UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I
CENTRO DE CIÊNCIAS E TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

FÁBIO DANTAS GUIMARÃES

Projeto de uma interface para uma Plataforma de Programação Visual destinado ao ensino de Robótica Educacional para crianças de 8 a 14 anos idade.

CAMPINA GRANDE
ANO 2019

FÁBIO DANTAS GUIMARÃES

Projeto de uma interface para uma Plataforma de Programação Visual destinado ao ensino de Robótica Educacional para crianças de 8 a 14 anos idade.

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Graduado em Ciência da Computação

Área de concentração: Usabilidade e Fatores Humanos

Orientador: Prof. Dr. Daniel Scherer

CAMPINA GRANDE
ANO 2019

G963p Guimarães, Fábio Dantas.
Projeto de uma interface para uma Plataforma de Programação Visual destinado ao ensino de Robótica Educacional para crianças de 8 a 14 anos idade [manuscrito] / Fabio Dantas Guimaraes. - 2019.
58 p. : il. colorido.

Digitado.

Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia , 2019.

"Orientação : Prof. Dr. Daniel Scherer , Coordenação do Curso de Computação - CCT."

1. Robótica Educacional. 2. Plataforma de Programação Visual. 3. Pensamento computacional. 4. Usabilidade. I. Título

21. ed. CDD 372.358

Elaborada por Giulianne M. Pereira - CRB - 15/714 BC/UEPB

FÁBIO DANTAS GUIMARÃES

Projeto de uma interface para uma Plataforma de Programação Visual destinado ao ensino de Robótica Educacional para crianças de 8 a 10 anos idade

Trabalho de Conclusão de Curso de Graduação em Ciência da Computação da Universidade Estadual da Paraíba, como requisito à obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 26 de Julho de 2019.

Prof. Dr. Daniel Scherer (UEPB)
Orientador(a)

Profa. MSC Luciana de Queiroz Leal Gomes (UEPB)
Examinador(a)

Prof. Dr. Davis Matias de Oliveira (UEPB)
Examinador(a)

Dedico o presente trabalho a todos aqueles que me acompanharam direta ou indiretamente durante minha graduação. Com especial atenção a Sandra Moema de Melo Dantas minha mãe e a todos os familiares, amigos e professor que contribuíram para o desenvolvimento de um pensamento crítico.

AGRADECIMENTOS

É preciso seguir esse grande algoritmo cheio de variáveis que é a vida, as vezes tão similar ao loop infinito ou parecida com um condigo recursivo, onde o fim se confundi com o começo. O fim que te abre novas portas para novos começos, e mesmo assim você se pergunta: “que linguagem de programação devo utilizar? ”. É como subir uma escada que só dá para ver o degrau atual e você tenta fazer o melhor naquela instancia de escada, como em um Algoritmo Guloso. Contudo muitas vezes a carga é tão pesada que é preciso “Dividir para conquistar” e para isso é importante dominar bons Frameworks, assim como ter grades amigos que te acompanham por toda a graduação. E como o Caixeiro Viajante é preciso visitando disciplina por disciplina em uma rede que possuía um único protocolo: aprender o máximo possível com experiências transformadoras.

O conhecimento herdado dos professores me deu a capacidade de desenvolver este trabalho, por isso agradeço todo o corpo docente que fez o papel de interface para que os usuários/alunos conseguissem entender da melhor forma o conteúdo, em especial ao meu orientador Prof. Dr. Daniel Scherer que além de me acompanhar dês do primeiro semestre, foi um dos principais parâmetros para que eu ingressasse no curso de Computação.

Agradeço a Deus por me proporcionar a vida e a oportunidade de fazer parte de curso de Ciência da Computação da UEPB; Aos meus familiares em especial a minha mãe Sandra Moema de Melo Dantas; A todos os meus colegas de graduação aqui representados por: Yorras Gomes, Bruno Santos e Thassio Lucena; Ao NUTES(Núcleo de Tecnologia Estratégicas em Saúde) por me proporcional http://nutes.uepb.edu.br/ trabalhar com grandes profissionais da área de tecnologia e saúde, como o Professor Dr. Paulo Barbosa e o aluno e grande programador Yasser Nascimento, assim como todos que fizeram parte da equipe Paciente, vencedora do Hack Fest contra corrupção de 2017.

Você é o que acredita, o que você aprendeu,
e o que você viveu (Murilo Gun)

RESUMO

Este trabalho mostra uma análise de algumas Plataformas de Programação Visual (PPV) para robótica educacional e faz um breve discursão sobre alguns artigos que tem como objetivo a inclusão da robótica educacional na educação infantil e facilitar o ensino de programação, para assim apresentar uma sugestão de interface para uma PPV, reunindo as características encontradas na maioria dos PPV descritos nos artigos pesquisados com as melhores ferramentas presentes nas PPVs analisados. A interface sugerida neste trabalho servirá para programar o Robô Praxedes e terá como usuário crianças de 8 a 14 anos de idade, com o intuito de passar o conhecimento do pensamento computacional para esse público alvo. Esta interface irá auxiliar desenvolvedores a implementar PPVs garantindo uma boa usabilidade do software desenvolvido.

Palavras-Chave: Robótica Educativa. Pensamento Computacional. Plataforma de Programação Visual. Usabilidade.

ABSTRACT

This paper presents an analysis of some Visual Programming Platforms (PPV) for educational robotics and makes a brief discussion about some articles that aim to include educational robotics in early childhood education and to facilitate the teaching of programming, in order to present a suggestion of interface for a PPV, gathering the characteristics found in most PPV described in the articles searched with the best tools present in the analyzed PPVs. The interface suggested in this work will be used to program the Praxedes Robot and will have as user children from 8 to 14 years old, in order to pass the knowledge of computational thinking to this target audience. This interface will help developers to implement PPVs ensuring good usability of the developed software.

Keywords: Educational robotics. Computational thinking. Visual Programming Platform. Usability.

SUMÁRIO

1. INTRODUÇÃO

Jeannette Wing apresentou para o mundo a teoria de que o pensamento computacional é uma competência essencial à todos os seres humanos e não apenas para cientistas da computação e, mais precisamente, de programadores (Wing, 206). Já Mitchel Resnick (2009) defende a necessidade de incorporar o ensino de programação já nos primeiros anos do Ensino Fundamental (K-12 nos Estados Unidos) como forma de desenvolver, nos alunos, competências e habilidades relacionadas ao Pensamento Computacional (apud Queiroz, Sampaio e Santos. 2016).

Para ser introduzir o conteúdo do Pensamento Computacional na educação infantil são necessárias ferramentas e métodos motivacionais para facilitar o entendimento e o engajamento das crianças. A Robótica Educacional vem sendo utilizada como uma técnica de aprendizado que permite desenvolver atividades e como ferramenta que estimula a criatividade dos alunos devido a sua natureza dinâmica e interativa, além de servir de motivação para estimular o interesse dos alunos no ensino tradicional (Gomes, Areias, Henriques e Mendes. 2008).

Contudo para fazer uso da Robótica Educacional, o aluno precisa ter conhecimento da sintaxe de uma linguagem de programação, requisito este que, por si só, já não é nada simples. Implica saber quais são os comandos que existem e os parâmetros necessários e/ou possíveis de se usar para cada comando. Além, obviamente, da grafia correta de cada comando e seu(s) parâmetro(s). Tal dinâmica pode ser um problema para alunos de séries iniciais e/ou que estão tendo o primeiro contato com o mundo computacional.

Uma alternativa a estas interfaces baseadas em linhas de comando são as interfaces baseadas em Linguagens de Programação Visual (VPL - *Visual Programming Language*) que, através de uma Plataforma de Programação Visual (PPV), fornecem uma metáfora para ajudar o usuário a criar uma determinada ação com um mínimo de treinamento (Sobrinha, Nascimento, Gomes e Neto. 2016). Segundo (Pasternak, 2009), às PPVs reduzem a carga cognitiva sobre os estudantes que aprendem sua primeira linguagem de programação.

O grupo de Robótica Educativa da Universidade Estadual da Paraíba possui um Kit de robótica de baixo custo (Silva e Scherer. 2013), porém a programação do Kit ainda é baseada no software do Arduino, ou seja, baseado em linhas de comando. Considerando que o foco deste trabalho são alunos de séries iniciais abordando o uso de robótica de baixo custo,

este trabalho apresenta o estudo para o projeto de um interface para uma PPV, associado aos componentes do Kit de robótica do grupo.

Este trabalho tem como objetivo analisar Plataformas de Programação Visual (PPV) voltadas a Robótica Educativa com um olhar crítico no uso das mesmas na educação infantil, para assim desenvolver um protótipo de um software para programação visual de robótica que irá controlar o Robô Praxedes (Silva e Scherer. 2013) com finalidade de introduzir conceitos computacionais na educação de crianças de 8 e 14 anos de idade, essa faixa etária foi escolhida devido ao nível de leitura das crianças baseado no Pacto Nacional pela alfabetização na Idade Certa – PNAIC. (SEEDF, 2018)

Objetivo Geral

Projetar uma interface para uma Plataforma de Programação Visual (PPV) de fácil entendimento e atrativa para que crianças de 8 a 14 anos de idade possam utiliza-la afim de praticarem a programação utilizando um robô construído com peças de baixo custo (Robô Praxedes).

Objetivos Específicos

- Analisar e comparar artigos que utilizam algum tipo de plataforma de programação visual para alcançar seu objetivo;
- Levantar pontos positivos e negativos de algumas Plataformas de Programação Visual e fazer um comparativo entre elas;
- Sugerir um layout para plataforma de programação visual para robótica educacional;
- Apresentar um protótipo de uma PPV;
- Criar uma forma de representação de blocos lógicos;
- Fazer um levantamento de possíveis ícones que serão utilizados na interface sugerida;
- Executar uma atividade, construindo uma programação com os blocos sugeridos neste trabalho.

Metodologia

Primeiramente foi feita uma pesquisa de alguns artigos no site de Periódicos da CAPES, SBIE, RBIE e CBIE; buscou-se artigos que possuíam como palavra-chave:

- Robótica Educacional,
- Educação Infantil,
- Ambiente de Programação para Crianças,
- Gamificação e Pensamento Computacional.

Em seguida, foi feita uma análise destes artigos com finalidade de descobrir o que estava sendo produzido sobre interface de plataformas de programação visual para robótica educacional.

Foi feita também uma pesquisa de algumas Plataformas de Programação Visual voltadas a robótica educacional e o levantamento de pontos positivos e negativos destas plataformas. Em seguida, foi feita uma análise destas plataformas a fim de categoriza-las por diferentes tipos de representação de uma programação, além de encontrar pontos em comum e divergentes entre elas.

Com base nessas pesquisas, sugerimos uma interface para uma PPV, juntando as características encontradas na maior parte dos artigos pesquisados com as melhores ferramentas presente nas plataformas analisadas, montando assim um grande quebra cabeça de forma que todas as peças juntas estejam harmoniosamente ligadas.

Por fim colocamos em prática nosso protótipo, construindo nele um simples algoritmo para vermos quais dificuldades teríamos ao desenvolver uma programação.

2. FUNDAMENTAÇÃO TÉORICA

Para um melhor entendimento faz-se necessário descrever alguns termos.

2.1 Conceitos

2.1.1 Pensamento Computacional

Segundo Jeannette M. Wing o Pensamento Computacional (PC) é o termo utilizado para abreviar a frase “pensar como um cientista da computação”; que é uma atividade mental de formulação de problemas e construção de uma solução computacional. Para utilizar o PC faz necessário o uso da abstração de problemas de modo que a solução daquele problema possa ser aplicada a outros problemas de mesma natureza. Em outras palavras uma pessoa que faz o uso do PC não necessariamente precisa ser um cientista da computação ou saber programar um computador, está é uma habilidade fundamental para todas as pessoas (Wing, 2006).

O artigo da (BBC, 2019) mostra que o pensamento computacional possui 4 principais técnicas, são elas:

- **Decomposição:** Dividir um problema grande e complexo em parte menores.
- **Reconhecimento de Padrões:** Encontrar semelhanças entre e dentro de problemas.
- **Abstração:** Reconhecer os fatores importantes, ignorando detalhes irrelevantes.
- **Algoritmos:** Criar uma solução mostrando o passo a passo.

O mesmo artigo descreve que todas as 4 técnicas têm o mesmo grau de importância para se aplicar o pensamento computacional da seguinte forma: “Elas são como pernas em uma mesa - se uma perna estiver faltando, a mesa provavelmente entrará em colapso. Aplicar corretamente todas as quatro técnicas ajudará ao programar um computador. ” (BBC, 2019).

A Base Nacional Comum Curricular (BRASIL, 2018) faz menções ao Pensamento Computacional na área de Matemática para o ensino fundamental e descreve também a importância dos algoritmos e de seus fluxogramas como objetos de estudo nas aulas de matemática assim como a habilidade de reconhecimentos de padrões para construir pensamentos de generalizações. Isso aponta que Pensamento Computacional está dando os seus primeiros passos na educação fundamental brasileira.

2.1.2 Robótica Educacional

Segundo Maisonnette (2002) a Robótica Educacional trata-se de uma aplicação da tecnologia na área pedagógica que oferece aos participantes, sejam eles alunos ou professores, a oportunidade de propor e solucionar problemas difíceis; muito mais do que só observar formas de solução. Isto se deve, uma vez que com a Robótica Educacional o aluno é motivado a observar, abstrair e inventar; criando seus modelos e construindo assim seu conhecimento através de suas próprias observações.

Tendo isso em vista Maisonnette faz referência a Papert para reforçar a importância da Robótica Educacional na educação infantil: “aquilo que é aprendido pelo esforço próprio da criança tem muito mais significado para ela e se adapta melhor às suas estruturas mentais” (Maisonnette, 2002.).

O mesmo autor descreve que é notório o potencial da robótica como ferramenta interdisciplinar, visto que a solução de um novo problema pode ultrapassar as barreiras da sala de aula, na tentativa de buscar solução para o problema o aluno questiona professores e conteúdos de outras disciplinas para poder ajudá-lo a encontrar o caminho mais eficiente.

Podemos observar que a Robótica Educacional é uma forte ferramenta para a implementação do Pensamento Computacional na educação infantil, pois a robótica faz o papel de estimular o aluno ajudando na formulação e solução de problemas criados pelo próprio aluno. Além disto, pode facilitar o entendimento de conceitos teóricos e auxiliar o professor a fazer demonstrações práticas, conduzindo assim a uma melhor troca de informações entre o professor e o aluno.

2.1.3 Linguagem de Programação Visual

Linguagens de Programação Visual ou VPL (sigla em inglês para *Visual Programming Language*) são linguagens de programação que permitem aos usuários criar programas manipulando graficamente os elementos do software, em vez de especificá-los textualmente. Tornando assim a programação mais acessível, pois reduz as dificuldades que os iniciantes encontram quando começam a programar.

Além de ajudar o programador expondo as funcionalidades disponíveis na tela, geralmente representados como blocos fazendo com que ele não precise fazer o uso de uma documentação de linguagem de programação, as VPLs ajudam ao programador descrever o

programa para uma outra pessoa que não tenha familiaridade com VPLs, facilitando entender um código VPL (Craft.ai, 2015).

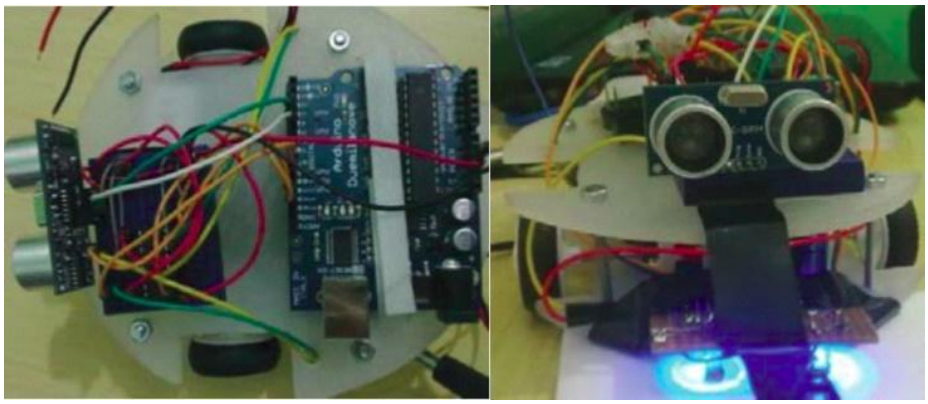
Segundo Erik Pasternak (2009), Linguagens de Programação Visuais destinam-se a fornecer metáforas para programadores. Estas metáforas muitas vezes se relacionam com atividades do mundo real e ajudam a criar um ambiente de programação mais natural e familiar aos usuários. O mesmo autor fala um pouco da história das VPLs, descrevendo que W. Sutherland é reconhecido como o criador da primeira VPL em 1966. Nos anos 70 e 80 as VPLs continuaram a serem desenvolvidas, porém elas encontraram dificuldades de se estabelecerem pois os recursos gráfico da época eram limitados, tendo em vista que para possuir uma boa linguagem de programação visual faz necessário o uso de bons recursos gráficos (Pasternak, 2009).

Para utilizar as VPLs é preciso uma Plataforma de Programação Visual (PPV), que por sua vez precisa possuir uma boa usabilidade para que o seu usuário consiga utilizá-la da melhor forma possível e se adaptar a mesma mais rapidamente. Neste trabalho serão analisadas algumas PPVs para assim sugerir uma estrutura de *layout* para PPV que tem como usuários crianças entre 8 e 14 anos.

2.1.4 Robô Praxedes

O grupo de pesquisa de Informática Educativa da Universidade Estadual da Paraíba desenvolveu um Kit de robótica de baixo custo baseado na plataforma Arduino. O Robô Praxedes (Figura 1) tem como finalidade simplificar o uso de plataformas Arduino, diminuindo a necessidade de conhecimentos complexos de eletrônica. Além disso, o Praxedes também tem como objetivo oferecer uma alternativa aos kits disponíveis no mercado com custo relativamente mais acessível (estima-se em torno de 10% do valor de kits comerciais proprietários, a exemplo de LEGO). Foi escolhida a plataforma Arduino pela sua disponibilidade, versatilidade e por seu grande número de componentes que podem ser agregados facilmente à placa Arduino (Silva e Scherer. 2013).

Figura 1 - Robô Praxedes



Fonte: Da Silava & Scherer (s.d)

2.2 Revisão da Literatura

Pesquisando no site de Periódicos da CAPEs, SBIE, RBIE e CBIE; buscou-se artigos que possuíam como palavra-chave:

- Robótica Educacional;
- Educação Infantil;
- Ambiente de Programação para Crianças;
- Gamificação;
- Pensamento Computacional.

Dentre os artigos encontrados, foram selecionados 14, dos quais:

- 5 dos periódicos da CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior),
- 4 dos periódicos da SBIE (Simpósio Brasileiro de Informática na Educação),
- 1 dos periódicos da RBIE (revista brasileira de informática na educação),
- 4 dos periódicos da CBIE (Congresso Brasileiro de Informática na Educação).

Todos eles serviram de base teórica para o desenvolvimento deste trabalho. Dos 14 artigos selecionados para a revisão de literatura, se destacaram 8 artigos por mostrarem experiências práticas com o uso de alguma plataforma de programação para Robótica Educacional.

A maioria dos artigos expõe o desenvolvimento de uma Plataforma de Programação Visual (PPV) para a Robótica Educacional, com diferentes públicos alvos e diferentes finalidades:

- Inclusão digital (Aquino, Farias, Crispim, Pinto e Vasconcelos. 2017) (Barros, 2008);
- Ensinar computação de forma simples e de fácil entendimento (Maia, Silva, Pazoti, Almeida e Pereira. 2014);
- Facilitar o acesso a programação de robô (Costella, Trentin, Amarante e Teixeira. 2017) (Almeida, Netto, Silva, Custódio. 2017) (Alves, Sampaio e Elia. 2014);
- Ensinar computação e outras disciplinas de forma divertida (Sobrinha, Nascimento, Gomes e Neto. 2016);
- Passar o conhecimento do Pensamento Computacional (Queiroz, Sampaio e Santos. 2016).

A justificativa mais encontrada nos artigos para o desenvolvimento de um PPV voltada a programação de robótica é o barateamento de custos para possuir kits de robótica. Esses artigos apresentam duas estratégias para conseguir alcançar seu objetivo, são elas:

- Desenvolver PPV que permita programar sistemas embarcados baseados em Arduino (Sobrinha, Nascimento, Gomes e Neto. 2016) (Alves, Sampaio e Elia. 2014) (Queiroz, Sampaio e Santos. 2016); ou
- Construir ambiente de aprendizagem de robótica gratuito e interativo para programação de robótica a distância.
    - Um sendo um ambiente para programação remota de um kit robótico Lego Mindstorms NXT (Almeida, Netto, Silva e Custódio. 2017), não tendo, portanto, controle do posicionamento do kit;
    - Já o outro, um ambiente para programação remota de um braço mecânico pois ele sempre estará no mesmo lugar ao iniciar a execução de um código (Costella, Trentin, Amarante e Teixeira. 2017).

Um ponto a se ressaltar é que a metodologia de programação por blocos está presente na maioria das PPVs propostas, tendo em cada bloco uma imagem ou cor associada para facilitar o entendimento da funcionalidade do bloco pelo usuário. Porém, os artigos não mostram quais métodos foram utilizados para definição de tais imagens ou cores.

Já sobre o layout das interfaces, foi concluído que predominantemente as PPVs apresentadas nos artigos analisados mostram (Figura 2):

- Menu principal na parte superior da tela;
- Comandos ou blocos lógicos listados mais à esquerda;

- Área de trabalho (que vem a ser o local da interface destinado ao desenvolvimento da programação) mais à direita.

Figura 2 - Representação Generalizada das interfaces apresentadas nos artigos pesquisados

A ausência de trabalhos futuros buscando a melhoria nas interfaces, está presente em todos os artigos pesquisados, nenhum artigo mostra a preocupação de como o seu usuário reage com a interface proposta, contudo apresentam que conseguem atingir seus objetivos: a inclusão digital. As três PPVs RoboEduc, DuinoBlock e DuinoBlocks4Kids tiveram destaques por possuírem mais explicações sobre suas interfaces e por fazerem testes e experimentos a PPV desenvolvida em oficinas de robótica assim como analisar o comportamento dos alunos nessas oficinas.

2.3 PPVs de destaque

2.3.1 RoboEduc

O artigo mostra a importância da inclusão digital citando o Relatório Anual da Organização das Nações Unidas (ONU) do ano de 2007, no qual declarou que as novas tecnologias são facilitadoras de avanços sociais. E desenvolveu a PPV RoboEduc que apresenta ao usuário dois módulos de uso, um destinado ao professor (autoria) e o outro ao aluno, ambas com diferentes funcionalidades. As interfaces foram desenvolvidas com biblioteca Qt 4.3 e ele possui uma linguagem de programação própria com o mesmo nome da PPV.

A avaliação da implementação do software em sala de aula foi feita através de entrevista com os professores das crianças que participaram da experiência, com a diretora da instituição e com os monitores da oficina de robótica educacional. O autor declara que as professoras e a diretora se sentiram satisfeitas com a implementação. A diretora declarou a

importância do projeto para educação de crianças e as professoras descreveram algumas mudanças no comportamento das crianças: “responsáveis e comprometidos com as atividades em sala de aula” (Barros, 2008).

Foi notado a partir dos comentários descritos dos professores e diretora que essa implementação ajudou a incluir as crianças no meio digital, porém não houve levantamento de dados para um relato mais preciso da experiência. O comparativo das notas dos alunos, antes e depois da experiência, ajudaria a consolidar o objetivo alcançado.

Os dois módulos de uso do RoboEduc (aluno e autoria) fazem com que a PPV seja interessante de ser aplicada em sala de aula, pois o professor (módulo autoria) pode cadastrar novos hardwares e novas tarefas, conduzindo assim o aluno em sua aprendizagem. Porém não há referências a estudos extras envolvendo a análise da usabilidade da plataforma desenvolvida.

2.3.2 DuinoBlock

Segundo os autores do artigo, com o intuito de promover a robótica educacional, o *DuinoBlock* tem o objetivo de facilitar a programação de robôs baseados em plataformas Arduino. Baseado nas Leis da Simplicidade proposta por Maeda apud (Alves, Sampaio e Elia. 2014), o ambiente foi desenvolvido com a metodologia de programação por blocos. Possui blocos de comandos divididos por categoria, sendo que cada categoria é representada por uma cor. Cada bloco se encaixa a outro se suas conexões fazerem sentido sintático (Alves, Sampaio e Elia. 2014).

Para a construção do ambiente virtual de programação *DuinoBlock* foi feita uma pesquisa de algumas PPVs e divididas em dois tipos de metodologia de criação de código visual: “a primeira é caracterizada pela formação de empilhamentos ordenados, em que o fluxo de execução se dá de cima para baixo” e “ a segunda abordagem é aquela em que o algoritmo tem uma estrutura de diagrama, onde o fluxo de execução segue por arestas e nós. ” (Alves, Sampaio e Elia. 2014).

Contudo os autores desta plataforma poderiam ter mencionado uma terceira metodologia de criação de código virtual, a criação de código que se dá da esquerda para a direita, estruturando os blocos lógicos na horizontal (ex.: Lego Boost, Lego Mindstorms EV3). Além disso os autores implementam na interface de sua plataforma elementos como a linguagem nativa (exibindo a programação desenvolvida pelo usuário (em blocos) em uma

linguagem de programação) e a funcionalidade que dá ao usuário a opção de criar seus próprios blocos lógicos, deixa a plataforma um pouco mais complexa.

2.3.3 DuinoBlock4kids

Destinada para alunos do ensino fundamental I o DuinoBlocks4Kids (DB4K) possui uma interface colorida simples, seus blocos possuem desenhos (ícones) diretamente relacionadas com as funções dos blocos. Esta PPV é uma evolução do *DuinoBlock*. Para se adaptar ao público foi necessário simplificar os seus conjuntos de blocos, retirando informações de detalhes relacionados a mecânica e partes físicas de robôs.

Para avaliar a PPV foram feitas oficinas com um total de 8 crianças de 1º, 2º ,3º e 4º ano do ensino fundamental. Segundo o autor, a avaliação da interface foi positiva: "todas as crianças demonstraram bastante facilidade no seu uso, não encontrando maiores problemas em localizar e organizar os blocos para a construção dos programas". Afirma também que as crianças escolhiam os blocos não pela leitura e sim por sua ilustração (Queiroz, Sampaio e Santos. 2016).

Assim como o *DuinoBlock*, o DB4K também mostra na sua interface a programação construída pelo usuário em linhas de código, a medida em que o usuário constrói sua programação visual. Em minha opinião, não é interessante pois pode tirar o foco do usuário iniciante. No caso do DB4K pode ser considerado mais grave, tendo em vista que seu público alvo são alunos do ensino fundamental I. O autor do artigo não explica como os ícones/desenhos expostos na interface da plataforma foram selecionados.

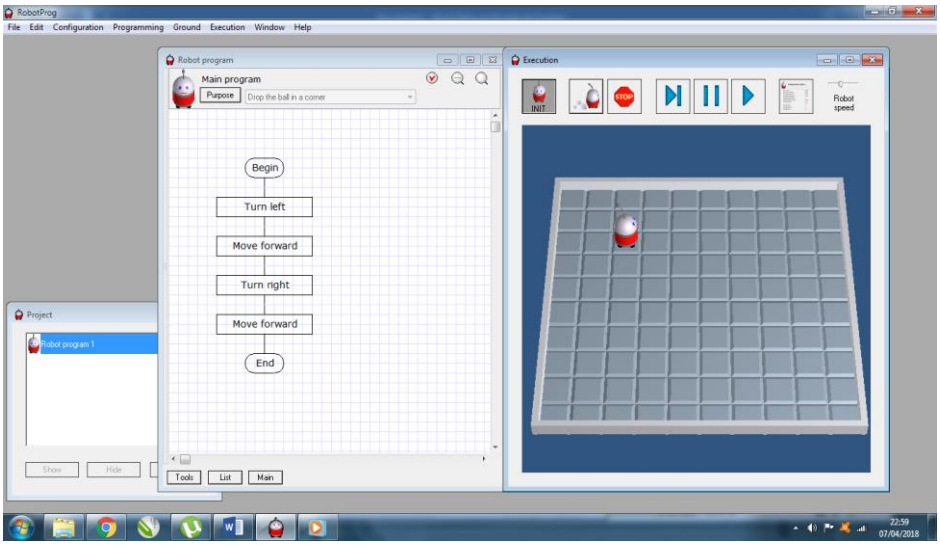
3. ANÁLISE DE PPVS RELACIONADAS

Este trabalho tem como objetivo analisar algumas PPVs, voltadas a Robótica Educacional de tal forma que uma criança de 8 à 14 anos de idade consiga se apropriar do conceito de Pensamento Computacional. Para isso foi feita uma pesquisa de algumas PPVs. Os ambientes de desenvolvimento foram escolhidos com a finalidade de categorizar tipos de PPVs, levantando em consideração a identificação dos pontos positivos e negativos de cada plataforma, para assim fazer uma análise e propor uma PPV para o Praxedes.

3.1 RobotProg

O software RobotProg possui uma forma de desenvolvimento diferenciada, nele o usuário irá construir um programa através de um fluxograma, arrastando os elementos do fluxograma para área de trabalho e interligando-as de uma forma lógica, para assim poder controlar o robô virtual. Simples de usar o RobotProg disponibiliza uma documentação e um tutorial que pode ser encontrado em seu site (RobotProg, 2019) (Figura 3).

Figura 3 - Interface da Plataforma RobotProg



Fonte: Capturarão de tela do Software RobotPtoh

3.1.1 Pontos Positivos:

- Divisão de níveis de complexidade, mostrando menos ferramentas quando o nível do desafio é baixo e expondo mais ferramentas (blocos) quando o nível do desafio é maior;
- Ferramenta “comentário”, introduzindo as boas práticas de programação para iniciantes;
- Código feito através de um fluxograma, deixando mais fácil a programação visual;
- Simulador virtual com robô animado deixando mais divertido a programação.

3.1.2 Pontos Negativos:

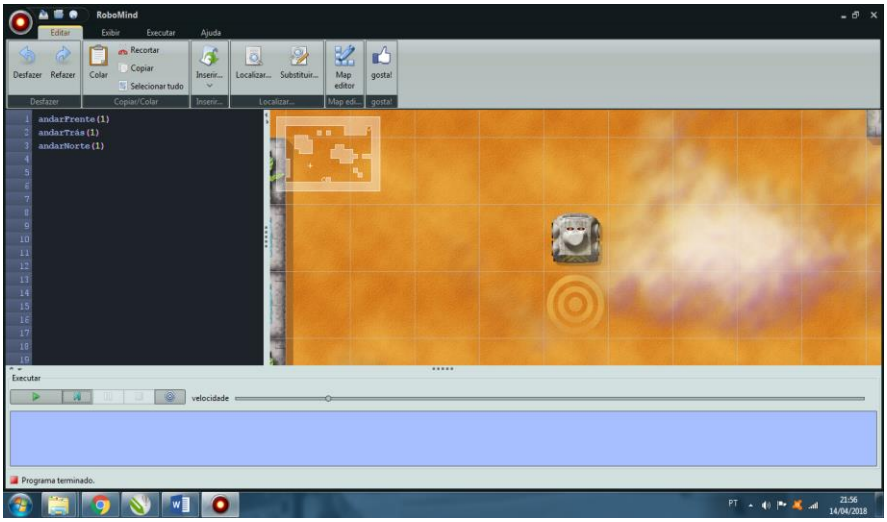
- Menus de ferramentas suspensos, essa manipulação pode deixar a interface da PPV muito difícil de ser usada, estressando o usuário;
- Não aceita comandos vindo do teclado, para deletar um bloco é preciso usar a ferramenta “Borrar” e não possui a função de desfazer, estressando o usuário avançado por não conseguir utilizar atalhos;
- Para cada passo é preciso adicionar um novo bloco fazendo com que o fluxograma fique muito extenso e o usuário faça ações repetidas muitas vezes;
- Elemento do fluxograma condicional aberto a escrita, o elemento de condicionais requer parâmetros para poder ser executado, além disso é necessário uma forma de comparação (ex: maior que , menor que, igual etc), no qual a plataforma deixa aberta para que o usuário preencha toda a estrutura de comparação fazendo com que o usuário inicial não consiga utilizar estes elementos;
- PPV com muitos bugs (ex: ao editar o nível de complexidade o programa para de funcionar impossibilitando de o usuário utilizar mais ferramentas).

3.2 RoboMind

Software gratuito que possui uma linguagem de programação própria (Robô), a qual consiste em um conjunto de regras voltadas para a programação de um robô usando os mesmos princípios de outras linguagens de programação. Com instruções básicas de

programação o usuário poderá comandar ações para o robô virtual realizar, tais como: andar, virar, olhar ao redor, pegar objetos, pintar, etc (Figura 4). (RoboMind, 2016)

Figura 4 - Interface da plataforma RoboMind



Fonte: Capturarão de tela do Software RoboMind

3.2.1 Pontos Positivos:

- Associado a um curso EAD com metodologias de gamificação;
- Ferramenta de refazer e desfazer bem expostas, facilitando com que o usuário não fique perdido na hora de executar sua programação;
- Possui uma linguagem de programação simples, possibilitando entendimento mais fácil pelo usuário (pseudocódigo);
- Simulador virtual com vários ambientes (ex: spiral, passBeacons, LineFollower, etc) deixando mais divertida a execução da programação;
- Mensagem de erros em primeira pessoa (ex: ‘eu não posso passar por objetos.), Fazendo com que a interface seja mais amigável;
- Controlar o robô virtual através de um controle. Assim, à medida que o robô é controlado através deste controle, a programação dos passos/ações que o robô executa são construídas automaticamente. Isto permite ao usuário ver como um programa é/pode ser construída.

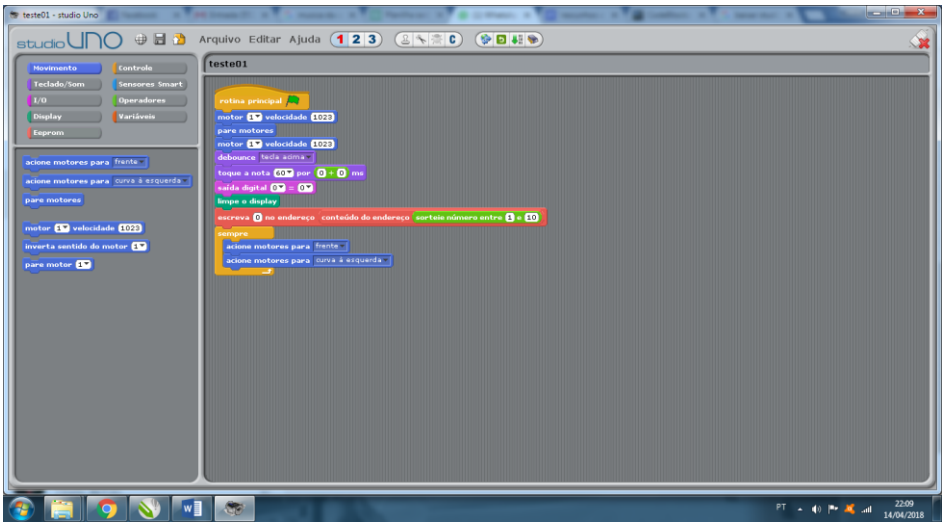
3.2.2 Pontos Negativos:

- Não possui blocos lógicos, o programa desenvolvido pelo usuário que irá controlar o robô virtual é construído com códigos selecionados (ex: Andarfrente(1)), deixando a programação mais textual do que visual;
- A conexão com um hardware (Robô) é pouco abordada, fazendo com que o usuário utilize mais o robô virtual presente na PPV;
- Não dá para identificar que a logo do programa é um botão em sua interface, o que dificulta utilizar os recursos que são encontrados ao clicar na mesma.

3.3 Studio UNO

O Studio UNO é um ambiente de programação gratuito, baseado no modelo de programação em blocos. O Studio UNO é adequado para alunos a partir dos 8 anos de idade. Feito para programar com a linguagem de programação educacional Scratch, do MIT Media Laboratory, e alguns comandos da linguagem C, o que permite a usuários avançados importarem bibliotecas de códigos existentes. (Figura 5) (UNO, 2012)

Figura 5 - Interface da Plataforma Studio Uno



Fonte: Capturarão de tela do Software Studio Uno

3.3.1 Pontos Positivos:

- Escolha de nível de complexidade, mostrando mais ou menos blocos de funções, demonstrando uma preocupação com o nível de conhecimento do usuário.

- Simulador de placa para executar o código, mostrando na prática o comportamento de uma placa UNO 1.1;
- Layout de interface mais utilizado, blocos do lado esquerdo e área de trabalho (onde será desenvolvido o código) do lado direito, facilitando a adaptação dos usuários que já utilizaram plantamos desse gênero;

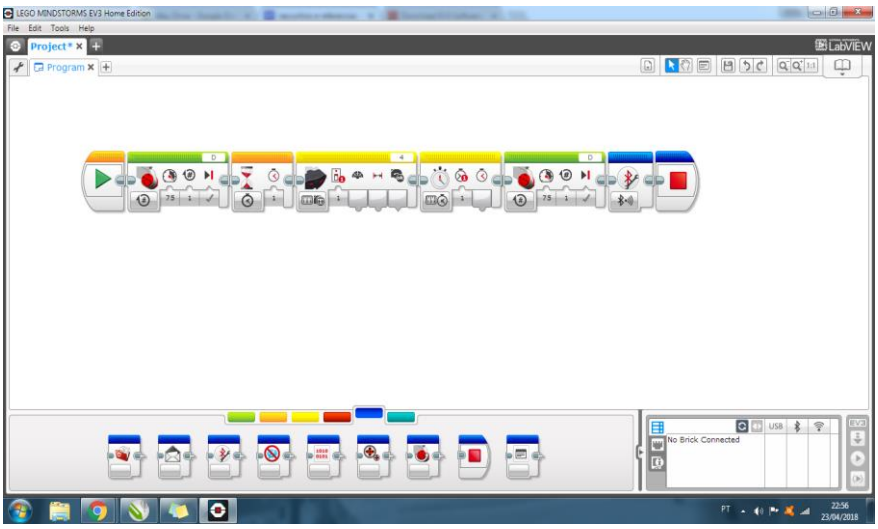
3.3.2 Pontos Negativos:

- Só programa o robô UNO, do mesmo desenvolvedor deste PPV.

3.4 Lego Mindstorems EV3

O software MINDSTORMS EV3 é uma versão de uma série de ambientes de desenvolvimento produzida pelo Grupo LEGO. Essa versão possui uma interface simples, porém moderna, trazendo maior conforto ao programar eventos mais avançados e facilitando a vida de novos programadores. O ambiente vem com alguns desafios de programação para os diferentes tipos de robôs da LEGO e com a interface (*drag-and-drop*) arraste e solte. (Figura 6) (EV3, 2019)

Figura 6 - Interface da Plataforma EV3



Fonte: Capturarão de tela do Software EV3

3.4.1 Pontos Positivos:

- Tutoriais em vídeo de:
  - Como usar a PPV,

- Como enviar uma programação para um kit,
- Como montar um robô específico,
- Exemplos de atividades,
- Mostra em função algumas ferramentas da PPV.

● Área de trabalho grande e limpa, deixando o usuário com mais espaço na hora de programar;

● Blocos com ícones, aderindo a programação visual e facilitando o entendimento da função do bloco pelo usuário;

● Conexão com o hardware bem definida e com boa quantidade de informações, deixando mais interessante a conexão e a execução da programação pelo robô, pois o usuário possui todas as informações do hardware na plataforma, fazendo com que o usuário não precise ficar analisando o hardware e o software ao mesmo tempo.

3.4.2 Pontos Negativos:

● Plataforma destinado apenas para Kits Lego, impossibilitando o uso com outros kits de robótica;

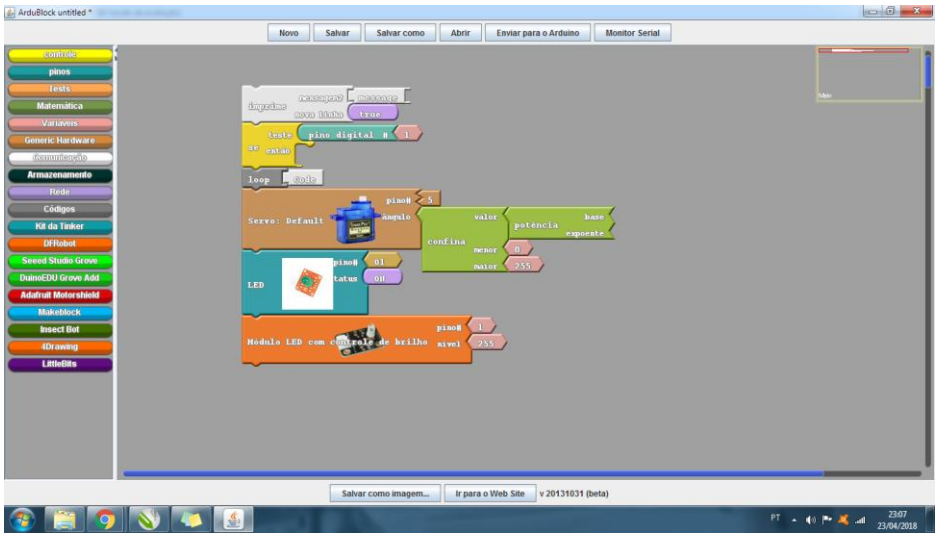
● Poderia ter mais aspectos de Gamificação, para engajar ainda mais o usuário;

● As categorias de blocos são identificadas apenas por cores, ou seja, apenas por um elemento gráfico, dificultando o reconhecimento das categorias.

3.5 Ardublock

Esta PPV tem o objetivo de introduzir usuários que não possuem o conhecimento de linguagens de programação, a desenvolver projetos para o Arduino. Em outras palavras ela é uma interface que possui a técnica de programação por blocos lógicos especificamente para a linguagem Arduino (Figura 7). (Ardublock, 2018)

Figura 7 - Interface da Plataforma ArduBlock



Fonte: Capturarão de tela do Software ArduBlock

3.5.1 Pontos Positivos:

- Layout de interface mais utilizado, blocos do lado esquerdo e área de trabalho (onde será desenvolvido o código) do lado direito, facilitando a adaptação dos usuários que já utilizaram plantamos desse gênero;
- Divisão de área de trabalhos e menu de blocos lógicos móvel, podendo modifica-las quando necessário, deixando a área de trabalho com mais espaço ou ampliando as informações dos blocos na aba blocos lógicos.

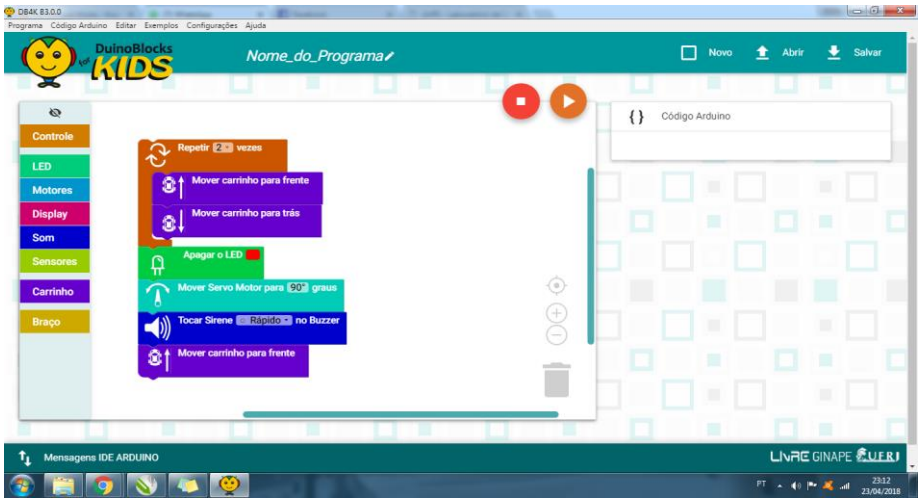
3.5.2 Pontos Negativos:

- Não possui ajuda, dificultando ainda mais a adaptação à plataforma;
- Muitas categorias com muitos blocos, confundindo o usuário iniciante.

3.6 DuinoBlocks4Kids (DB4K)

Esta PPV foi desenvolvida com base na plataforma *DuinoBlock* porém com uma interface mais infantil. O DB4K foi criado com o foco em ensinar conceitos básicos de programação para crianças e permite o desenvolvimento em blocos para placas de prototipagem eletrônica Arduino (Figura 8).

Figura 8 - Interface da Plataforma DB4k



Fonte: Capturarão de tela do Software ArduBlock

3.6.1 Pontos Positivos:

- Blocos com funções simples, facilitando entender qual a funcionalidade de cada bloco;
- Categorias de blocos não apenas por cor, mas também por uma representação textual;
- Opção de ocultar menu de blocos lógicos, fazendo com que a área de trabalho fique com mais espaço para ser usada pelo usuário;
- Exemplos de códigos prontos, facilitando o entendimento pelo usuário através de exemplos de programações prontas;
- Ferramentas com representações simples e bem localizadas.

3.6.2 Pontos Negativos:

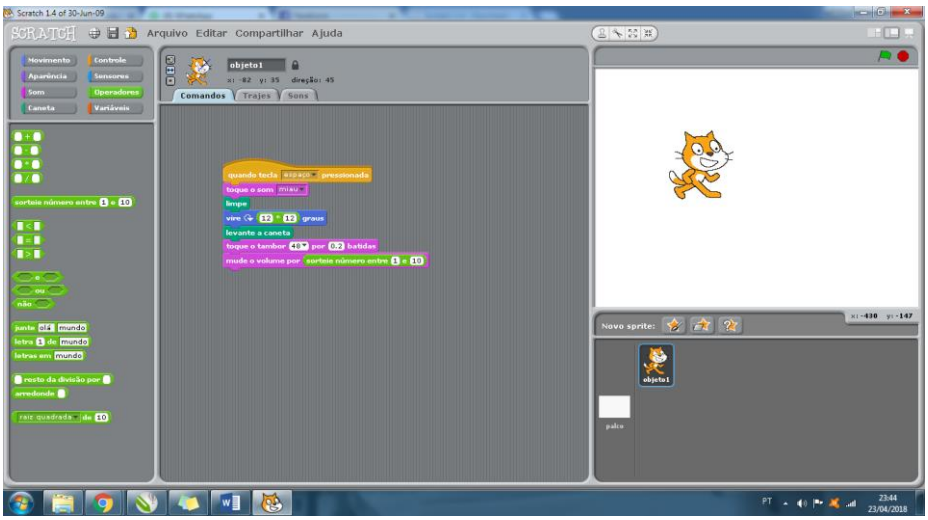
- Mostrar o programa em linha de código é bom, mas não para os iniciantes;
- Não disponibiliza tutorial.

3.7 Scratch

Umas das PPVs mais conhecidas é o Scratch, que foi criada em 2007 pelo Instituto Tecnológico de Massachussets (MIT) e pelo grupo KIDS da Universidade da Califórnia. Com o objetivo de ajudar crianças de 8 anos no aprendizado de conceitos matemáticos e computacionais, é ideal para pessoas que estão começando a programar (Figura 9). Além de trabalhar com a técnica de programação em blocos, o Scratch permite a personalização do

seu projeto através da incorporação de imagens e sons externos, bem como a possibilidade de desenhar e gravar sons dentro da ferramenta. (Scratch, 2019)

Figura 9 - Interface da Plataforma Scratch



Fonte: Capturarão de tela do Software Scratch

3.7.1 Pontos Positivos:

- Faz o uso de um simulador virtual, assim não é obrigatório possuir um kit de robótica para ter o feedback de sua programação;
- Blocos simples que usam pouco espaço na vertical, fazendo com que seja melhor usada a área de trabalho, pois sua programação é na vertical;
- Ajuda rápida, tutorial muito bem desenvolvido, mostrando exemplos simples e descrevendo as funcionalidades de cada bloco;
- Exemplos de códigos prontos, facilitando o entendimento pelo usuário através de exemplos de programações prontas.

3.7.2 Pontos Negativos:

- Um número grande de blocos lógicos, podendo deixar usuário iniciante estressado e perdido;
- O bloco de iniciar um programa não se encontra na área de trabalho, fazendo com que o usuário entre na categoria “controles” e em seguida arraste um bloco de iniciar para área de trabalho, só assim sua programação poderá ser executada;

- A categoria “controles” deveria se localizar em primeiro lugar, tendo em vista que é preciso arrastar um bloco de iniciar para começar a programar;
- Não exclui os blocos com o botão “delete” do teclado, é preciso arrastar para fora da área de trabalho.

3.8 Lego Boost

Destinado para dispositivos móveis, este aplicativo possui gráficos de boa qualidade, interface lúdica e usa a metodologia de gamificação para prender a atenção de seus usuários (Figura 10).

Figura 10 - Interface da Plataforma BOOST

Fonte: Capturarão de tela do Software BOOST

3.8.1 Pontos Positivos:

- Representação dos Robôs físicos animados, o usuário faz a ligação entre o robô físico e sua representação no aplicativo, deixando a interface do aplicativo mais elucidativa;
- Os programas desenvolvidos ao serem salvos mostram uma minivizualiação, que podem serem vistas antes mesmo de abrir um determinado programa;
- Tutorial passo a passo de como montar o kit abordado;
- Blocos possuem uma alta explicação, que pode ser visualizada ao segurar o click por alguns segundos;
- Uso da metodologia de gamificação deixando mais desafiador programar.

3.8.2 Pontos Negativos:

- Feito apenas para dispositivos móveis é destinado apenas para produtos Lego, como este aplicativo é muito complexo com muitas ferramentas, seria melhor se fosse destinado para desktop;
- Programas gravados possuem uma minivizualiação antes de serem abertos. A medida que o número de programas salvos aumenta fica difícil de achar um programa específico;
- Criar um bloco lógico é uma boa possibilidade, porém muito complexa para o público alvo deste trabalho;
- Blocos lógicos de PPVs da linha Lego são adequados para telas grandes, pois os seus ícones muitas vezes são representação de uma parte do seu Kit, como motores, sensores e garras, mas em uma tela pequena fica difícil fazer uma ligação entre bloco e função.

4. COMPARAÇÃO DE PPVS

Inicialmente foram levantados 10 ambientes de programação em robótica com intuito de avaliar suas interfaces, listar pontos positivos e negativos, e fazer um comparativo entre todos eles. Dentre esses softwares foram excluídos dois por fugirem das características de uma PPV (a IDE padrão do Arduino (ARDUINO, s.d.) e o software Maria Mole (Porto, s.d.)). Dentre as PPVs foi notado que existem 4 maneiras de representar a programação desenvolvida em sua área de trabalho, são elas:

Por Fluxograma: construção de um procedimento (algoritmo) graficamente através de símbolos geométricos interconectados. Ex. RobotProg. (Figura 11a).

Por Linhas de Códigos Simples: através de um pseudocódigo utilizando uma linguagem simples e natural ao usuário. Ex: RoboMind. (Figura 11b).

Blocos Lógicos na Horizontal: construir um algoritmo encaixando blocos lógicos e organizando-os na horizontal. Ex: Lego Boost. (Figura 11c).

Blocos Lógicos na Vertical: construir um algoritmo com blocos lógicos organizados na vertical. Ex: Scratch. (Figura 11d).

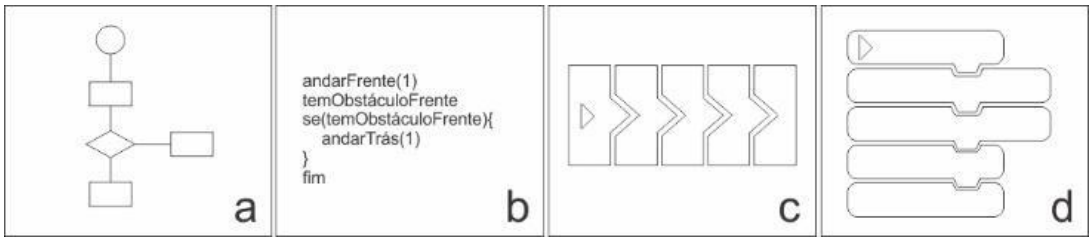


Figura 11 - Formas de programação encontradas

Os ambientes de programação da linha Lego fazem pouco uso de palavras em suas interfaces fazendo jus ao termo Programação Visual. Além disso, seus materiais de ajuda e tutoriais são expostos em seu menu principal facilitando a inicialização de usuários que nunca utilizaram o software. Os ambientes selecionados neste trabalho da linha lego são o EV3 desenvolvido para desktop e o Lego Boost destinado para dispositivos móveis.

Já os softwares que seguem a linha do Scratch usam mais textos em seus blocos lógicos; normalmente esta linha de softwares organizam seus blocos lógicos verticalmente. Alguns destes softwares fazem o uso de ícones nos blocos para facilitar o entendimento

(ArduinoBlock e o DB4K). O Studio UNO apesar de ser uma cópia quase que total do Scratch (quando se trata de interface) possui uma funcionalidade que o usuário pode escolher o nível de complexidade dos blocos lógicos, mostrando mais blocos quando o usuário escolher a opção de maior nível de conhecimento.

A simulação virtual destaca-se no ambiente de programação RoboMind, mas também existe para as plataformas Scratch e RobotProg. Visualiza-se virtualmente como seria o comportamento de um robô ao receber os comandos definidos pelo usuário. Já o Studio UNO apresenta um simulador de uma placa Arduino. Ter uma simulação virtual em uma plataforma de programação voltada a robótica ajuda a incluir usuários que não possuem o hardware de robô.

5. CONSTRUÇÃO DE UMA PROPOSTA DE INTERFACE

O desafio desta parte do trabalho é de juntar todos os pontos positivos das PPVs relacionadas aqui em uma mesma interface e tendo os pontos negativos como paramentos para não seguir. Como fundamento, temos a comparação dos artigos aqui trabalhados que apresentam o estado da arte nas áreas de usabilidade, Plataforma de Programação Visual e da implementação Robótica Educacional na educação básica para crianças de 8 a 14 anos.

5.1 Tela de Apresentação

Antes do usuário ir para o ambiente de desenvolvimento do programa com blocos lógicos, foi notada a necessidade de uma tela de apresentação baseada no Lego Mindstorms EV3; contendo material de apoio (ex.: tutorial em vídeos e manual passo a passo de como usar o software), assim como a conexão do software a um curso de robótica online, onde o professor ou responsável poderá ver o desenvolvimento do aluno, tal qual ocorre no RoboMind.

Ainda na tela de apresentação (Figura 12) foi sugerido o uso de Gamificação, baseado no aplicativo para dispositivos móveis Lego Boost. Assim, chama-se mais atenção do usuário e o estimula a resolver os desafios propostos pelo software. Ademais, o aluno pode adquirir conhecimentos computacionais à medida que resolve estes desafios.

Figura 12 - Modelo sugerido de página de apresentação

Criamos uma breve narrativa onde o personagem com características de um robô tenta reconstruir seu planeta natal e precisa da ajuda do usuário deste software. A medida que o usuário ajuda o personagem conseguindo completar os desafios propostos, novos desafios são desbloqueados e é possível ganhar estrelas digitais (medalhas) à medida que o usuário desenvolve um programa abrangendo todo os requisitos da atividade proposta.

Por fim, um botão destacado na cor verde, para dar ideia de partida/avançar, no canto inferior direito da tela, fazendo com que o usuário percorra toda a tela para chegar neste botão que irá direciona-lo ao ambiente de desenvolvimento onde o mesmo poderá programar livre de desafios propostos pela parte gamificada do software.

5.2 Ambiente de desenvolvimento

Foi notado que todos os ambientes de programação visual descritos nos artigos pesquisados para o desenvolvimento deste trabalho possuem uma estrutura padrão composta de menu de blocos lógicos a esquerda da tela e a área de trabalho, espaço destinado ao desenvolvimento da programação feita pelo usuário que irá controlar um possível robô, mais a direita, assim como a maioria dos softwares relacionados pesquisados, tendo isso em vista foi decidido por usar a mesma estrutura neste trabalho como mostra a Figura 13.

Figura 13 - Proposta Sugerida de Layout

5.3 Ícones

Uma das maiores preocupações deste trabalho é em relação a compreensibilidade dos ícones apresentados, principalmente os ícones que representam as categorias e blocos lógicos, pois foi notada uma dificuldade de entender algumas representações icônicas das PPVs desenvolvidas nos artigos aqui debatidos, assim como nas PPVs relacionadas descritas neste trabalho (Seção 4). Portanto foram levantadas 3 imagens para cada ícone da PPV aqui sugerida, essas imagens foram retiradas de algumas PPVs aqui comparadas e criadas com finalidade de atender da melhor forma o público alvo (Figura 14).

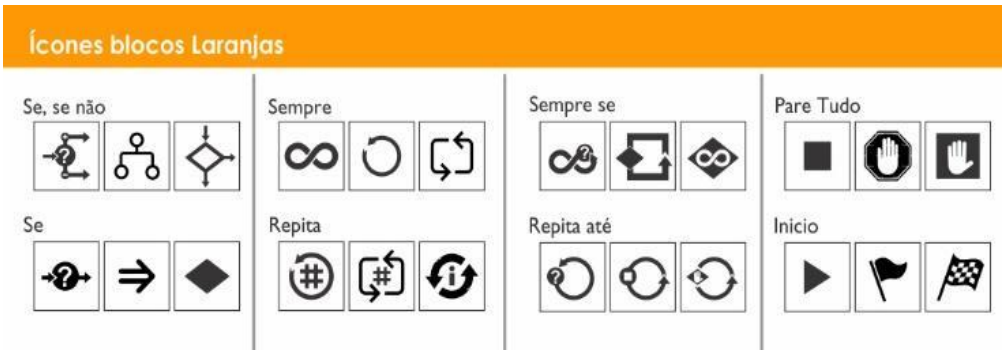


Figura 14 - Exemplos de possíveis símbolos para determinado ícone no PPV sugerido

Contudo é preciso executar um Teste de Compreensão sugerido por Eliana Formiga em seu livro Símbolos Gráficos Métodos de Avaliação de Compreensão (Formiga, 2012), teste este que ficará para trabalhos futuros devido à falta de tempo de ser discutido aqui. Também será preciso fazer o teste de usabilidade da interface.

5.4 Nível de complexidade

Os níveis aqui definidos foram escolhidos a partir da idade escolar brasileira, colocando mais elementos gráficos e substituindo textos nos níveis mais baixos. Já para níveis maiores, aumentou-se a relação de texto. O ensino fundamental brasileiro possui 9 anos de duração atendendo estudantes entre 6 e 14 anos de idade (BRASIL, 2018). Tendo isso em vista foram definidos os seguintes níveis de complexidade:

- **Nível 1:** Baseados no Pacto Nacional pela alfabetização na Idade Certa – PNAIC (SEEDF, 2018) que estabelece a obrigatoriedade de alfabetizar todas as crianças até o final do 3º, decidimos pular os 2 primeiros anos do ensino fundamental pois em nossa

interface utilizamos linguagem textual, fazendo com que o primeiro nível da plataforma projetado neste trabalho seja destinado a crianças entre 8 e 10 anos de idade. Neste nível o usuário irá construir sua a programação na horizontal com blocos mais simples e possuindo menos funcionalidades disponíveis. Terá um espaço sinalizando onde deverá ser colocado o bloco, não possibilitando a poluição da área de trabalho ao construir o programa.

- **Nível 2:** Destinado a usuários com idade entre 10 e 13 anos, a programação será feita na vertical com elementos gráficos que permitem notar uma concatenação na programação, fazendo assim uma ligação da programação visual com a programação por linhas de códigos. Isso é importante pois um dos objetivos deste ambiente de desenvolvimento é ensinar o usuário a programar não só por elementos visuais (programação por blocos) como também prepará-lo para programação tradicional. Aqui o usuário terá acesso a mais elementos de programação do que no nível 1, ou seja, terá mais blocos do que o nível anterior.

- **Nível 3:** 14 anos em diante, nesse nível o usuário fica mais próximo de uma linguagem de programação por códigos. Nessa idade o aluno está saindo do ensino fundamental, portanto já terá adquirido habilidade com linguagem e conhecimento de técnicas matemática com Geometria, Álgebra, Grandezas e Medidas, Probabilidade e Estatística. A programação também será na vertical e o usuário poderá acompanhar o código Arduino no lado direito da tela a medida em que ele constrói sua programação com blocos na área de trabalho do ambiente de desenvolvimento.

5.5. Blocos Lógicos

Foram desenvolvidas duas representações de blocos lógicos, uma para o primeiro nível de complexidade e outra para os demais níveis. Para os blocos do nível 1, decidimos fazê-los compactos se aproximando de um quadrado, para que o usuário tenha mais espaço e assim construa uma programação na horizontal, seguindo a linha do Lego Boost (Figura 15a). Já nos demais níveis, os blocos foram baseados nos blocos do EV3 porém destinados a uma programação na vertical. Para isso precisamos deixar os blocos mais retangulares ocupando pouco espaço na vertical e fazermos com que a indicação de início de um programa seja entendida da melhor maneira possível. Como o ícone que representará esse bloco é um “play/start”, que é similar a uma seta apontando para a direita, o próximo comando deveria

vir na sequência do lado direto do bloco iniciar, deixando assim o bloco iniciar programa diferenciado dos demais (no nível de complexidade 2 em diante)(Figura 15b).

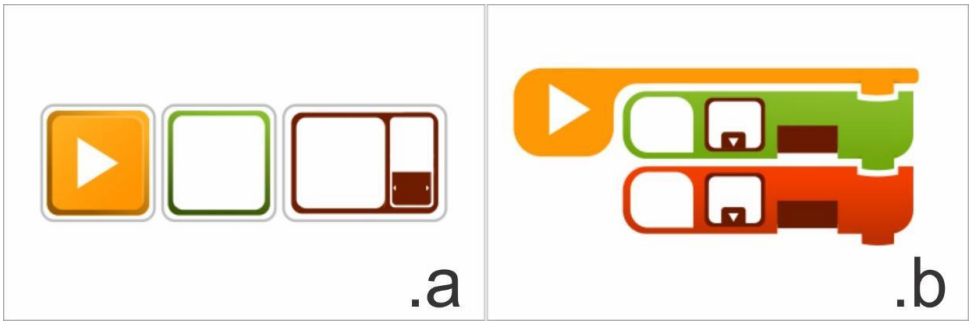


Figura 15 - Layout de blocos Lógicos

Após criamos o layout dos blocos (Figura 15) seria preciso saber quais funcionalidades cada bloco deveria ter, para listar essas funcionalidades nos deparamos com 3 linhas de pesquisa a serem seguidos:

- Se basear em PPV já consolidadas;
- Se basear nas funcionalidades do hardware (Robô Praxedes);
- Se basear em uma linguagem de programação.

Decidimos inicialmente seguir a 3ª linha, ou seja, seguir os blocos lógicos de uma plataforma já consolidada, que neste caso escolhemos a PPV Scratch, por ter blocos da categoria “controle” muito próximo a uma linguagem de programação. Contudo o Scratch possui muitos blocos e foi preciso simplificá-los, para assim reduzir o número de blocos.

Nos baseamos apenas nos blocos da categoria “controle” por terem estruturas lógicas essenciais em uma programação, entretanto esses mesmos blocos requerem uma entrada de dados, que serão fornecidos pelo Robô Praxedes. Por isso foi preciso seguir também a 3ª linha da pesquisa paralelamente, construindo os blocos da categoria “controle” observando os blocos que Scratch possui e como o Robô Praxedes pode atendê-los.

Ao fim de construir estes blocos notamos que poderíamos dividir essa categoria de blocos em duas categorias:

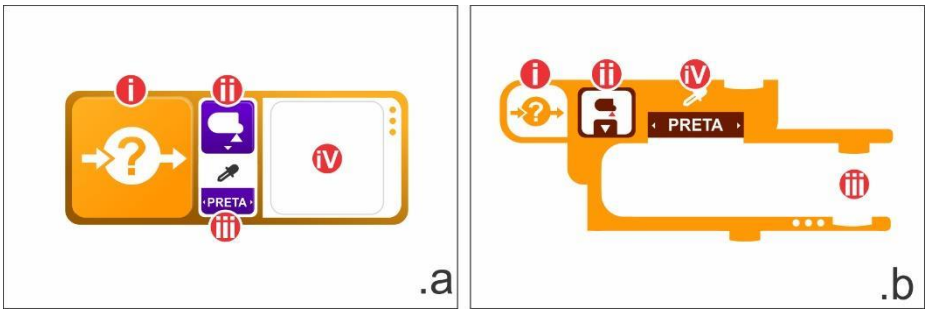


Figura 16 - Exemplo Bloco Condicionais (.a Nível 1 e .b Nível 2)

- Categoria formada pelos blocos lógicos que precisam do uso de condicionais denominado “Condicionais”(Figura 16), que tem a seguinte estrutura:
  I. Representação da Funcionalidade do bloco em questão;
  II. Sensor escolhido para a captura de dados;
  III. Parâmetro de comparação que irá executará a condicional;
  IV. Espaço para encaixar blocos que irão compor a condicional/laço;

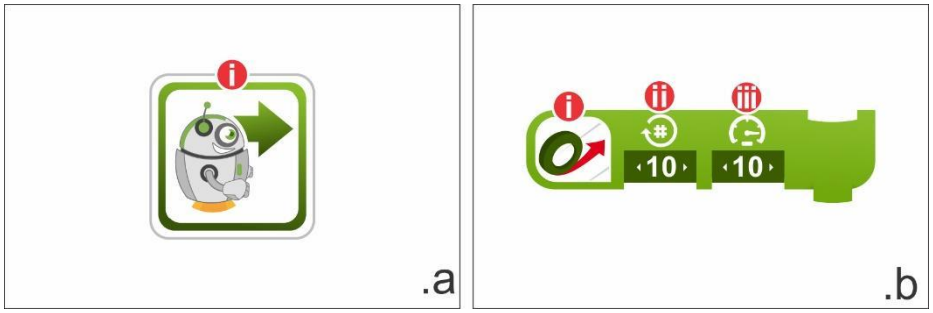


Figura 17 - Exemplo Bloco Controle (.a Nível 1 e .b Nível 2)

- Categoria formada por blocos que não necessitam de condicionais denominada “Controle” (Figura 17), que tem a seguinte estrutura:
  I. Representação da Funcionalidade do bloco em questão;
  II. Número de repetições dessa funcionalidade;
  III. Velocidade dessa funcionalidade;

Os demais blocos das demais categorias foram criados observando apenas a estrutura do Robô Praxedes (Figura 18).

Figura 18 - Representação dos blocos lógicos

5.6 Menu de Blocos Lógicos

Os blocos lógicos estão separados por categorias localizadas na parte esquerda da tela, como a maioria dos softwares desse gênero, pesquisados neste trabalho. Possui cores chamativas e diferentes umas das outras para representar cada categoria visualmente. Além das cores, as categorias possuem um ícone associado para facilitar a memorização, ícone este que é associado diretamente às funções dos blocos que estão contidos na categoria específica. Categorizamos os blocos nos seguintes seguimentos:

- Controles
- Motores
- Sensores
- Leds
- Comentários

Para acessar os blocos de uma determinada categoria basta clicar na aba da mesma e assim poder arrastar os blocos para a área de trabalho. Quando uma categoria de bloco está aberta o usuário só poderá ver as etiquetas de cores das demais categorias. Para poder visualizar as abas com os ícones de todas as categorias basta clicar novamente na aba que está aberta.

Para um melhor uso da área de trabalho, a caixa que contém as abas de categorias de blocos lógicos pode ser ocultada, basta clicar no ícone (olho) localizado na parte superior

direita desta caixa. Para voltar a ser visível basta clicar novamente, desta vez no ícone do olho cortado (Figura 19).

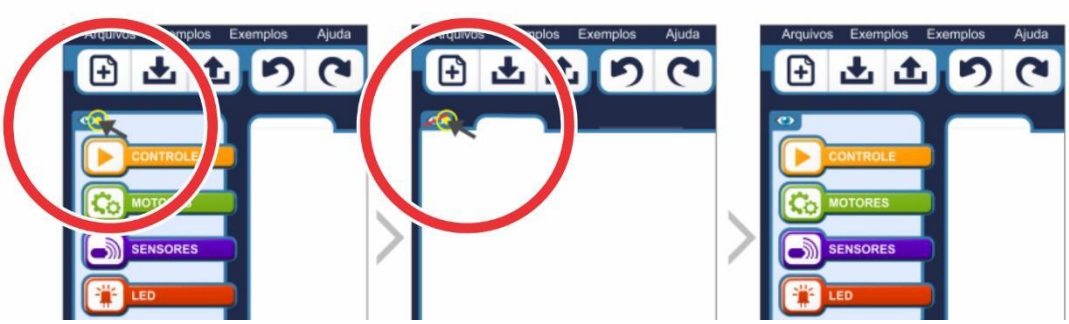


Figura 19 - Ocultar aba Bloco Lógicos

5.7 Simulador Virtual

Com o intuito de atingir um maior número de usuários foi sugerido o uso de um simulador virtual para que o resultado da programação desenvolvida na área de trabalho seja apresentado nele. Sem o simulador virtual, obrigatoriamente, o usuário deverá possuir um robô Praxedes, pois só assim o usuário conseguirá ver o resultado de sua programação na prática.

Mesmo que o custo de possuir um robô Praxedes seja baixo, sua construção requer conhecimento que o usuário do PPV sugerido neste trabalho pode não possuir. Assim seguimos a linha das plataformas que possuem simulador virtual, tais como Scratch, RoboMind e RobotProg. A aba do simulador virtual estará localizada no canto superior direito da interface logo acima da aba destinada ao tutor virtual como mostra a figura 20, ambas as abas poderão ser ocultadas ao clicar no ícone (olho) assim como na aba dos blocos lógicos.

Figura 20 – Simulador Virtual

5.8 Tutor Virtual e Mensagens de erros

Nenhuma das PPVs analisadas neste trabalho possui um tutor virtual. O mais próximo disso são as mensagens de erros encontradas na PPV RoboMind (que são em primeira pessoa) e notamos a importância dessa ferramenta em nossa PPV pois “a figura do tutor se torna muito relevante nos ambientes virtuais de aprendizagem” (Felipe Pacheco, 2015). Juntamos as mensagens de erros em primeira pessoa do RoboMind com nosso tutor virtual, fazendo uma ligação do que acontece na simulação virtual com o tutor, pois neste caso o personagem incluso em nossa simulação virtual é o mesmo de nosso tutor virtual como mostra a Figura 21.

Figura 21 - Tutor Virtual com mensagem de erro em primeira pessoa

5.9 Conexão com o dispositivo

A conexão da PPV com o Robô Praxedes é fundamental para gerar uma experiência da programação desenvolvida com o ambiente físico. Para isso precisamos deixar a interface a mais clara e simples possível, para que assim não demande do usuário o entendimento de operação do hardware para começar a programar o Robô Praxedes.

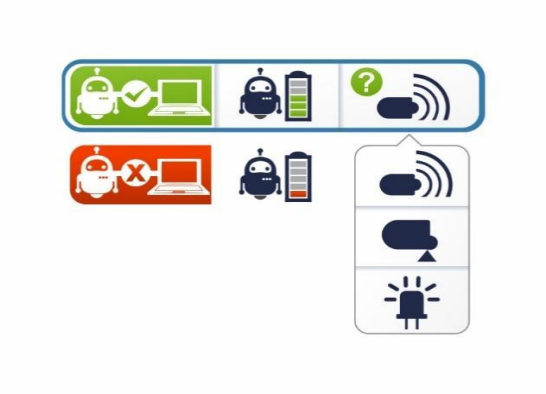


Figura 22 - Representação de Conexão com o robô

Simplificamos a representação de conexão, a exemplo da plataforma EV3, e deixamos esta aba bem visível para que o usuário sempre tenha esta informação a seu alcance, sem necessitar navegar pelos menus. A representação exposta na Figura 22 mostra a situação da conexão com o robô Praxedes: verde para conectado e vermelho para desconectado; além disso na aba conexão mostra o estado da bateria do robô. Por fim a plataforma deverá mostrar todos os sensores conectados ao robô Praxedes.

5.10 Visão Geral

Além de trabalhar cada parte da interface individualmente é importante moldar as partes para que o todo seja harmonioso e agradável de usar, tendo em vista que o público alvo desta plataforma são crianças de 8 a 14 anos de idade. Então era importante que a interface não fosse séria demais ao ponto de não atrair o interesse dos usuários, muito menos deixar muito colorida e infantilizada para não afastar os usuários com idades mais avançadas.

O fundo escuro da plataforma destacando com o fundo claro das abas ajudam os usuários a se localizar mais facilmente na interface, pois desta forma ele pode notar quais partes da interface são mais importantes para o uso da plataforma. Quanto mais claro (quanto maior o contraste) mais importante aquela área é (Figura 20).

O menu superior é composto por gatilhos essenciais para a adaptação rápida a plataforma. Esse foi dividido em 5 partes:

- Arquivos, responsável por salvar, abrir e criar um novo programa; (Figura 23a)

- Desfazer e Refazer, ao iniciar o uso de aplicativo normalmente cometemos erros e nada melhor que uma função que permita o usuário ir e voltar em suas ações; (Figura 23b)
- Nível de complexidade, onde o usuário escolhe o nível de complexidade que deseja trabalhar; (Figura 23c)
- Monitor de conexão com Hardware; (Figura 23d)
- Configurações, onde o usuário poderá voltar para a tela de apresentação e fazer pequenas configurações na interface. (Figura 23e)

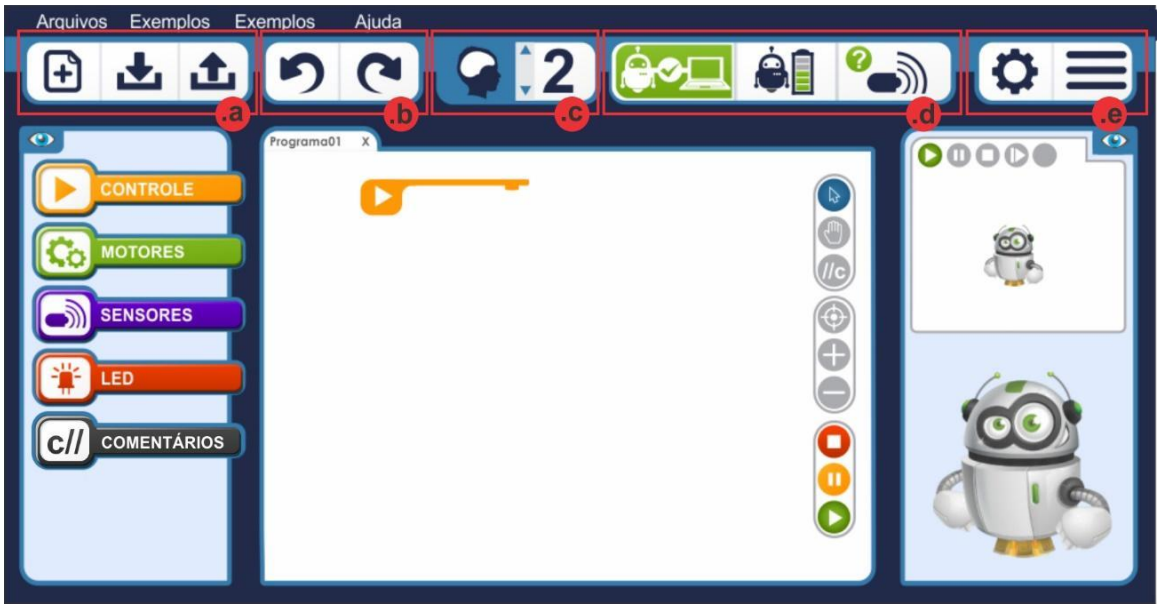


Figura 23 - Interface de PPV sugerida

6. ANÁLISE A PARTIR DE EXECUÇÃO DE ATIVIDADE

Para testar a execução do ambiente desenvolvido e o nível de complexidade ao construir um programa usando os blocos definidos para cada nível executamos a seguinte atividade:

O robô andando para frente até achar um obstáculo (com sensor de toque). Então o robô gira para a direita 90 graus. Se ele girar 360 graus sem sair do lugar, ele liga o LED para avisar que está preso.

Figura 25 - Programa feito nos diferentes níveis

Não conseguimos efetuar essa atividade com as funcionalidades que nossa plataforma possuía, pois se o robô girasse 90 graus para a direita 4 vezes consecutivas ele estaria preso e só assim poderia ligar o LED, mas não possuímos uma forma de contabilizar quantos movimentos repetidos o robô fazia. Para resolver isso tivemos inicialmente a ideia de criar um bloco de variável que seria adicionado ou subtraído uma unidade em seu valor e assim poderíamos efetuar a atividade proposta (Figura 25). Contudo notamos que não seria interessante incluir o bloco de variável para o primeiro nível, tendo em vista que este nível é direcionado para iniciantes e o conceito de variável requer mais entendimento de programação. Já no nível 2 é interessante fazer o uso do bloco variáveis, que irá apenas modificar o valor de uma variável, pois decidimos seguir a linha do *Scratch* no qual possui uma aba que o usuário pode criar uma variável sem precisar inicializar a mesma na área de trabalho (Figura 26).

Figura 26 - Aba Variável

Na programação do nível 1 não ficou claro quais os blocos estão dentro do laço “Para Sempre”, assim como não ficou claro quando termina este laço. Além disso a programação horizontal, com pulos de linha, deixa mais complicado o entendimento do fluxo do programa. Foi preciso desenhar linhas indicadoras para orientar o usuário a direção do fluxo construído por ele (Figura 27).

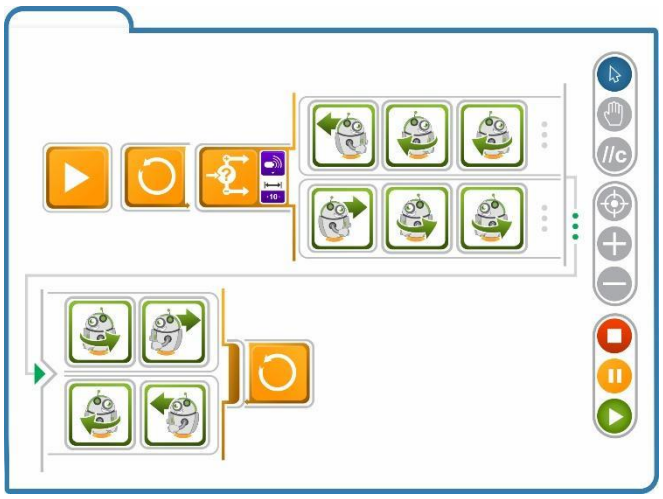


Figura 27 Estrutura de programação Nível 1 aperfeiçoado

Já no nível 2 o ícone “+” e o ícone “-” que representam as condicionais do laço “se” podem levar a outro entendimento, como por exemplo ter mais informações ou menos informações. Portanto o ícone “+” foi substituído pelo número “1” e o ícone “-” foi substituído pelo número “2”.

Outro ponto que também precisou de edição no nível 2 foi nos blocos de condicional quando se faz necessário o uso de uma variável. Aqui preciso que usuário escolha um agente que será encarregado de fazer a comparação, em outras palavras escolher entre maior que, menor que ou igual para fazer o comparativo de condicional (Figura 29).

Figura 3 - Exemplo de bloco condicional com uso de variável

7. CONCLUSÃO

A Robótica educacional é uma estratégia de tentar engajar o aluno a aprender assuntos de várias áreas do conhecimento. Para fazer uso dessa estratégia no ensino fundamental é importante ter em mão uma boa plataforma de programação visual (PPV) para suprir a necessidade do conhecimento de sintaxes de uma determinada linguagem de programação. Hoje a linha Lego possui uma das melhores PPV do mercado, contudo suas plataformas só programam kits produzidos pela própria Lego tornando difícil o acesso a essa interação por questões financeiras. Para superar esse obstáculo muitos pesquisadores desenvolveram novos ambientes de programação e esse trabalho teve o objetivo de tentar reunir todos os pontos positivos da linha Lego como também dos demais softwares.

Com base nos artigos pesquisados e os softwares aqui citados sugerimos uma proposta de interface para uma PPV unindo algumas ferramentas que virão a facilitar o entendimento do software pelo o usuário, assim como deixar a plataforma simples e agradável para conseguir incentivar o aluno a aprender ainda mais sobre diversas áreas do conhecimento. Essas ferramentas foram:

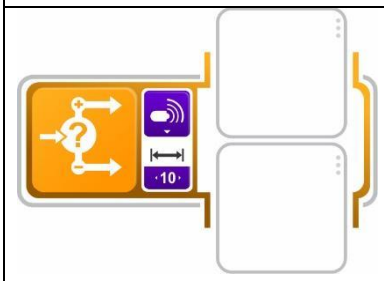
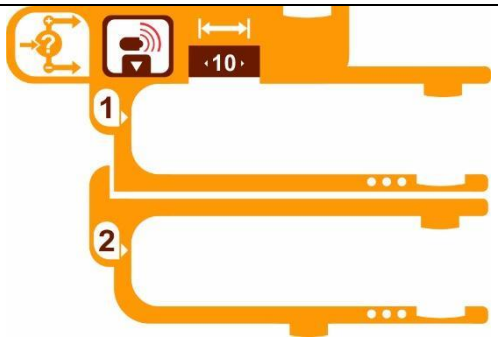
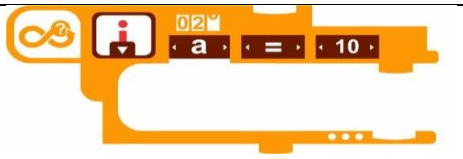
- Tutor Virtual;
- Simulador Virtual;
- Tela de Apresentação;
- Aspecto de Gameficação;
- Diferentes funcionalidades por nível de conhecimento;
- Ícone em categorias de blocos.

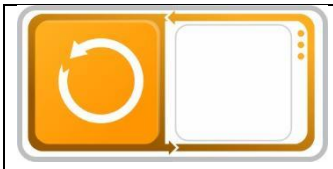
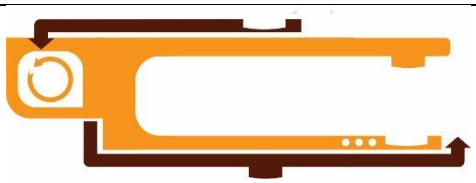
Contudo para que possamos avaliar esta forma de interface é preciso fazer um teste de usabilidade com crianças 8 a 14 anos de idade assim como um teste de compreensão dos ícones utilizado. Também faz necessário, como trabalho futuro, um estudo mais aprofundado sobre quais blocos devem conter na plataforma e categoriza-los seguindo a linha de pesquisa que se baseia em uma linguagem de programação; em outras palavras, é preciso criar os blocos lógicos baseados em uma linguagem de programação estruturada trabalhando paralelamente com a evolução do Robô Praxedes. Além disso será preciso implementar a plataforma para fazer um testar fielmente a usabilidade da plataforma aqui sugerida.

8. ANEXOS

8.1 Blocos Lógicos

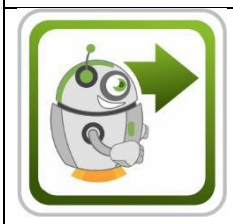
8.1.1 Blocos da categoria Controle

Nível de complexidade 1	Nível de complexidade 2 e 3	Descrição
		**Início:** para iniciar qualquer programação deve se usar este bloco, por isso ele já está inserido na área de trabalho ao abrir uma nova programação por padrão.
		**Se, se não:** Nesta condicional o usuário pode escolher o sensor que lhe dará os valores para fazer uma comparação e assim executar a condicional, o bloco se comporta diferente para os diferentes sensores. Caso a condicional entre no se o software irá ler o primeiro conjunto de blocos lógicos se entrar no "se não" o programa executará o segundo conjunto de blocos.
		**Se:** Ao escolher um sensor o usuário estrutura sua condicional, caso a condicional resulte um valor verdadeiro o programa irá ler o conjunto de blocos inserido dentro desta condicional, caso o valor seja negativo o programa ignorará o conjunto de blocos lógicos inseridos nesta condicional.
		**Sempre se:** Segue a mesma estrutura da condicional "Se", contudo o programa irá repetir o conjunto de blocos lógicos inserido nessa condicional repetidamente sem parar.
		**Repita até:** O conjunto de blocos dentro dessa condicional serão executados repetidamente sem para até que o resultado da condicional for verdadeiro o laço termina
		**Espera até:** Faz com que a programação fique parada até que o valor desta condicional seja verdadeiro.

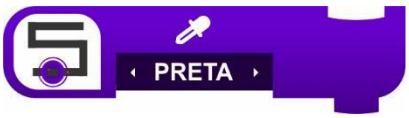
		**Sempre:** O conjunto de blocos irá ser executado repetidamente para sempre.
		**Repita:** O conjunto de blocos irá ser executado "n" vezes, "n" este que é escolhido pelo usuário.
		**Para:** Esse bloco força a parada de um laço ou sinaliza o término da programação.
		**Esperar:** A execução do programa fica parada por "n" segundo, "n" este que é definido pelo usuário.

8.1.2 Blocos da categoria Motores

Nível de complexidade 1	Nível de complexidade 2 e 3	Descrição
		**Girar a Esquerda:** No nível 1 o robô irá girar para a esquerda 90 graus; No nível 2 e 3 o usuário escolhe quantos graus o robô irá girar para a esquerda além de escolher a velocidade que o robô faz esse movimento.
		**Girar a Direita:** No nível 1 o robô irá girar para a direita 90 graus; No nível 2 e 3 o usuário escolhe quantos graus o robô irá girar para a direita além de escolher a velocidade que o robô faz esse movimento.

		**Andar para traz:** No nível 1 o robô irá andar para trás um passo (essa distância equivale a uma rotação inteira de sua roda associada ao motor); No nível 2 e 3 o usuário escolhe quantos passos o robô irá dar para trás além de escolher a velocidade que o robô faz esse movimento.
		**Andar para frente:** No nível 1 o robô irá andar para frente um passo (essa distância equivale a uma rotação inteira de sua roda associada ao motor); No nível 2 e 3 o usuário escolhe quantos passos o robô irá dar para frente além de escolher a velocidade que o robô faz esse movimento.

8.1.3 Blocos da categoria Sensores

Nível de complexidade 1	Nível de complexidade 2 e 3	Descrição
		**Seguir Linha:** O robô irá seguir a linha da cor escolhida pelo usuário, contudo como nosso hardware é limitado o robô só conseguirá distinguir as cores preta e branca.

8.1.4 Blocos da categoria LED

Nível de complexidade 1	Nível de complexidade 2 e 3	Descrição
		**Desligar LED:** O robô irá desligar um LED especifico, LED este escolhido pelo usuário.
		**Ligar LED:** O robô irá ligar um LED especifico, LED este escolhido pelo usuário.

8.1.5 Blocos da categoria Variáveis

Nível de complexidade 2 e 3	Descrição
	**Atribuir:** O usuário poderá atribuir um valor ao uma determinada variável escolhendo-a neste bloco e escolher o valor a ser atribuído a essa variável, variável essa criada na aba variáveis anteriormente a atribuição.
	**Excluir:** O usuário poderá excluir uma determinada variável escolhendo-a neste bloco, variável essa criada na aba variáveis anteriormente a atribuição.
	**Incrementar:** O usuário poderá incrementar um valor ao uma determinada variável escolhendo-a neste bloco e escolher o valor a ser incrementado a essa variável, variável essa criada na aba variáveis anteriormente a atribuição.

8.1.6 Blocos da categoria Comentários

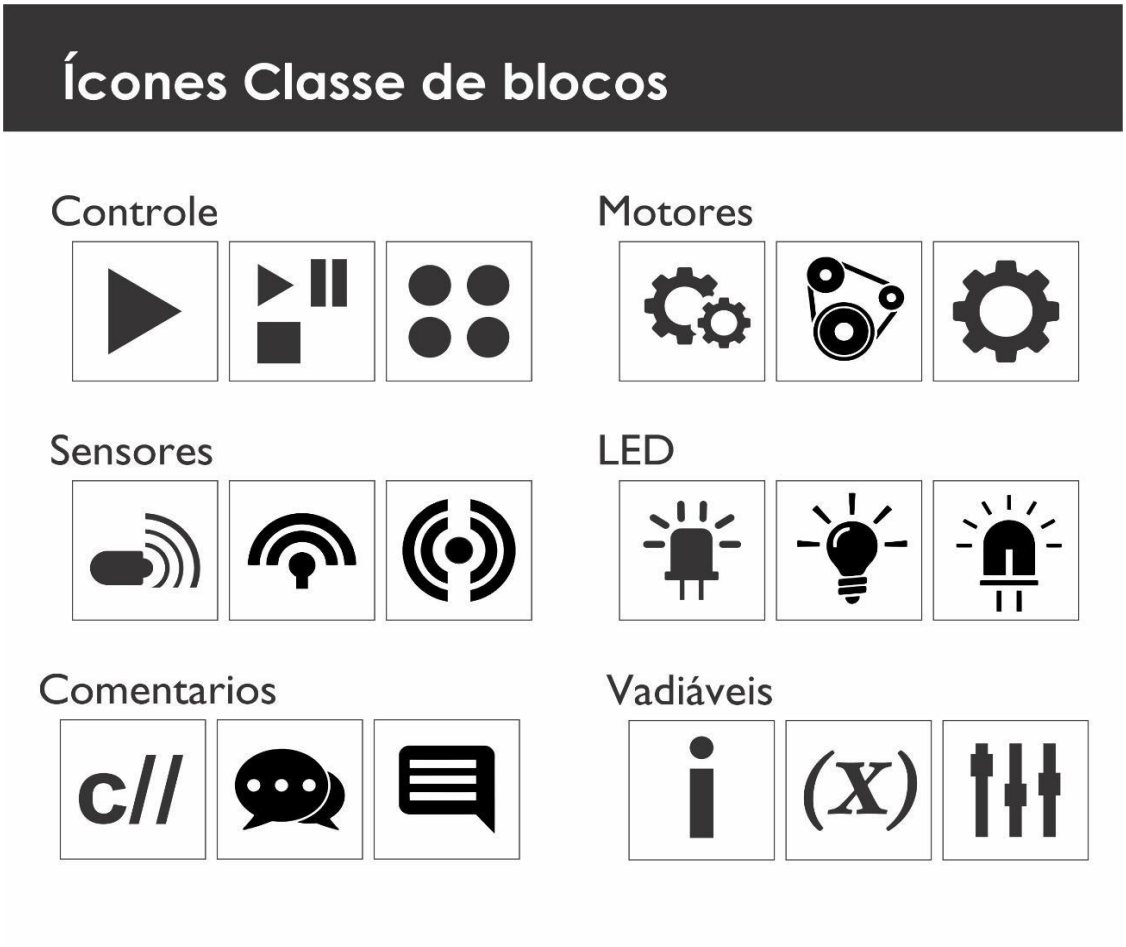
Nível de complexidade 1	Nível de complexidade 2 e 3	Descrição
		**Comentário:** Ao inserir este bloco abrirá uma caixa de texto para usuário escrever seu comentário, esse comentário poderá ser visto ao clicar no bloco inserido na programação.

8.2 Ícones Levantados

8.2.1 Ícones Blocos Motores

8.2.2 Ícones De Atributos de Blocos

8.2.3 Ícones de Classe de Blocos



8.2.4 Ícones dos Blocos Controle

9. REFERÊNCIAS BIBLIOGRÁFICAS

Alecrim, Emerson. *Boost é o novo kit robótico da Lego.* 2016. https://tecnoblog.net/205725/lego-boost-robotica/.

Almeida, Thais Oliveira, José Francisco de Magalhães Netto, Romero Gomes da Silva, e Tiago de Paula Custódio. "Laboratório Remoto de Robótica como Elemento Motivador para a Aprendizagem de Programação." *Anais do XXVIII Simpósio Brasileiro de Informática na Educação (SBIE 2017)*, 2017: 665-674.

Alves, Rafael Machado, Fábio Ferrentini Sampaio, e Marcos da Fonseca Elia. "DuinoBlocks: Desenho e Implementação de um Ambiente de Programação Visual para Robótica Educacional ." *Revista Brasileira de Informática na Educação*, 2014: 126-140.

Aquino, Lygia Matos, Matheus Pires de Farias, Italo de Vasconcelos Crispim, Vandilberto Pereira Pinto, e Francisco Herbert Lima Vasconcelos. "Proposta de um curso semipresencial de robótica educacional utilizando a plataforma Arduino." *Principia*, 2017.

ARDUINO. *Arduino IDE.* s.d. https://www.arduino.cc/en/Main/Software (acesso em 22 de Junho de 2019).

Barros, Renata Pitta. "RoboEduc- Uma ferramenta para programação." *RoboEduc- Uma ferramenta para programação.* Natal, Rio Grande do Norte, 27 de 06 de 2008.

BBC Bitesize. *Introduction to computational thinking.* s.d. https://www.bbc.com/bitesize/guides/zp92mp3/revision/1 (acesso em 20 de 10 de 2018).

BRACKMANN, CHRISTIAN PUHLMANN. "DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL ATRAVÉS DE ATIVIDADES DESPLUGADAS NA EDUCAÇÃO BÁSICA." PORTO ALEGRE, RIO GRANDE DO SUL, 2017.

Costella, Leonardo, Marco Antônio Sandini Trentin, Victor Grando do Amarante, e Adriano Canabarro Teixeira. "Construção de ambiente de ensino de robótica remota: democratizando o desenvolvimento do pensamento computacional em alunos da educação básica." *Anais do XXVIII Simpósio Brasileiro de Informática na Educação (SBIE 2017)*, 2017: 354-363.

Dehouck, Rémi. *The maturity of visual programming.* 29 de 9 de 2015. http://www.craft.ai/blog/the-maturity-of-visual-programming/.

Felipe Pacheco, Paula Cristina Dias Sardinha. "A IMPORTÂNCIA DO TUTOR EM AMBIENTES DE ENSINO--APRENDIZAGEM E FERRAMENTAS DE AVALIAÇÃO EM EAD." *Comunicação & Mercado/UNIGRAN* , 10 de julho de 2015: 142-150.

Formiga, Eliana. *Símbolos Gráficos Métodos de Avaliação de Compreensão.* São Paulo : Blucher, 2012.

Francisco Ioneiton da Silva, Daniel Scherer. “Praxedes: Protótipo de Um Kit Educacional de Robótica.” *Revista EaD & tecnologias digitais na educação*, 2013: 44-56.

Geovania Cezana Araujo Cunha, Luciana Pelissari Barraqui, Sergio Antônio Andrade de Freitas. “Uso da gamificação nos anos iniciais do ensino fundamental brasileiro.” *Anais do XXVIII Simpósio Brasileiro de Informática na Educação*, 2017: 1742-1744.

Gomes, A., Areias, C., Henriques, J., & Mendes, A. J. “Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte.” *Revista Portuguesa de Pedagogia*, 2008: 161-179.

Lego. *EV3 SOFTWARE DOWNLOAD (PC/MAC).* s.d. https://www.lego.com/en-us/mindstorms/downloads/download-software.

Maia, Rafael de Oliveira, Francisco Assis da Silva, Mário Augusto Pazoti, Leandro Luiz de Almeida, e Danillo Roberto Pereira. “DESENVOLVIMENTO DE UM DISPOSITIVO PARA APOIO AO ENSINO DE COMPUTAÇÃO E ROBÓTICA.” *Colloquium Exactarum*, 2014: 71-85.

Maisonnette, Rogers. *a utilização dos recursos informatizados a partir de uma relação.* s.d. http://livrozilla.com/doc/1690955/a-utiliza%C3%A7%C3%A3o-dos-recursos-informatizados-a-partir-de-uma-...

MINISTÉRIO DA EDUCAÇÃO. “Base Nacional Comum.” *BASE NACIONAL COMUM CURRICULAR.* 2018.

O que é o SCRATCH ? s.d. http://programacaoscratchupf.blogspot.com/2012/05/ola-pessoal-bem-vindos-esse-blog.html.

Pasternak, Erik. “Visual Programming Pedagogies and Integrating Current Visual.” Pitsburgo, 19 de 8 de 2009.

Queiroz, Rubens Lacerda, Fábio Ferrentini Sampaio, e Mônica Pereira dos Santos. “DuinoBlocks4Kids: Ensinando conceitos básicos de programação a crianças do Ensino Fundamental I por meio da Robótica Educacional.” *Anais dos Workshops do V Congresso Brasileiro de Informática na Educação (CBIE 2016)*, 2016: 1169-1178.

Resnick, Mitchel. “Scratch: programming for all.” *communications of the acm*, 2009: 60-67.

RobotProg. *Aprenda programação e divirta-se.* s.d. http://www.physicsbox.com/indexrobotprogen.html.

Sobrinha, Vitória Heliane P. S., Gabriela Roberta A. do Nascimento, Ruan Delgado Gomes, e Otacílio de Araújo Ramos Neto. “Plataforma para auxílio ao ensino de programação e robótica pedagógica.” *Principia*, 2016.

UNO. *Studio UNO.* 2012. http://www.robouno.com.br/studiouno/overview.

Welcome to RoboMind.net. s.d. http://www.robomind.net/en/index.html.

Wing, Jeannette M. “Computational Thinking.” *COMMUNICATIONS OF THE ACM*, 2006: 33-35.

—. “SOCIAL ISSUES IN COMPUTING.” *SOCIEDADE DE BENEFÍCIOS DE PENSAMENTO COMPUTACIONAL*. New York: New York: Academic Press, 10 de 01 de 2014.

SEEDF, 2018. *PNAIC-Pacto Nacional para Alfabetização na Idade Certa.* [Online] Available at: http://www.se.df.gov.br/pnaic-pacto-nacional-pela-alfabetizacao-na-idade-certa/
[Acesso em 30 jul. 2019].