de comandos principal e duas sequências adicionais. As sequências adicionais correspondem à sequências de comandos que podem ser chamadas na sequência principal. Também é possível a utilização de condicionais, recursividade e outros conceitos de programação.

Na Figura 2 é possível conhecer a interface gráfica com o usuário do jogo LIGHTBOT.

Figura 2 – Jogo LIGHTBOT.



Fonte: Santos (2010).

Enquanto o LOGO e o SCRATCH são linguagens de programação e não possuem um objetivo determinado, o LIGHTBOT é um jogo e possui um objetivo que deve ser concluído

³ Disponível em: <<https://www.apple.com/br/ios/ios-14/>>. Acesso em: 25 nov. 2020.

⁴ Disponível em: <https://www.android.com/intl/pt-BR_br/>. Acesso em: 25 nov. 2020.

⁵ Disponível em: <<https://bit.ly/33uOlki>>. Acesso em: 30 nov. 2020.

para que o usuário consiga passar de nível. Em relação à forma de se programar, o LIGHTBOT é programado por meio de botões de ação e há uma limitação no uso destes botões que variam de acordo com a fase a ser jogada, já o LOGO é programado utilizando uma linguagem escrita e simplificada e o SCRATCH utilizando blocos de comandos.

2.4 LEGO MINDSTORMS

Os kits MINDSTORMS fazem parte de uma linha do brinquedo LEGO®, lançada comercialmente em 1998 e voltada para a educação tecnológica. Inicialmente, assim como foi mencionado no capítulo [1 Introdução](#), o projeto foi inspirado por Seymour Papert e desenvolvido em uma parceria entre o Media Lab do Massachusetts Institute of Technology (MIT) e o Lego Group ([FRIEDRICH et al., 2012](#)).

Segundo [Friedrich et al. \(2012, p.03\)](#):

O sistema LEGO MINDSTORMS é formado por quatro tipos de sensores, três motores e um controlador central, além de um conjunto de peças da linha tradicional (tijolos cheios, placas, rodas) e da linha Lego Technic (tijolos vazados, motores, eixos, engrenagens, polias e correntes). Cada componente tem suas funcionalidades específicas: os motores são os responsáveis por movimentar a estrutura das montagens; os sensores são os responsáveis pela coleta de informação junto ao meio externo; o controlador central, também conhecido por módulo RCX (Robotic Command Explorer) é responsável pela parte inteligente, contendo o software que gerencia o sistema.

Além desses sensores, o conjunto LEGO MINDSTORMS inclui peças de montar, permitindo criar humanoides, veículos, guindastes, animais, entre muitas outras coisas. ([Figuras 3 e 4](#)).

Os conjuntos do LEGO MINDSTORMS são utilizados, além da função lúdica, para o desenvolvimento de projetos de pequeno e médio porte, estimulando a criatividade e a solução de problemas do cotidiano por parte dos alunos ([FRIEDRICH et al., 2012](#)).

Uma grande vantagem da utilização dos kits LEGO MINDSTORMS é sua compatibilidade entre versões. Ou seja, os kits mais novos possuem a mesma base dos kits antigos e são compatíveis entre si, possibilitando o uso da plataforma de programação um do outro e até mesmo o uso dos sensores mais novos em controladores mais antigos. Nos novos lançamentos, o grupo LEGO MINDSTORMS aprimora a qualidade dos sensores e atualiza o design das peças, as modernizando, porém mantendo o mesmo formato e tamanho.

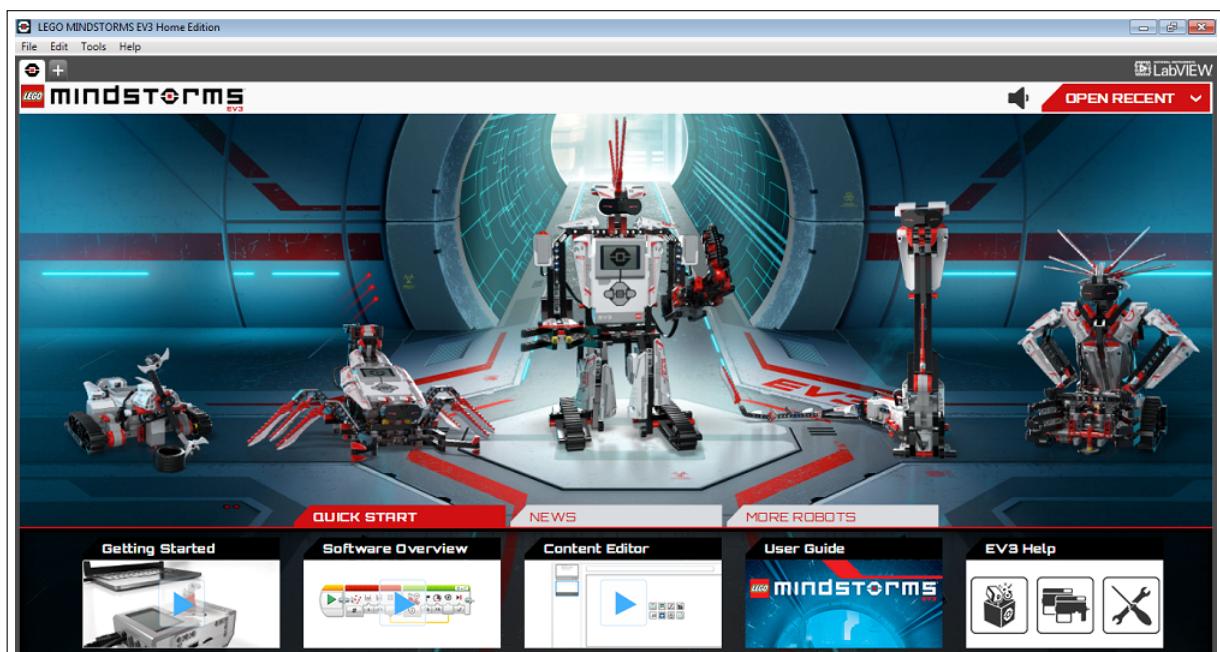
A Figura [5](#) mostra o software EV3 que faz parte do Kit LEGO MINDSTORMS, denominado LEGO MINDSTORMS EV3 *Home Edition*. Essa é uma plataforma descomplicada e acessível para a utilização. A programação é feita através de blocos pré-programados e não com linhas de código. Cada bloco possui uma funcionalidade e algumas configurações próprias, esses blocos podem ser de algumas categorias diferentes, sendo elas: blocos de ação,

Figura 3 – As peças de um kit LEGO MINDSTORMS EV3.



Fonte: Hutchinson (2013).

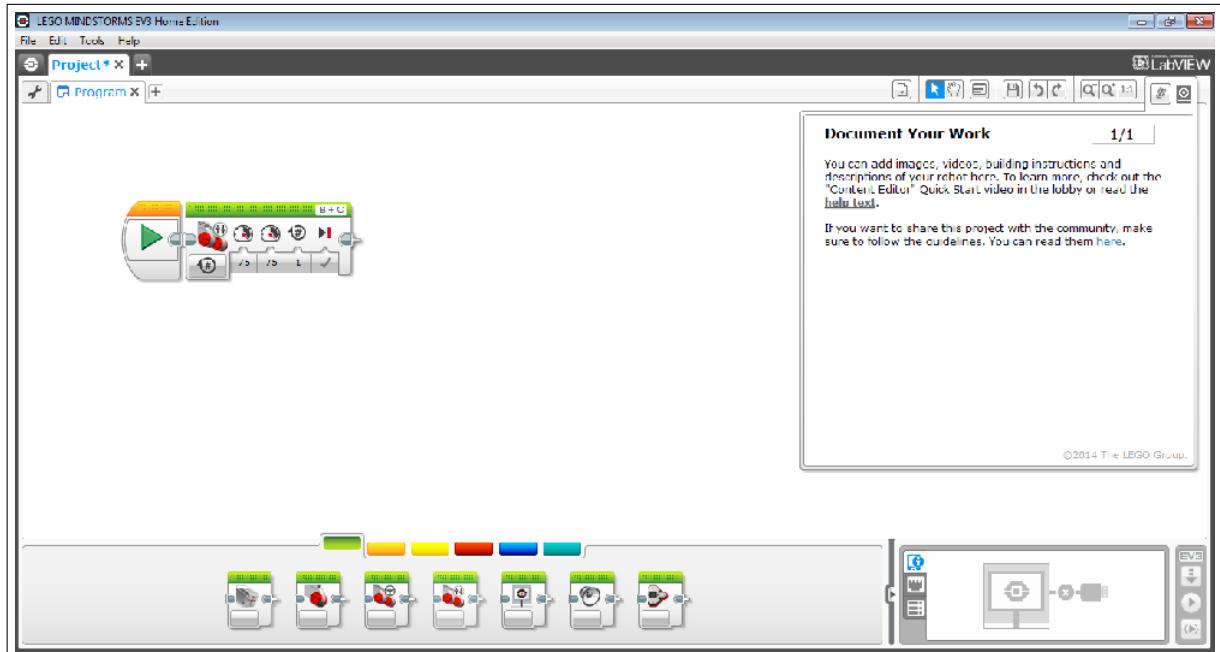
Figura 4 – Página inicial do LEGO MINDSTORMS EV3 *Home Edition*, onde exibe algumas possibilidades de robôs projetáveis do Sistema LEGO.



Fonte: elaborado pelo autor.

blocos de fluxo, blocos de sensor, blocos de dados, blocos avançados e blocos mais utilizados (SHIMABUKURO; ZAMBON, 2016).

Figura 5 – Página de programação do LEGO MINDSTORMS EV3 Home Edition.



Fonte: elaborado pelo autor.

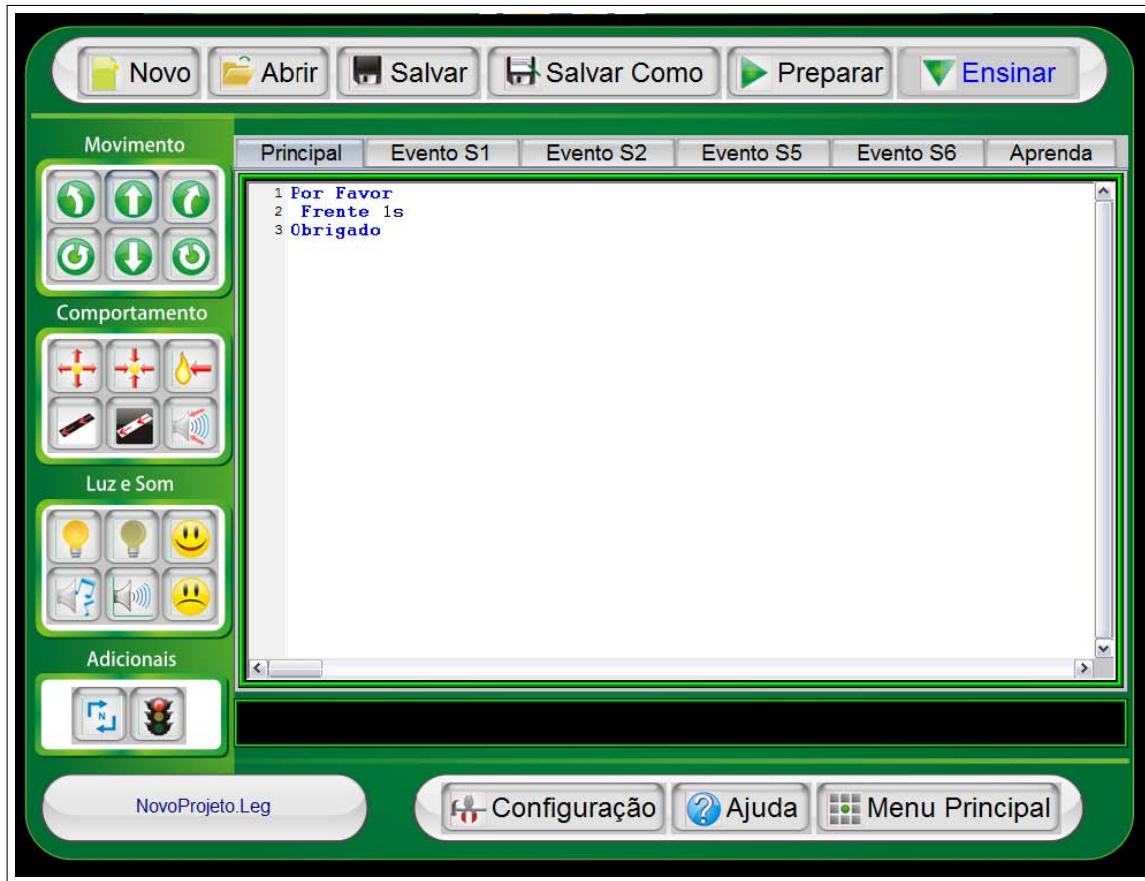
A programação é feita através do encaixe dos blocos de comando, assim como acontece no SCRATCH, visto em [2.2 Software SCRATCH](#). Para criar um novo programa é necessário apenas arrastar para o centro o bloco desejado, conectando-o ao bloco de *Start*, o qual já está presente ao iniciar um novo projeto no *software*, no caso de ser o primeiro comando do programa, ou no último bloco de comando presente na área de programação.

2.5 LEGAL

O ambiente de desenvolvimento LEGAL foi desenvolvido pela empresa PETE e faz parte de um kit de desenvolvimento conhecido como KIT ALFA, o qual se trata de um sistema modular para projetos de robôs, dispositivos eletrônicos, mecânicos e computacionais.

O LEGAL é uma interface gráfica com o usuário e uma linguagem de programação que possui sua sintaxe em português. Assim como dito por [Lisboa \(2015\)](#), "foi criada para estimular o raciocínio lógico, respeitando a capacidade cognitiva de usuários em diferentes faixas etárias". Além dessas vantagens, essa linguagem de programação agrupa conceitos de valor social, devido a utilização de "por favor" e "obrigado" como comandos para iniciar e finalizar seus códigos, respectivamente. O ambiente de programação LEGAL pode ser visto na Figura 6.

Figura 6 – Interface de programação do ambiente de desenvolvimento LEGAL.



Fonte: <http://www.unoeste.br/Fipp/robotica/programacao.aspx>.

Este *software* foi pensado para crianças e adolescentes com nenhum conhecimento sobre programação, mas programadores mais experientes também podem utilizá-lo, sendo que a funcionalidade é idêntica, porém o código pode conter conceitos de lógica de programação mais avançados (PETE, 2020).

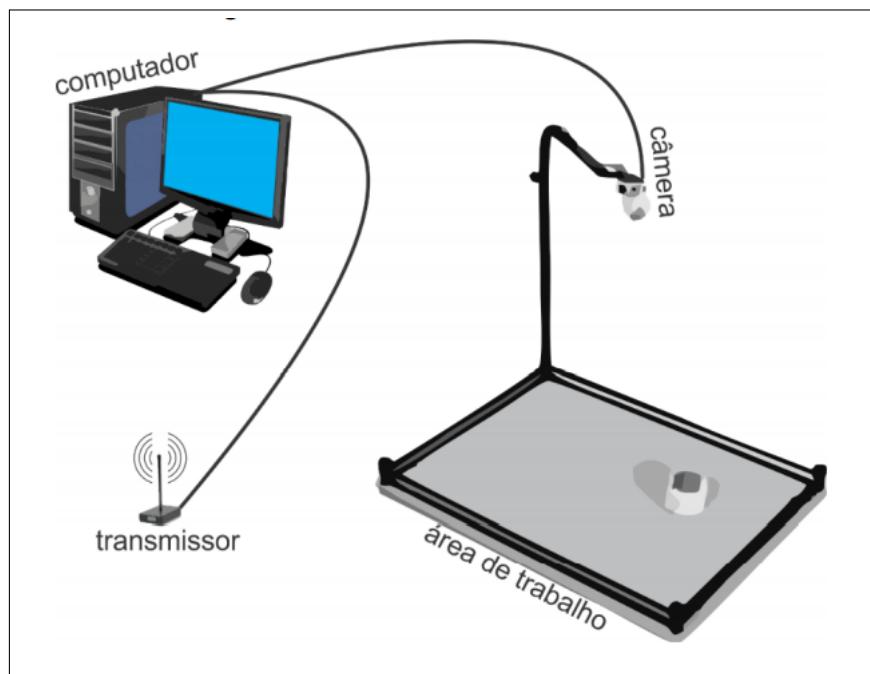
No LEGAL, além de programar, também é possível controlar e coletar dados. O controle é feito através de leituras dos sensores e atuadores, o que permite obter os valores que os sensores estão captando e quais são os estados dos atuadores. Já para a coleta de dados, é possível programar o módulo de controle para que essa coleta ocorra de forma autônoma ou para realizar coleta de dados em tempo real (PETE, 2020).

2.6 AEDROMO

O AEDROMO (Ambiente Experimental e Didático com Robôs Móveis), disponível no LISDI (Laboratório de Integração de Sistemas e Dispositivos Inteligentes) do Departamento de Computação da UNESP de Bauru, é um ambiente versátil, amigável e escalável que suporta uma ampla gama de experimentos. Este ambiente é composto basicamente por pequenos robôs

que interagem entre si e com outros objetos sobre uma área de trabalho horizontal e plana para realizar tarefas específicas. Além dessa área de trabalho, para sua formação há também objetos passivos, um ou mais robôs, dois computadores, uma câmera global do tipo *webcam* e um transmissor (Figura 7).

Figura 7 – Ambiente do AEDROMO.



Fonte: [Lisboa \(2015\)](#).

De acordo com [Filho et al. \(2006\)](#), as premissas na concepção e desenvolvimento deste ambiente são a flexibilidade de adaptação, uso por várias disciplinas e o baixo custo. Os experimentos, neste ambiente, podem ser motivados para fins de pesquisa, aprendizagem ou, entretenimento.

Destaca-se ainda que o AEDROMO pode ser programado em qualquer linguagem de programação que suporte *sockets*, para que dessa forma seja possível a conexão entre o servidor (AEDROMO) e um ou mais clientes.

A *webcam* presente na arquitetura do AEDROMO é responsável por captar todo o ambiente quanto os robôs e objetos nele presente. Com ela pode ser dispensado o uso de sensores mais sofisticados. Concomitantemente, ela atribui ao ambiente maior amplitude de possibilidades pedagógicas para ensino e pesquisa ([SOUSA, 2015](#)).

As imagens capturadas pela câmera são enviadas ao computador servidor, onde as mesmas são adquiridas e processadas. O processamento determina as posições cartesianas (x , y) e o ângulo dos objetos (robôs, bolas, cubos, entre outros) inseridos na área de trabalho ([ALVES et al., 2011](#)).

Além do servidor responsável pelo controle dos robôs, o AEDROMO também conta

com um servidor de simulação, cuja finalidade é melhorar sua acessibilidade por não necessitar do hardware físico, sendo uma forma de ensino de custo reduzido, o que permite a utilização do AEDROMO por mais pessoas.

O simulador foi desenvolvido em Processing⁶, que além de ser uma linguagem de programação, é um ambiente de desenvolvimento integrado de código aberto, considerado um *software* livre, capaz de gerar arquivos executáveis para os sistemas operacionais mais utilizados atualmente. A linguagem de programação Java é a base do Processing.

Na arquitetura do AEDROMO há dois tipos de computadores: o Computador do Servidor e o Computador do Aluno funcionando como cliente do ambiente. O Computador do Servidor é responsável pela implementação do sistema de visão computacional e pela comunicação com os robôs. Por outro lado, o Computador do Aluno é onde o aplicativo de controle (Software do Aluno) é desenvolvido e onde se insere o LOGIKID. Este aplicativo é um cliente que recebe do servidor as posições cartesianas de todos os objetos presentes na área de trabalho e envia ao servidor os comandos de acionamento de cada robô, para que o servidor encaminhe estes comandos ao robô. A comunicação entre o cliente e o servidor é dada através de uma rede Ethernet utilizando o protocolo UDP/IP (User Datagram Protocol over Internet Protocol), permitindo que seja escrito um aplicativo de controle para cada robô em qualquer linguagem de programação que suporte *socket's*. Adicionalmente, os aplicativos de controle podem ser executados no mesmo computador ou em computadores remotos.

Sabendo disso, a linguagem de programação utilizada foi Java, a qual suporta utilização de *socket's* e é orientada à objetos. O Java foi criado na década de 90 e utiliza a ideia de máquina virtual, camada entre o sistema operacional e a aplicação, possibilitando a independência de sistema operacional. Ou seja, o Java permite que a aplicação funcione da mesma forma em todos os sistemas operacionais em que for executado, característica relevante para permitir a execução do AEDROMO em qualquer máquina e/ou sistema operacional.

Como foi citado anteriormente, o LOGIKID é um programa cliente que se conecta com o AEDROMO. Para a utilização do software, foi desenvolvida uma linguagem interpretada, sendo que seu interpretador e sua estruturação foram desenvolvidos em Java. A interface gráfica com o usuário foi desenvolvida em Processing, uma linguagem de programação de código aberto e ambiente de desenvolvimento integrado.

⁶ Disponível em: <<https://processing.org/>>. Acesso em: 01 dez. 2020.

3 Desenvolvimento do Projeto

O projeto proposto pôde ser descrito por quatro etapas principais, sendo que, na primeira etapa, foi feito um levantamento literário sobre a área do tema.

Este projeto conta com o ambiente real e o ambiente simulado do AEDROMO juntamente com o antigo *software* LOGIKID desenvolvido por [Lisboa \(2015\)](#), os quais foram estudados cautelosamente na segunda etapa. O LOGIKID recebe do AEDROMO a posição do robô e dos obstáculos no ambiente. Dessa forma, o software é capaz de calcular as distâncias entre eles e indicar ao robô qualquer alteração necessária em sua rota para evitar colisões com os obstáculos.

Na terceira etapa foram feitas implementações na linguagem de programação do LOGIKID para que fossem aceitas as funcionalidades propostas no *software*. Parte dessa linguagem estava previamente estruturada, entretanto, no decorrer do desenvolvimento deste projeto, melhorias foram feitas para que o LOGIKID pudesse contribuir ainda mais para o estudo de lógica de programação.

E, por fim, na última etapa a interface gráfica com o usuário foi reestruturada a fim de permitir o uso pelo usuário das modificações implementadas no decorrer da terceira etapa.

3.1 Materiais

Para o desenvolvimento deste projeto, foram utilizados o AEDROMO, o qual está disponível no LISDI (Laboratório de Integração de Sistemas e Dispositivos Inteligentes) do Departamento de Computação, Faculdade de Ciências da UNESP de Bauru. Além dele, foi utilizado o seu simulador, desenvolvido em Processing.

A interface com o usuário foi desenvolvida em linguagem de programação Processing a partir do *software* do LOGIKID já existente antes do início deste projeto.

O interpretador da linguagem do LOGIKID foi desenvolvida no ambiente de desenvolvimento integrado (IDE) Eclipse¹, também em Java.

3.2 Linguagem do *software* LOGIKID

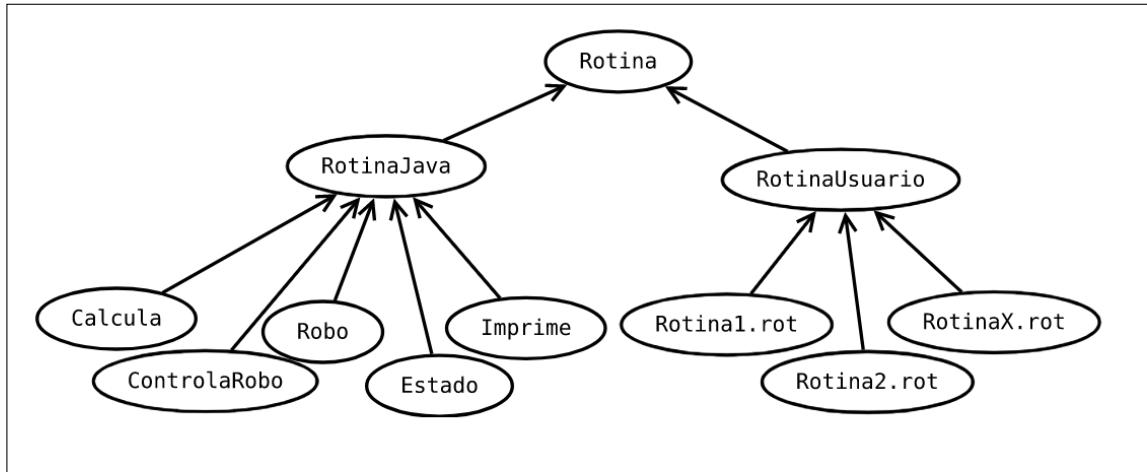
A linguagem do *software* LOGIKID é uma peça fundamental no projeto, pois é a forma de conexão do usuário com o robô físico. Ela é dividida em rotinas responsáveis por todas as

¹ Disponível em: <<https://www.eclipse.org/downloads/>> . Acesso em: 02 dez. 2020.

ações e coleta de informações dos robôs. Essas rotinas possuem todas as suas instruções em texto definidas em português, o que facilita seu entendimento.

As classes *RotinaJava* e *RotinaUsuario* são descendentes da classe *Rotina*, a qual é a base da linguagem desenvolvida. A estrutura descrita da linguagem pode ser observada na Figura 8.

Figura 8 – Estrutura da linguagem do software LOGIKID.



Fonte: elaborado pelo autor.

As funções descendentes da classe *RotinaJava* são essenciais para a manipulação do robô do AEDROMO. Isso ocorre, pois elas permitem a definição de rotinas chamáveis pela linguagem do LOGIKID através do Java, possibilitando que códigos em Java possam interagir com o AEDROMO, algo que seria impossível apenas com as rotinas declaradas em texto pelo usuário.

A função *ControlaRobo*, por exemplo, determina qual é o robô que está sendo controlado pelo cliente, sabendo que cada cliente pode locomover apenas um robô e o servidor pode conter um ou mais robôs. Já a função *Estado*, que pode ser observada na segunda linha do código presente na Figura 12, permite que o usuário saiba as coordenadas do seu robô, assim como sua angulação (para onde está orientado). A partir dessas funções é possível saber a posição do robô que está sendo controlado e programá-lo para executar uma tarefa específica, como ir até um objeto do campo e carregá-lo até uma coordenada determinada.

Cada função da linguagem do LOGIKID pode ser escrita com uma classe Java descendente da classe *RotinaJava* ou armazenada em um arquivo separado, interpretada pela classe *RotinaUsuario*, neste arquivo se define as operações de programação.

Estas funções são chamadas pelo nome da classe Java ou do arquivo seguido pelos parâmetros que serão enviados, podendo ser números literais, variáveis ou expressões matemáticas.

Normalmente é o usuário que define a rotina principal (esta através do ambiente

LOGIKID) e outras rotinas auxiliares (utilizando um editor de texto). O programa do usuário trata-se de uma rotina que chama outras rotinas pré-definidas ou definidas pelo próprio usuário, sendo que cada rotina é chamada como um novo comando. Além disso, sabendo que o usuário pode definir novas rotinas e comandos, ele possui a possibilidade de estender e ampliar a linguagem.

Comparando duas funções para somar dois valores em Java para a integração ao LOGIKID e a mesma rotina na linguagem do LOGIKID, temos as funções *Soma* para o LOGIKID definidas na linguagem Java (Figura 9) e na linguagem do *software* (Figura 10). Em ambos os casos são recebidos três parâmetros para a soma, é realizada a soma entre os dois últimos parâmetros e o resultado é armazenado no primeiro. A função mencionada na linguagem Java só é utilizada para que sejam feitos acessos a recursos não disponíveis na linguagem interpretada. Na Figura 11 há uma função responsável por chamar a função *Soma* da Figura 10.

Figura 9 – Função *Soma* em Java.

```
class Soma extends RotinaJava {

    void executa(String linChamada[], Map<String, Object> locais) {
        super.executa(linChamada, locais);
        Float f;
        try {
            f = expressao.valor(linChamada[2]) + expressao.valor(linChamada[3]);
            String dest = variavelDestino(linChamada[1]);
            locais.put(dest, f);
        } catch (Exception e) {
            erro(e.getMessage() + "\n Argumento invalido.");
        }
    }
}
```

Fonte: elaborado pelo autor.

Figura 10 – Função *Soma* na linguagem do LOGIKID.

- | |
|---|
| 1. Soma & x y z
2. Calcula x y+z
3. fim |
|---|

Fonte: elaborado pelo autor.

Figura 11 – Função *testeSoma* responsável por chamar a função *Soma* na linguagem do LOGIKID.

```

1. testeSoma x y
2.     Soma x y 10
3.     fim

```

Fonte: elaborado pelo autor.

Durante a programação de um arquivo ".rot", o usuário escreve rotinas em "português estruturado" realizando chamadas de funções tanto oferecidas pela linguagem, quanto desenvolvidas anteriormente pelo próprio usuário, localizadas em outros arquivos ".rot".

A Figura 12 apresenta o conteúdo do arquivo *deslocaRobo.rot* que contém um exemplo básico de declaração de uma função, onde é definida a função *deslocaRobo* e seus parâmetros, a qual chama a função *Estado*, que retorna a posição do robô 1 no AEDROMO; caso o *x* do robô 1 seja maior do que 200 milímetros, chama a função *irPara* com as coordenadas recebidas nos seus parâmetros.

Figura 12 – Exemplo da definição de uma função chamada *deslocaRobo* definida na linguagem do LOGIKID que chama as funções *Estado* e *irPara*.

```

1. deslocaRobo x y
2.     Estado ROBO1 xr yr ar
3.     se xr>200
4.         irPara x y
5.     fim
6. fim

```

Fonte: elaborado pelo autor.

O recebimento e envio de parâmetros, visto nas funções *deslocaRobo* e *irPara* da Figura 12, são parte da nova particularidade da linguagem do software LOGIKID. A partir disso, há a possibilidade de programar funções com recursividade (com passagem de parâmetro por referência). Um exemplo de função com recursividade pode ser visto na Figura 13, com uma rotina recursiva do cálculo de Fibonacci.

Nota-se na primeira linha do código apresentado na Figura 13 o nome de definição da função seguida pelos parâmetros; o & comercial “&” antes do parâmetro “r” indica que esta variável é passada por referência.

Outra particularidade implementada à linguagem possibilitou a manipulação de vetores e matrizes, bem como a chamada de funções matemáticas com dois parâmetros. Outra característica nova é a possibilidade da exibição de textos a partir da execução da função *Imprime*. Essas implementações foram possíveis através da atualização do interpretador da linguagem, o qual analisa os componentes presentes no comando e determina o tratamento necessário.

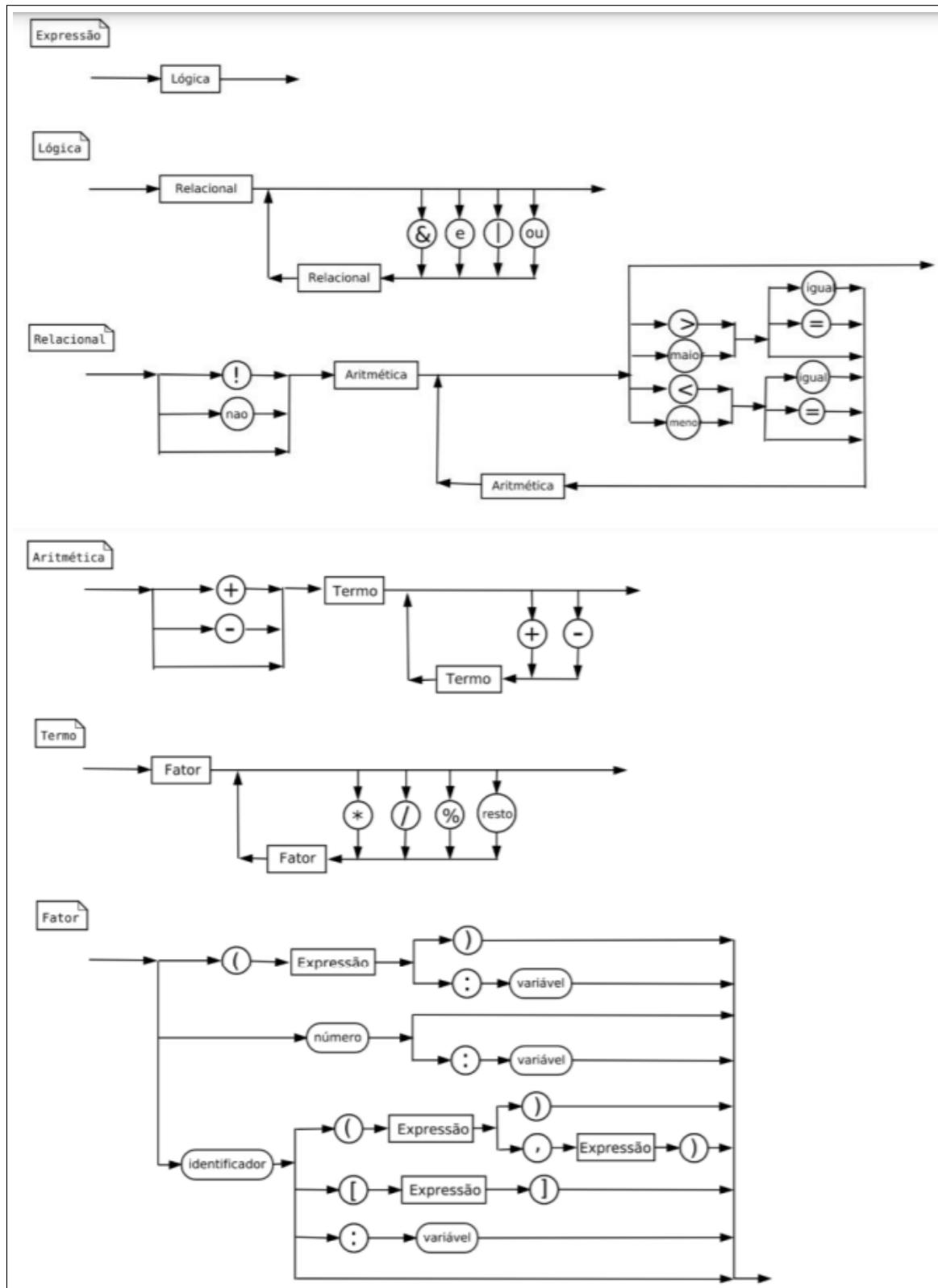
Figura 13 – Função com recursividade.

```
1. fibonacci & r n
2.     se n = 0
3.         Calcula r 0
4.     senao
5.         se n = 1
6.             Calcula r 1
7.         senao
8.             Calcula r1 0
9.             Calcula r2 0
10.            fibonacci r1 n-1
11.            fibonacci r2 n-2
12.            Calcula r r1+r2
13.            Imprime n
14.        fim
15.    fim
16. fim
```

Fonte: elaborado pelo autor.

A Figura 14 mostra um diagrama do interpretador da linguagem do LOGIKID que permite a utilização de funções matemáticas e a manipulação de vetores e matrizes, por exemplo. Quando uma expressão é identificada, essa sequência de análise (Figura 14) começa a ser feita pelo interpretador.

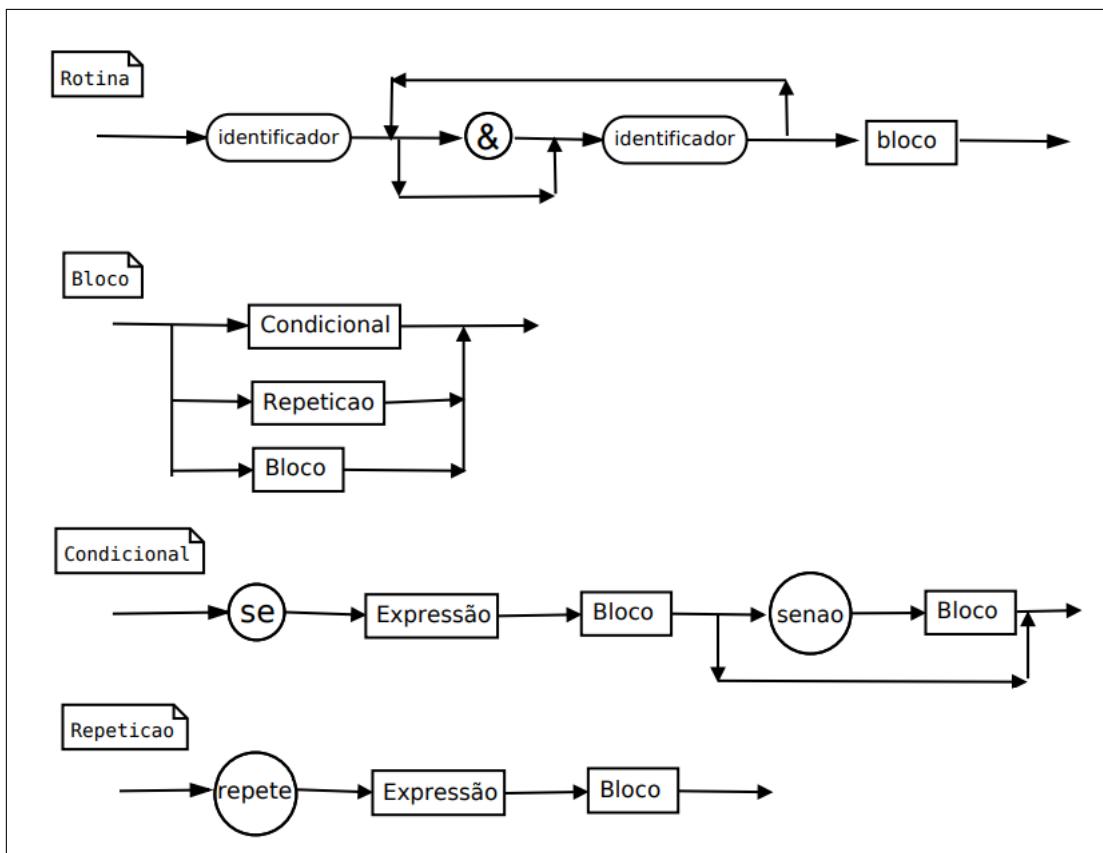
Figura 14 – Diagrama do interpretador da linguagem do LOGIKID.



Fonte: elaborado pelo autor.

Já na Figura 15 há um diagrama do interpretador da linguagem do LOGIKID representando o funcionamento do comando condicional "se" e o comando de repetição "enquanto" indicado pela palavra "repete" na linguagem do LOGIKID.

Figura 15 – Diagrama do interpretador da linguagem do LOGIKID - Comando se e *repete*.

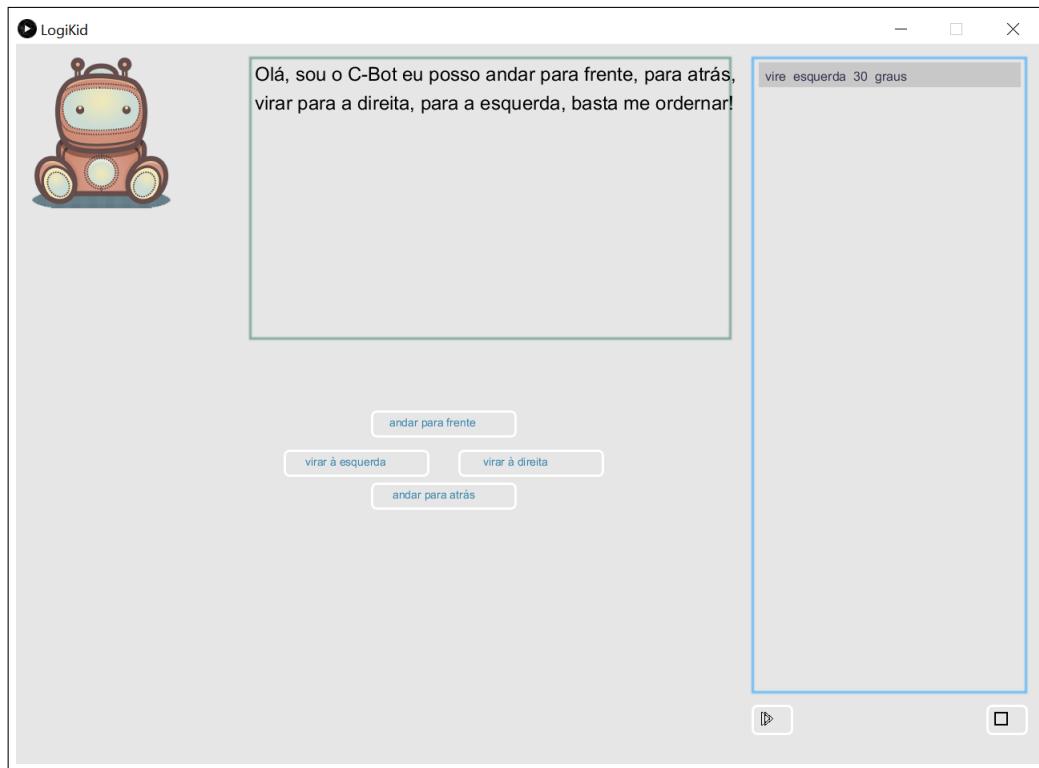


Fonte: elaborado pelo autor.

3.3 Interface com o usuário

O objetivo principal do LOGIKID é realizar movimentos por um robô dentro do ambiente do AEDROMO. A Figura 16 mostra a interface original do LOGIKID, desenvolvido por [Lisboa \(2015\)](#). É possível observar que há três áreas principais, a de comunicação do "robô" (mascote) com o usuário, a de seleção de instrução e a de programa, onde as instruções estão sendo adicionadas, construindo, assim, o programa final. Na parte inferior da área de programa há dois botões, sendo o da esquerda para iniciar a execução do código no servidor do AEDROMO e o da direita para pausar tal execução. Já na área de seleção de instrução, há quatro opções de botões: andar para frente, virar à esquerda, virar à direita e andar para trás.

Figura 16 – Interface gráfica com o usuário - LOGIKID.

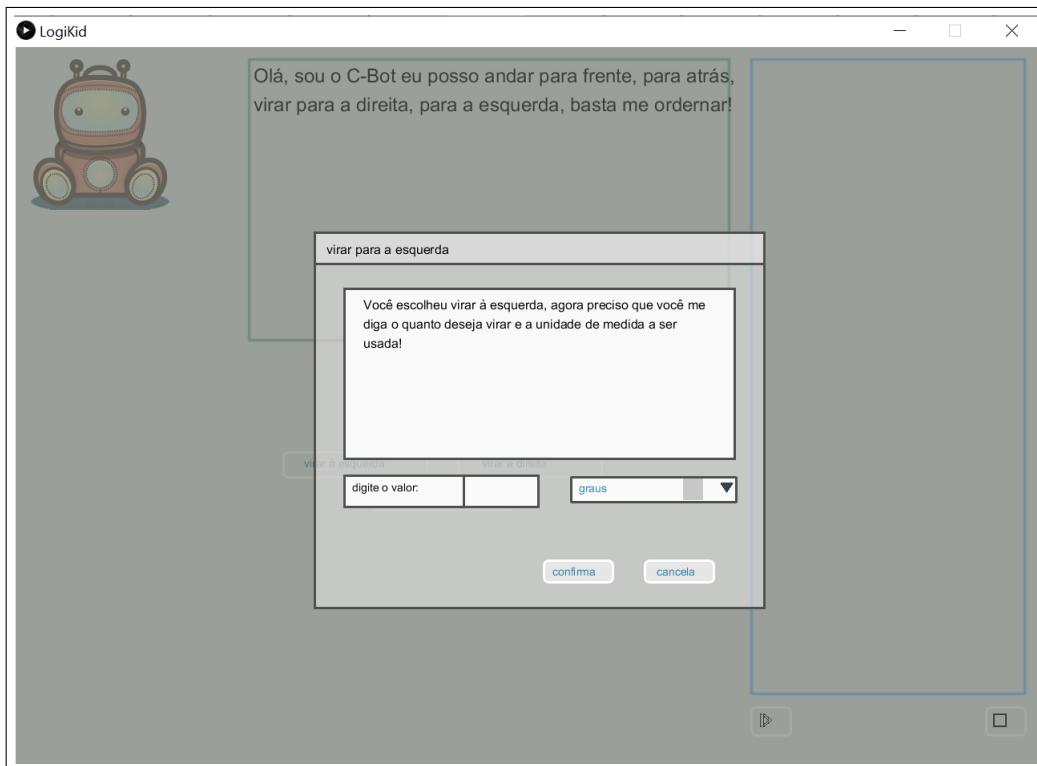


Fonte: elaborado pelo autor.

Para adicionar uma instrução, a criança precisa selecionar um dos quatro botões para indicar qual ação ele irá fazer - andar ou virar - e, então, uma janela aparece para que seja completado os dados daquela instrução (Figura 17).

Na janela há, novamente, um área de comunicação entre o robô e o usuário, onde o robô incentiva a criança à completar as informações da instrução. Há também dois campos para indicar o valor e a unidade de medida da ação a ser feita. Isto é, se a opção desejada foi "virar à esquerda", por exemplo, o usuário deve selecionar se ele deseja virar em graus ou em radianos e digitar o valor a ser virado pelo robô. Por fim, há dois botões, o de confirmar e o de cancelar, definindo assim, se a instrução será adicionada ao programa ou não.

Figura 17 – Janela para completar as informações da instrução a ser adicionada.



Fonte: elaborado pelo autor.

3.3.1 Reestruturação do LOGIKID

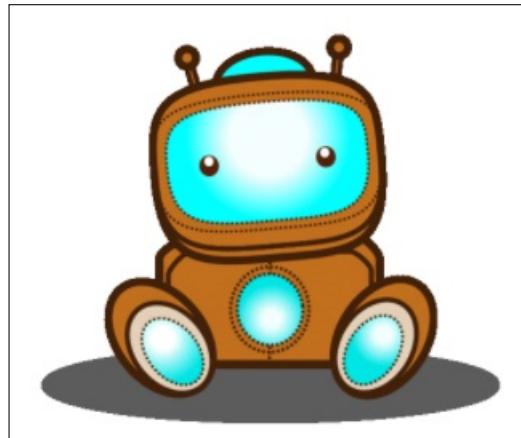
Neste projeto, o *software* foi reestruturado para que novas funcionalidades e experimentos pudessem ser desenvolvidos no LOGIKID, ou seja, para que as novas características da linguagem fossem aproveitadas plenamente.

O *layout* do *software* seguiu o mesmo padrão de interface da versão apresentada na Figura 16, entretanto, sua estrutura interna foi completamente remodelada.

A primeira alteração na interface foi com relação ao mascote. O mascote é a figura de um robô que decora a interface do Logikid, presente no canto superior esquerdo da interface, Figura 16, houve alteração de cores, com o intuito de deixá-lo com cores mais vibrantes e, consequentemente, mais atrativo. Como o mascote antigo era apenas uma imagem, não foi possível alterá-la, tendo que ser desenvolvido uma versão similar para aplicar as modificações desejadas (Figura 18).

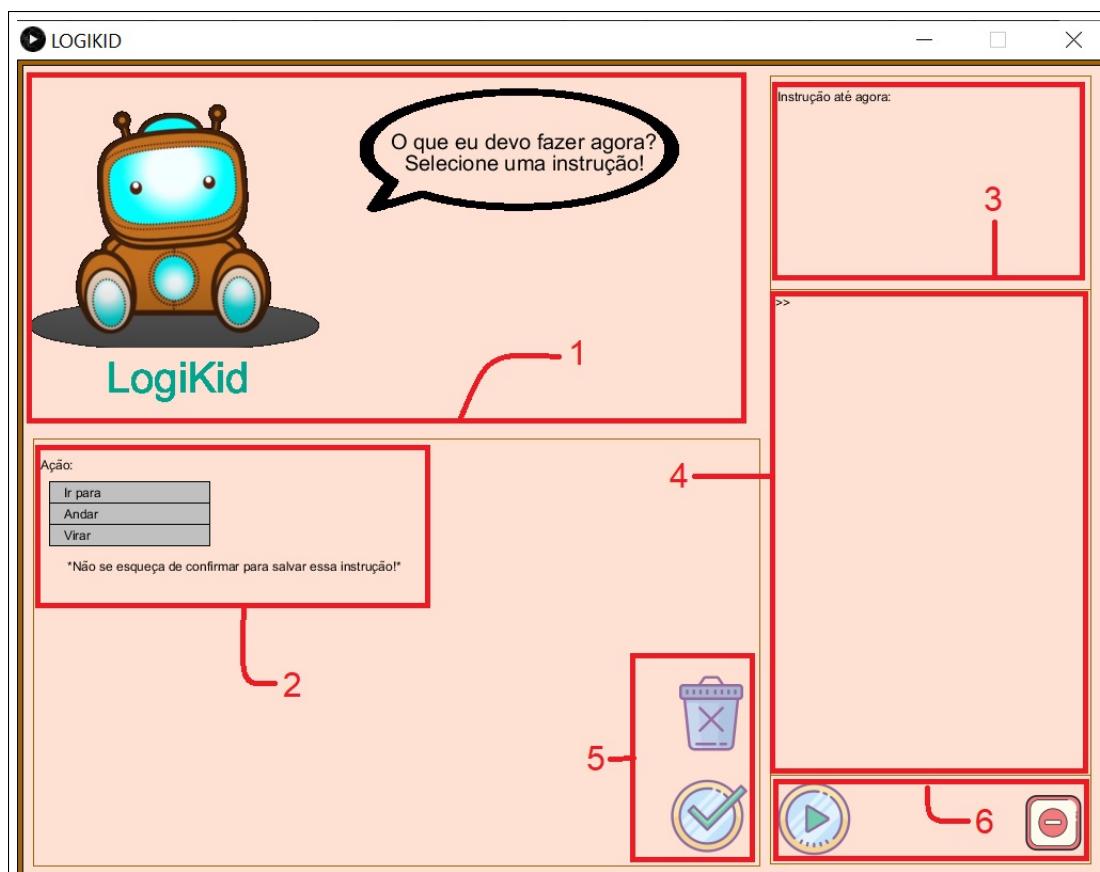
Para realizar a programação de uma tarefa, os movimentos são escolhidos pelos alunos/usuários. Como pode ser visto na Figura 19, há algumas áreas separadas dentro da tela principal do LOGIKID. São elas:

Figura 18 – O robô digitalizado - versão final.



Fonte: elaborado pelo autor.

Figura 19 – Interface do LOGIKID desenvolvida no Processing.



Fonte: elaborado pelo autor.

1. Área de comunicação entre robô/mascote e usuário: essa comunicação é feita pelo balão de fala, instruindo o aluno à como montar a instrução;
2. Área de criação de instrução: nessa área há trocas de perguntas e respostas para que o aluno vá montando a instrução desejada;
3. Área de instrução: é onde se exibe o comando que está sendo criado até o momento, antes de ser confirmado para a adição ao programa;
4. Área do programa: painel onde os comandos finalizados são organizados e apresentados. Há um símbolo (">>") indicando qual é a linha selecionada;
5. Área de botões da instrução: nela há dois botões - o de limpar e o de confirmar - onde o usuário controla a instrução que está sendo criada no momento;
6. Área dos botões do programa: há os botões de executar - o da esquerda - e o de *stop* - o da direita.

Aprofundando mais nas funcionalidades de cada área apresentada, a de comunicação (área 1) possui um balão de fala dinâmico que apresenta perguntas para o usuário se nortear e decidir qual a próxima ação ele quer adicionar ao programa. A cada momento da criação do comando apenas uma pergunta é apresentada pelo mascote, diminuindo a quantidade de informações na tela para evitar distrações e confusões por parte do usuário.

É na área de criação de instrução (área 2) que os botões são atualizados a cada parte da criação de um comando. Nela são exibidos, também, caixas de entrada de texto para digitar algumas informações cruciais, como quantos centímetros o robô andará para frente ou quantos graus ele virará para o lado esquerdo ou direito. Assim como acontece na área de comunicação (área 1), na de criação de instrução também é apresentado uma questão por vez ao usuário.

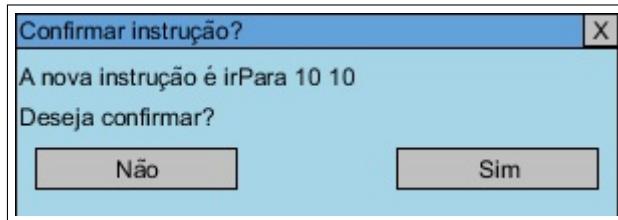
Na área de instrução (área 3), o comando em "português estruturado" é criado. Dessa forma, o usuário consegue acompanhar como ficam em texto as opções selecionadas na área de criação de instrução e ele começa a assimilar e a se familiarizar com o funcionamento da lógica de programação.

O símbolo existente na área do programa (">>") - área 4 - indica a última linha do programa normalmente, entretanto é possível selecionar outras linhas. Tal implementação foi realizada para que, em um trabalho futuro, seja possível adicionar comandos no meio do código, não só ao final dele, além da possibilidade de excluir ou alterar algum comando já confirmado.

Como foi mencionado anteriormente, na área de botões de instrução (área 5) há dois botões. Sendo que ao pressionar o limpar, a última seleção da instrução que está sendo construída é desfeita, ação que permite voltar quantas seleções o usuário quiser. Já ao pressionar o confirmar, uma caixa de diálogo (Figura 20) é exibida para confirmar a ação. A caixa de diálogo é necessária para o usuário confirmar se aquela ação realmente é a ação que ele gostaria

de montar e isso se dá, pois ainda não há opções de editar ou excluir alguma instrução. Em um trabalho futuro, onde essas opções sejam implementadas, a utilização dessa caixa de diálogo não será mais necessária. Após o aceite na caixa de diálogo, então, a instrução é adicionada ao programa e, consequentemente, à área do programa (área 4);

Figura 20 – Caixa de diálogo exibida para confirmar a instrução que será adicionada ao programa.



Fonte: elaborado pelo autor.

Na área dos botões do programa (área 6), os botões existentes são responsáveis por criar um arquivo ".rot" temporário que será executado no AEDROMO ou no seu simulador (o botão de executar) e também por parar a execução do programa (botão de *stop*). Nota-se que o usuário não é capaz de salvar o programa ou continuar algum programa criado nesta versão do *software*.

Por fim, na Figura 21, há a interface do menu inicial do LOGIKID, onde contém uma opção para sair do *software*, uma opção para iniciar a programação e uma opção para obter informações sobre o projeto, respectivamente os botões da esquerda para a direita. Nota-se também a presença do mascote e um balão de fala para a comunicação com o usuário.

Figura 21 – Tela do menu inicial do LOGIKID.



Fonte: elaborado pelo autor.

3.4 Exemplo de utilização

Nesta seção está sendo apresentado um exemplo de criação de uma instrução *irPara 10 10*.

Ao inicializar o *software*, o usuário se depara com a tela de menu inicial, Figura 21, e deve clicar no botão central para iniciar e, então, a tela exibida é a presente na Figura 22.

Figura 22 – Tela exibida após o menu inicial.



Fonte: elaborado pelo autor.

Para cumprir o exemplo proposto, o próximo passo é selecionar o botão "Ir Para" (primeiro botão da área de criação de instrução - área 2). Após isso, a tela do software fica igual à apresentada na Figura 23 e o botão de "Ponto (x, y)" é o próximo a ser selecionado.

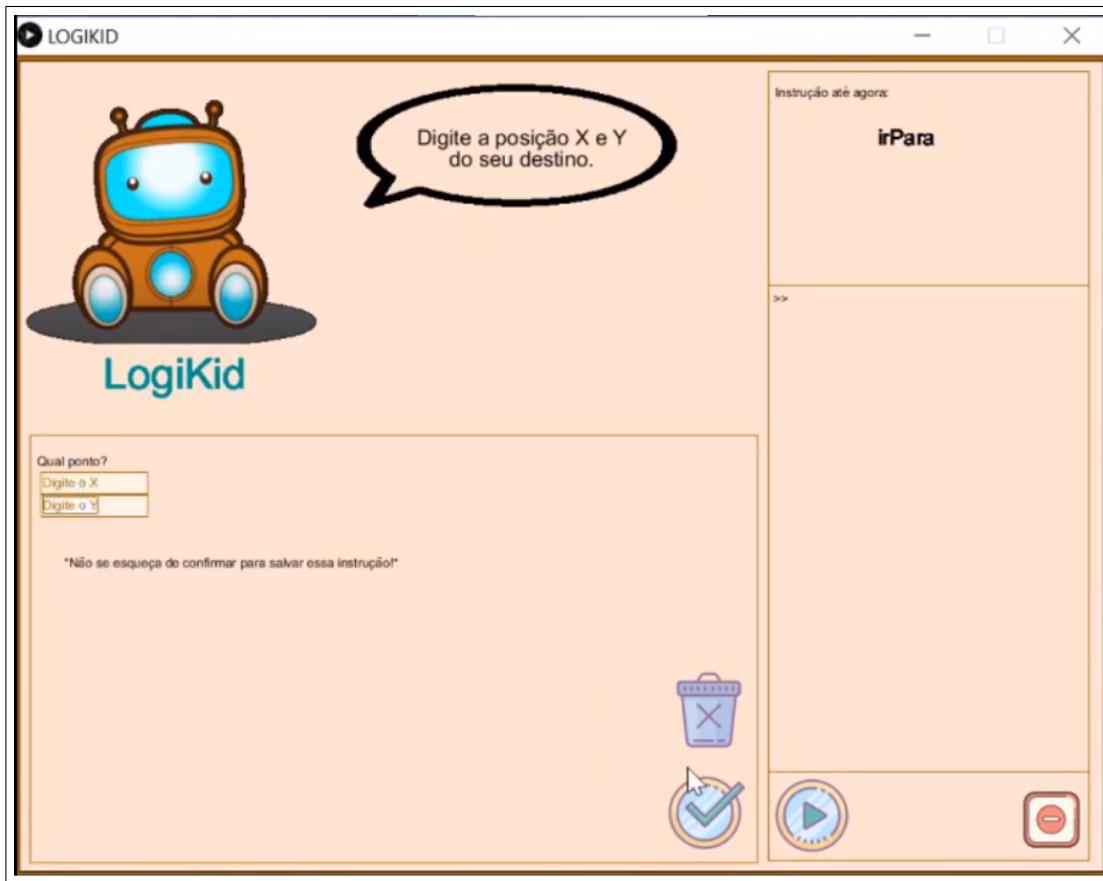
Figura 23 – Tela exibida após a seleção da opção "Ir Para".



Fonte: elaborado pelo autor.

A página seguinte, mostrada na Figura 24, pede então para que o usuário digite o valor de X e o valor de Y, que no caso deste exemplo é 10 (dez) para ambos os campos.

Figura 24 – Tela exibida após a seleção da opção "Ponto (x, y)".



Fonte: elaborado pelo autor.

Após inserir o valor desejado para X e para Y, o usuário deve clicar no botão de confirmação da área de botões da instrução (área 5), ação que exibe uma caixa de diálogo, como a da Figura 20. E a tela exibida é a da Figura 25.

Figura 25 – Tela exibida após a inserção dos valores de X e Y.



Fonte: elaborado pelo autor.

Após a confirmação da nova instrução, ela é inserida na área do programa (área 4), e a tela do software é a apresentada na Figura 26. Ao selecionar o botão de executar localizado na área dos botões do programa (área 6), o usuário consegue visualizar o seu código em execução no AEDROMO ou no seu simulador. Uma imagem do simulador do AEDROMO pode ser vista na Figura 27.

Figura 26 – Tela exibida após a confirmação da instrução.



Fonte: elaborado pelo autor.

Figura 27 – Simulador do AEDROMO.



Fonte: elaborado pelo autor.

4 Contribuições

Este trabalho apresenta as seguintes contribuições para a linguagem do LOGIKID:

1. Passagem de parâmetros por referência: sendo necessário, apenas, acrescentar um “&” como primeiro parâmetro no início da função, tendo em vista que o mesmo não é requisitado ao chamar a função;
2. Manuseio de vetores e matrizes: para isto é utilizado uma variável seguida por um índice entre colchetes ([índice], por exemplo), no caso de um vetor, ou dois índices dentro de seus respectivos colchetes ([índice_linha][índice_coluna], por exemplo), no caso de uma matriz;
3. Exibição de textos ao usuário: onde o texto entre aspas é apenas mais um argumento da função Imprime (nota-se que nenhuma forma de concatenação é necessária);
4. Capacidade de chamar funções matemáticas com dois parâmetros: sendo que é necessário informar a função matemática requerida seguido por seus parâmetros separados por vírgula dentro de parênteses (atan2(y-y_inicial, x-x_inicial), por exemplo).

Tais melhorias permitem um comportamento reativo ou híbrido aos robôs, ampliando os experimentos possíveis de serem desenvolvidos no software LOGIKID, inclusive os que envolvem recursividade.

Em relação à interface gráfica com o usuário, este trabalho contribuiu da seguinte forma:

1. Exibição do comando sendo criado: o comando não é visível apenas quando adicionado ao programa, ele pode ser monitorado durante a criação da instrução;
2. Seleção da linha do programa: através do símbolo ">>" é possível acompanhar em qual linha a instrução está sendo inserida;
3. Exibição de uma pergunta: o usuário tem acesso à apenas uma tomada de decisão por vez, o que não gera dúvidas e nem confusão. Somente quando uma decisão é feita, de uma ação andar por exemplo, que outra pergunta surge, seguindo o exemplo: vai poder decidir se quer andar em centímetros ou em pixels e somente após essa decisão que aparecerá a pergunta de quantos centímetros ou pixels o robô andará;
4. Possibilidade de voltar: o usuário pode voltar um ou mais passos, caso tenha mudado de ideia em relação à instrução desejada.

Como forma de manter as boas práticas de programação, todas as partes da interface foram divididas em classes, mantendo a modularização do código e organizando-o para facilitar possíveis atualizações futuras.

As melhorias mencionadas permitem que a interface gráfica com o usuário se comunique de forma concordante com a nova linguagem do LOGIKID, usufruindo de todas as características da mesma. Ademais, facilitou a manipulação da interface tanto para os usuários quanto para os programadores que possivelmente atualizarão o *software* LOGIKID no futuro.

5 Conclusão

O *software* desenvolvido, além de ser baseado em pressupostos de *softwares* utilizados e avaliados positivamente no contexto de ensino-aprendizagem como LOGO, SCRATCH e LIGHTBOT, conta também com o desenvolvimento de uma linguagem de programação integrada a robôs.

As características do LOGIKID são baseadas numa estrutura existente que é o AE-DROMO e a sua interface que gera o código de programação de acordo com a escolha do aluno, resultando em um programa que faz o robô agir de acordo com tais escolhas.

O *software* LOGIKID tem o intuito de possibilitar o desenvolvimento, nos alunos, de um pensamento computacional, introduzindo a lógica de programação sem o conhecimento de uma linguagem de programação.

O diferencial do LOGIKID é o uso de botões para escrever códigos de programação que controlam um robô, além de indicar o equivalente desse código em "português estruturado" para os alunos começarem a se adaptar à um algoritmo e à uma linguagem de programação.

Em trabalhos futuros relacionados ao LOGIKID, há a possibilidade de implementar opções para edição e exclusão dos comandos adicionados ao programa. Uma característica interessante para o projeto é a inserção de mensagens de áudio nos campos que possuem texto, para dinamizar e atrair ainda mais as crianças, passando a sensação de interação direta com o mascote do *software*. Além disso, a criação de linhas de comando separadas para exibir como exemplo ao usuário, como "o que acha de tentar o comando *irPara 10 10?*", dessa forma ele terá alguns exemplos do que fazer quando se sentir perdido ou sem criatividade.

Outra característica que pode ser desenvolvida em um trabalho futuro é a gamificação do LOGIKID, para que os usuários tenham objetivos mais definidos, aumentando seu nível conforme aprende novos conceitos através do *software*. A gamificação pode ser também uma motivação extra para que os usuários continuem aprendendo através da utilização do *software* LOGIKID.

Referências

- ALVES, S. F.; FILHO, H. F.; PEGORARO, R.; CALDEIRA, M. A.; YONEZAWA, W. M.; ROSÁRIO, J. M. Ambiente educacional de robótica direcionado a aplicações em engenharia. *X Simpósio Brasileiro de Automação Inteligente, São João del-Rei*, 2011.
- BENITTI, F. B. V. Exploring the educational potential of robotics in schools: A systematic review. *Computers Education*, v. 58, n. 3, p. 978 – 988, 2012. ISSN 0360-1315. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0360131511002508>>. Acesso em: 02 dez. 2020.
- BIGGS, J. Light-bot teaches computer science with a cute little robot and some symbol-based programming. 2013. TechCrunch. Disponível em: <<https://techcrunch.com/2013/06/26/light-bot-teaches-computer-science-with-a-cute-little-robot-and-some-symbol-based-programming/>>. Acesso em: 01 dez. 2020.
- EDUSCRATCH, M. T. por. *Computação Criativa: uma introdução ao pensamento computacional baseada no conceito de design*, set. 2011. 2011.
- FERRUZZI, E. C. Considerações sobre a linguagem de programação logo. *Seminário apresentado no grupo de estudos de inteligência artificial aplicada à matemática. SantaCatarina: GEIAAM, setembro*, 2001.
- FILHO, H. F.; PEGORARO, R.; CALDEIRA, M.; ROSÁRIO, J. Aedromo-an experimental and didactic environment with mobile robots. In: *Proceedings of The 3rd International Conference on Autonomous Robots and Agents*. [S.I.: s.n.], 2006.
- FRANÇA, R. S. D.; AMARAL, H. J. C. do. Proposta metodológica de ensino e avaliação para o desenvolvimento do pensamento computacional com o uso do scratch. In: *Anais do Workshop de Informática na Escola*. [S.I.: s.n.], 2013. v. 1, n. 1, p. 179.
- FRIEDRICH, R. V.; SANTOS, D. S. dos; KELLER, R. dos S.; PUNTEL, M. D.; BIASOLI, D. Proposta metodológica para a inserção ao ensino de lógica de programação com logo e lego mindstorms. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.I.: s.n.], 2012. v. 23, n. 1.
- HUTCHINSON, L. Review: Lego mindstorms ev3 means giant robots, powerful computers. 2013. Ars Technica. Disponível em: <<https://arstechnica.com/gadgets/2013/08/review-lego-mindstorms-ev3-means-giant-robots-powerful-computers/>>. Acesso em: 30 nov. 2020.
- LISBOA, C. O. *LOGIKID: software para o ensino de lógica e programação com o AEDROMO*. 2015. 54 f. Monografia (Trabalho de Conclusão de Curso) — Universidade Estadual Paulista “Júlio de Mesquita Filho” (UNESP), Bauru, 2015.
- LOGO, P. Por que logo? 2009. Disponível em: <<http://projetologo.webs.com/texto2.html>>. Acesso em: 01 dez. 2020.
- LOPES, D. d. Q. *A exploração de modelos e os níveis de abstração nas construções criativas com robótica educacional*. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, 2008.

- MICHELON, N. O uso do software educacional como suporte de produção e autoria no ensino fundamental. 2012.
- NETO, V. d. S. M. A utilização da ferramenta scratch como auxílio na aprendizagem de lógica de programação. In: *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*. [S.l.: s.n.], 2013. v. 2, n. 1.
- PETE, R. Legal. 2020. PETE Robótica. Disponível em: <<https://pete.com.br/software-legal/>>. Acesso em: 02 dez. 2020.
- SANTOS, F. K. dos. *JAM - Um Jogo de Aprendizagem Multidisciplinar*. 2010. 76 f. — Universidade Estadual do Oeste do Paraná (UNIOSTE), Cascavel, 2010.
- SCRATCH. Acerca do scratch. 2020. Disponível em: <<https://scratch.mit.edu/about>>. Acesso em: 27 nov. 2020.
- SHIMABUKURO, T. S.; ZAMBON, K. L. *Análise e estudo do uso de LEGO MINDSTORMS para o desenvolvimento de competências em Ensino de Programação*. 2016. 21 f. Monografia (Relatório final de Iniciação Científica PIBIC-Jr) — Colégio Técnico Industrial "Prof. Isaac Portal Roldán- Faculdade de Engenharia de Bauru (FEB - UNESP), Bauru, 2016.
- SOUSA, A. M. *Área de trabalho dinâmica para o ambiente experimental e didático com robôs móveis*. 2015. 50 f. Monografia (Trabalho de Conclusão de Curso) — Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP), Bauru, 2015.
- SOUZA, L. F.; REIS, G. L.; PEREIRA, E. B. A robótica educacional como elo de integração entre o ensino fundamental e de graduação pelo uso da linguagem logo. In: *III Workshop de Robótica Educacional*. [S.l.: s.n.], 2012.