Madrugada: Um Ambiente de Robótica Educacional para o Ensino de Programação e Matemática com *Hardware* Livre

Julio C. S. Ferreira[1], **Rafael G. Cerci**[1], **Helio H. L. C. Monte-Alto**[2]

[1]Universidade Federal do Paraná – Palotina – PR – Brasil

{julio.silva, rafael.cerci}@ufpr.br

[2]Universidade Tecnológica Federal do Paraná – Curitiba – PR – Brasil

heliohenrique3@gmail.com

Resumo. *O uso de tecnologias, tais como a robótica educacional, em sala de aula, torna o processo de ensino e aprendizagem mais lúdico, contribuindo para despertar um maior interesse nos estudantes. A ferramenta abordada neste trabalho é um robô que desenha em uma lousa e que implementa a linguagem de programação educacional Logo por meio de blocos similares ao Scratch, sendo de fácil manipulação em sala de aula e de baixo custo para a sua construção, por ser baseado em hardware e software livres. Nessa ferramenta, os alunos escrevem programas, por meio de uma linguagem de blocos, que fazem com que o robô desenhe em uma lousa utilizando conceitos matemáticos, contribuindo para com o desenvolvimento do pensamento computacional e matemático dos alunos. Este trabalho apresenta a montagem do hardware, a arquitetura do software, detalhes de implementação e ideias para trabalhos futuros.*

1. Introdução

Educar é uma tarefa colaborativa entre professores e alunos. O aprendizado, se não acontecer de forma natural e atrativa para o aluno, torna-se árduo para o aluno e cansativo para o professor. O ensino tem sido feito, ao longo de muitos anos, por meio de símbolos, signos e outros instrumentos que estabelecem a relação entre os alunos e os objetos do conhecimento. Pensando em outros meios de apoiar e instigar a aprendizagem, o estudo que levou à concepção do protótipo apresentado neste artigo visa à procura de meios alternativos de ensino com a finalidade de melhorar o desempenho dos alunos e estimular um maior interesse pelas disciplinas de Matemática e Computação.

O presente trabalho apresenta o desenvolvimento de um robô desenhador de lousa, apelidado de Madrugada, que implementa a linguagem de programação educativa Logo, tendo seu projeto físico baseado no robô artístico de código aberto *Makelangelo* [Makelangelo 2019]. O Logo é uma linguagem de programação educacional criada por Seymour Papert [Papert 1972], que tem por objetivo utilizar a programação de computadores para desenvolver o aprendizado da matemática, especialmente a geometria. Esta linguagem, disponível em diversas versões, consiste em um ponteiro, comumente chamada de "tartaruga", para o qual é possível programar comandos para que se mova num espaço bidimensional. Por onde a "tartaruga" passa nesse espaço, ela desenha uma linha, como se estivesse segurando uma caneta ou lápis. Assim, as crianças aprendem a programar desenhando figuras geométricas e até mesmo desenhos mais complexos. A Figura

DOI: 10.5753/cbie.wcbie.2019.1302

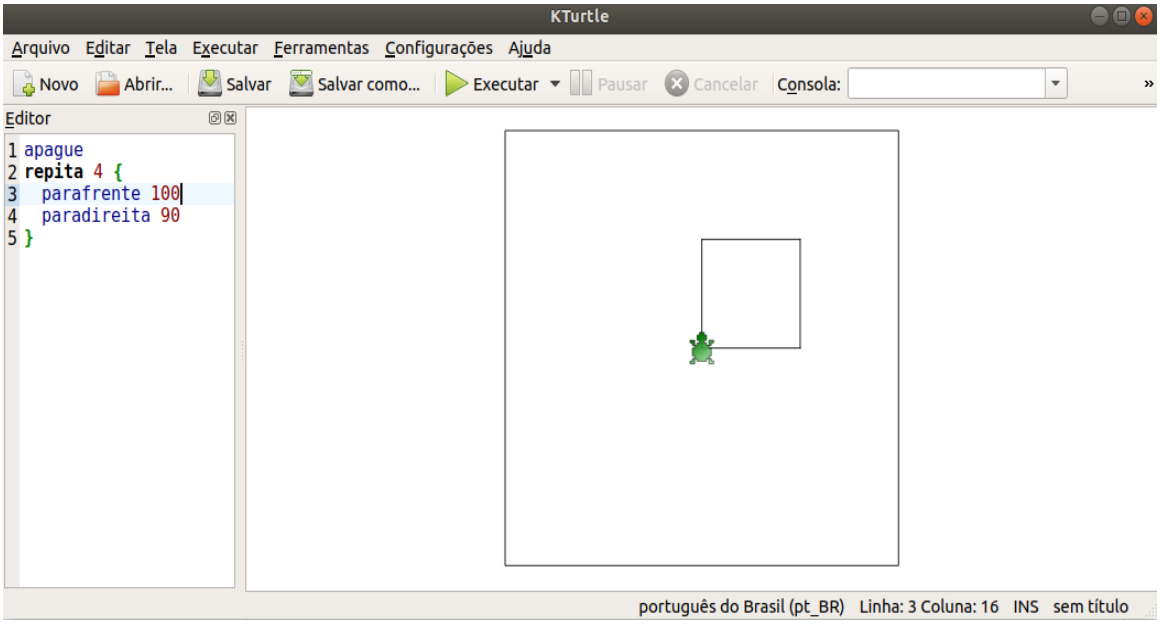

Figura 1. Imagem do *software* KTurtle, uma das muitas implementações da linguagem Logo.

1 mostra um exemplo de um *software* baseado em Logo no qual são inseridos comandos para desenhar um quadrado.

O objetivo deste trabalho é criar uma linguagem que interaja com o robô desenhador de lousa de modo similar à linguagem Logo, instigando e permitindo o aprendizado tanto de Matemática quanto de programação de computadores de forma lúdica. Pretende-se, assim, colocar à disposição do professor uma ferramenta que torne o ensino de Matemática e programação mais significativo para os alunos. Colocando em prática alguns conceitos matemáticos, o aluno poderá visualizar o emprego prático do conteúdo em questão, despertando, assim, um maior interesse pela busca e construção do conhecimento matemático e computacional.

A Seção 2 apresenta a metodologia que foi utilizada neste trabalho. A Seção 3 apresenta a montagem física do robô desenhador de lousa. A Seção 4 apresenta uma descrição da arquitetura de *software*. A Seção 5 apresenta um detalhamento sobre o desenvolvimento do *software* do protótipo. A Seção 6 discorre sobre algumas ideias para próximos passos. Por fim, são apresentadas algumas considerações finais.

2. Metodologia

O desenvolvimento deste trabalho foi dividido nas seguintes etapas: (I) pesquisa sobre projetos similares ao pretendido; (II) montagem do *hardware* do robô utilizando a plataforma Arduino; (III) utilização de *software* fornecido pelos projetos similares encontrados para realização dos testes do *hardware*; (IV) planejamento e projeto da arquitetura do *software* do ambiente Logo; (V) implementação da *firmware* (programa gravado diretamente no *hardware*) para o Arduino; (VI) implementação da linguagem Logo; e (VII) implementação da interface com o usuário.

Tendo encontrado o *Makelangelo* como uma solução de código aberto (*software*

livre) para a movimentação básica de uma lousa robótica, foi iniciada a montagem do robô. Uma vez que o *hardware* ficou pronto e funcional, o próximo passo foi o desenvolvimento do ambiente de *software* que dá apoio à programação da "tartaruga" de modo integrado à lousa.

3. Desenvolvimento do *Hardware*

Primeiramente foi feito um planejamento para a montagem do *hardware*, assim como o estudo do *software* e da plataforma de desenvolvimento do *Makelangelo*. Para alguns dos itens usados para confeccionar o robô foram utilizados materiais reciclados. A seguir são listados os itens básicos utilizados para a confecção do robô:

- Uma plataforma de prototipagem eletrônica Arduino;
- Uma ponte H dupla L298N;
- Módulo *bluetooth* HC-06;
- Um quadro branco de 60 x 90 cm;
- Dois motores de passo Nema 17, usados para a movimentação da caneta e afixados nos cantos superiores da lousa;
- Duas polias GT2, conectadas aos motores de passo;
- 3 metros de correia aberta 5mm, partida em duas partes de 1,5 metros, passadas pelas duas polias;
- Um servomotor, utilizado para encaixar a caneta e realizar o movimento de levantar e abaixar a caneta;
- Uma fonte 5V 2A;
- Um resistor de 470 Ohms;
- Um resistor de 750 Ohms;
- Parafusos M4 (pode ser utilizado fita dupla face);
- Presilhas plásticas;
- Duas garrafas PET de 600 ml, ligadas às pontas das correias, necessárias para dar pesos de sustentação para a caneta;
- Uma peça de suporte para a caneta, que pode ser impressa usando uma impressora 3D. No nosso caso, para a versão final foi utilizada uma tartaruga de plástico perfurada de modo a permitir encaixar a caneta, como pode ser visto na Figura 3.

Alguns itens podem ser trocados por materiais alternativos. A Figura 2 apresenta uma foto da primeira versão do *hardware* montado. A Figura 3 apresenta uma foto do *hardware* atual e um exemplo do *hardware* em funcionamento.

4. Arquitetura do Ambiente de Programação Logo

A Figura 4 apresenta uma visão geral da arquitetura de *software* do ambiente de programação. A arquitetura é dividida em 3 (três) camadas: (1) camada de *firmware*; (2) camada de interpretador da linguagem; e (3) camada de interface com o usuário. Cada uma das camadas deve ter interfaces bem definidas com as demais camadas. A camada de *firmware* deve implementar uma API (*Application Programming Interface*) que define os comandos básicos no nível de *hardware* e os implemente na linguagem de programação da plataforma Arduino. A camada de interpretador deve implementar uma API de comandos de alto nível da linguagem Logo (e.g., o comando `parafrente(x)`, que permite fazer a

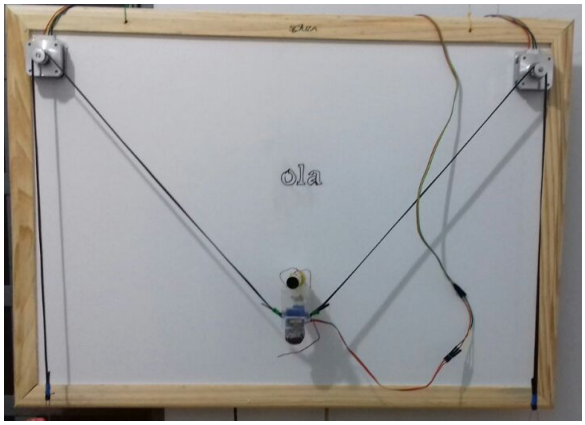

Figura 2. Primeira versão do robô desenhador de lousa montado.

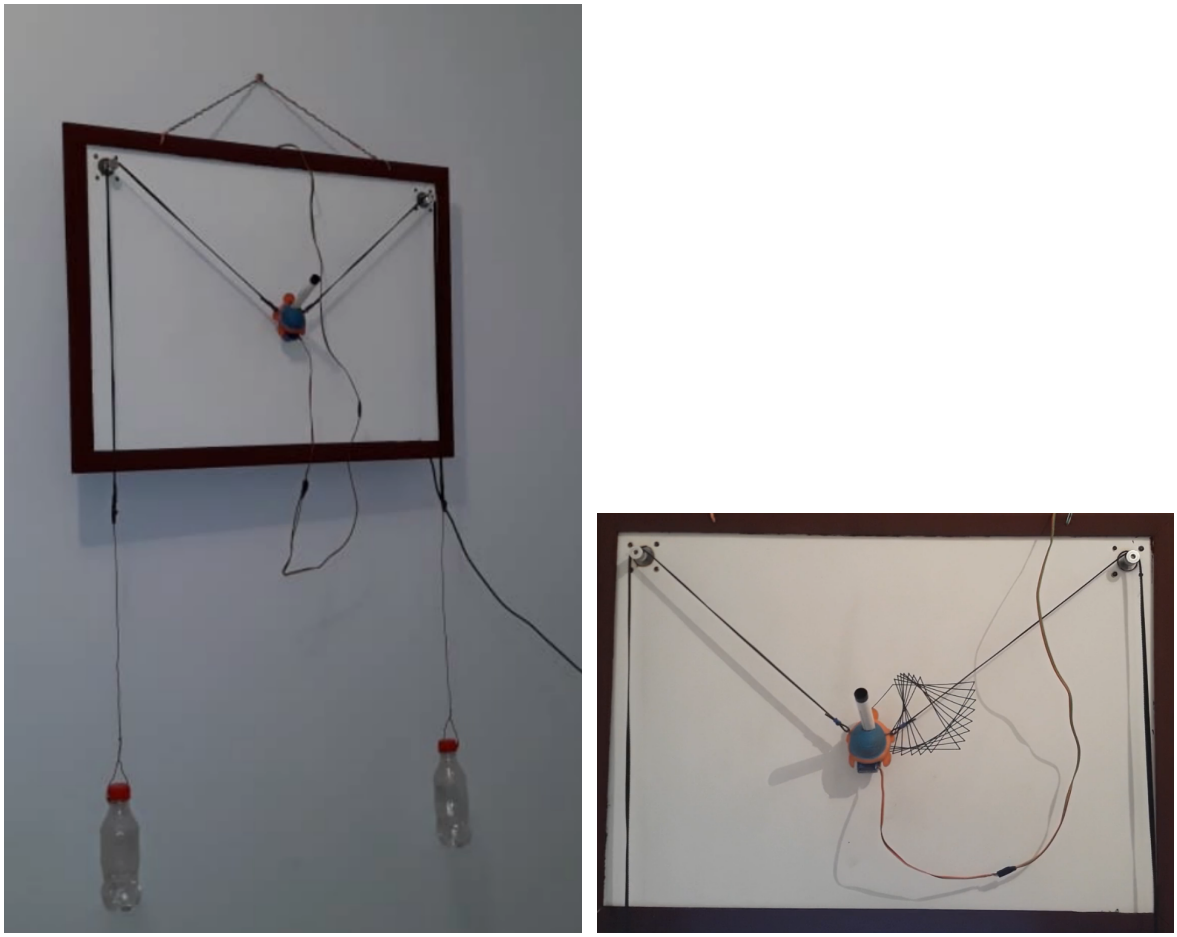

Figura 3. (a) Versão final do robô desenhador de lousa montado; (b) Versão final do robô em execução (ver código da Figura 6).

"tartaruga" andar x passos para frente, e o comando `girar(a)`, que permite fazer a "tartaruga" girar `a` graus). Enfim, a camada de interface com o usuário deve implementar o modo como o usuário inserirá os comandos que a "tartaruga", representada pela caneta na lousa, deverá executar. A fim de tornar a atividade com o uso do Madrugada o mais lúdico

e amigável possível, a interface com o usuário foi implementada como uma linguagem de programação em blocos, similar à linguagem *Scratch* [Maloney et al. 2010]. Também pretende-se criar diferentes versões da interface com o usuário além da que executará em PCs, tais como para *smartphones* e *tablets*, permitindo, assim, uma maior facilidade de acesso, principalmente para uso em sala de aula.

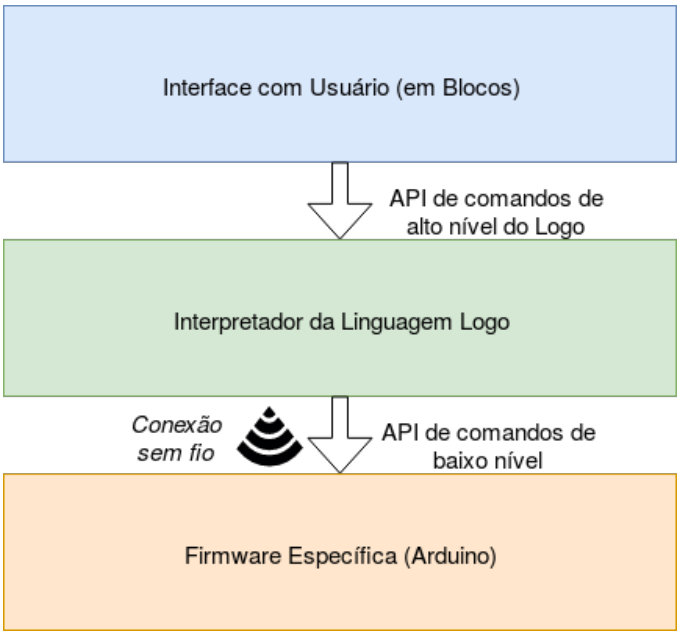

Figura 4. Representação da arquitetura do ambiente de programação.

5. Desenvolvimento do *Software* e Resultados

A fim de se implementar a arquitetura apresentada na seção anterior, foi utilizada a linguagem C++ para a implementação da *firmware*, tirando proveito da biblioteca *Adafruit Motor Shield* (AMS), de código aberto, e tomando como base a *firmware* do Makelangelo que utiliza a AMS. A *firmware* fornece uma interface por meio de comandos recebidos pela porta serial do módulo *bluetooth* ligado ao Arduino. Os comandos disponibilizados são os seguintes:

- `M <DX> <DY>`: move a tartaruga de acordo com o vetor de deslocamento (`DX`, `DY`);
- `G <X> <Y>`: move a tartaruga para a posição (`X`, `Y`);
- `U`: levanta a caneta da tartaruga;
- `D`: abaixa a caneta da tartaruga;
- `H`: move a tartaruga para a posição inicial $(0, 0)$;

Em seguida, a camada de interpretador da linguagem Logo foi implementada na linguagem Python. Neste módulo, foi implementada a classe `Turtle`, que encapsula o estado atual da tartaruga. O estado da tartaruga é composto pelos atributos: `x`, `y`, `angle` e `pen_is_down`, que indicam, respectivamente, a posição (x, y), o ângulo em que a tartaruga está apontando, e um valor *booleano* que indica se a caneta está abaixada ou não. Também foram implementados os métodos de alto nível que manipulam o estado da tartaruga e enviam os comandos apropriados para a *firmware* por meio da interface *bluetooth*. Os seguintes métodos foram implementados:

- `forward(d)`: move a tartaruga `d` passos na direção em que ela está apontando. Calcula os valores `dx` e `dy`, atualiza os valores de `x` e `y` da tartaruga, e envia o comando `M <dx> <dy>`;
- `backward(d)`: o mesmo que `forward(d)`, mas movendo a tartaruga para a direção oposta à que ela está apontando.
- `turn_left(a)` e `turn_right(a)`: atualiza o ângulo atual da tartaruga somando-se/subtraindo-se o ângulo a;
- `pen_up()` e `pen_down()`: levanta/abaixa a caneta da tartaruga, atualizando o atributo `pen_is_down` e enviando o comando apropriado (`U` ou `D`);
- `go_to(x, y)`: move a tartaruga diretamente para a posição (x, y) desejada, atualizando diretamente os atributos `x` e `y` e enviando o comando `G <x> <y>`.

Além da classe `Turtle`, foi criado um módulo interpretador, que recebe um programa escrito em Python contendo chamadas aos métodos de `Turtle` e o interpreta, resultando em uma sequência de instruções que manipulam a tartaruga.

O próximo passo foi a criação de uma interface gráfica que permite criar programas que manipulam a tartaruga dinamicamente. Para isso, foi implementada uma interface *Web* utilizando o Blockly[1], um *framework* de desenvolvimento de linguagens de programação em blocos criado pelo Google, e que vem sendo utilizado em diversos projetos, tais como o MIT App Inventor[2] e o Scratch 3[3]. Ele é totalmente baseado em HTML5 e Javascript, e permite a tradução de código (transpilação) do código em blocos para várias linguagens de programação tradicionais, tais como Python, Javascript, Lua, PHP e Dart. Assim, foram criados os blocos correspondentes aos métodos de alto nível que manipulam a tartaruga, que são então convertidos para instruções na linguagem Python de modo a serem executadas pelo módulo interpretador. A Figura 5 mostra o painel de blocos que podem ser usados. Além dos blocos personalizados criados para a tartaruga, o Blockly fornece blocos genéricos que se traduzem em construções utilizadas em programação de computadores, tais como instruções condicionais (`se... então... senão...`), instruções de *loop* (e.g., `repita <n> vezes...`), além de instruções matemáticas, de manipulação de *strings* (cadeias de caracteres), de definição de funções, entre outras.

Também foi utilizado o *framework* Flask para implementar o servidor *Web* da aplicação, responsável por receber os comandos da interface gráfica e submetê-las ao interpretador da linguagem. Embora sejam utilizadas tecnologias *Web*, a fim de permitir a compatibilidade com o Blockly, a aplicação é executada localmente em um *notebook* ou computador de mesa.

Uma visão geral da interface gráfica é apresentada na Figura 6. Na parte inferior da interface, encontra-se um botão para carregar a lista de todos os dispositivos *bluetooth* próximos, bastando, então, escolher o dispositivo correspondente à lousa do Madrugada para começar a utilizar o sistema. A parte principal da interface é composta pelo painel à esquerda, onde é feita a programação por meio de blocos, conforme mostrado na Figura 5, e pelo painel à direita, que mostra o código em Python gerado a partir dos blocos, permitindo ao usuário conhecer o código que será realmente executado pelo interpretador. Esse código não é editável, sendo disponibilizado unicamente com o objetivo de que o usuário

[1] `https://developers.google.com/blockly/`
[2] `https://appinventor.mit.edu/explore/`
[3] `https://scratch.mit.edu`

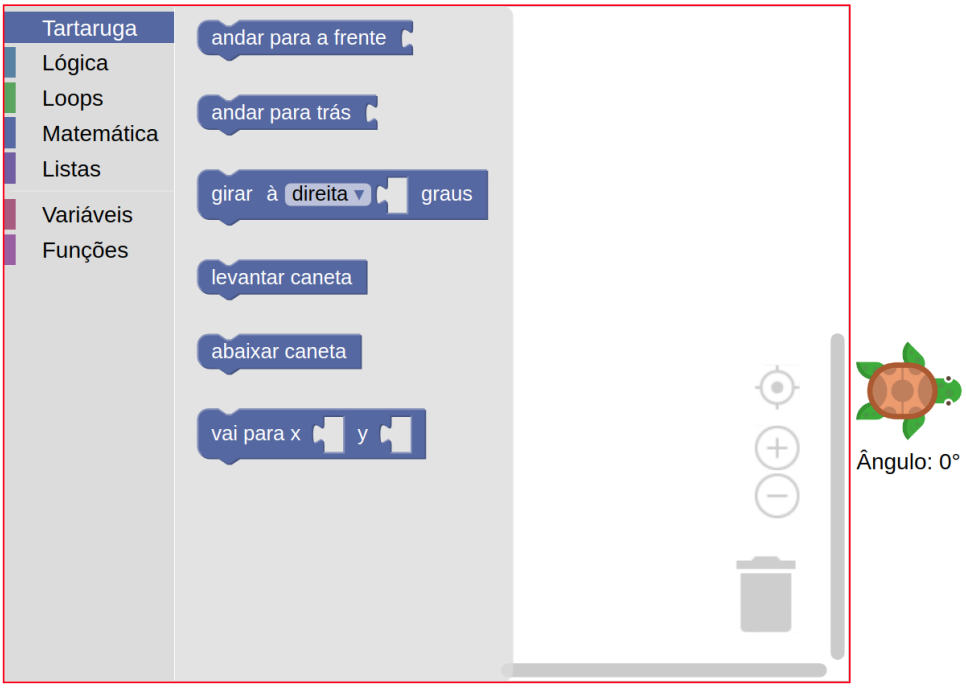

Figura 5. Painel de blocos do Madrugada.

possa aprender um pouco da linguagem de programação textual que realmente é executada. Entre os dois painéis, há a imagem de uma tartaruga e o ângulo para o qual ela está apontando. Esta imagem muda de acordo com o ângulo em que está apontando, de modo a deixar claro para o usuário a direção em que ela se encontra. O ângulo 0º corresponde à tartaruga apontando para o lado direito. Acima do painel esquerdo, encontra-se o botão *Executar* que envia os comandos para o módulo interpretador. Acima do botão *Executar*, encontram-se botões que permitem salvar e carregar programas em um formato XML específico aceito pelo *framework* Blockly, de modo a permitir que os usuários salvem e compartilhem seus programas.

O *software* do Madrugada está licenciado sob a *GNU General Public License* (GNU GPL), Versão 3, sendo permitida sua cópia e distribuição livremente. O código-fonte está disponibilizado no GitHub[4].

6. Trabalhos Futuros

O objetivo do protótipo foi demonstrar a possibilidade da criação de um produto que pode ser utilizado no ensino de programação e matemática para alunos da Educação Básica, utilizando para isso apenas materiais de baixo custo, e *hardware* e *software* livres. Futuramente, pretende-se estender o protótipo por meio da implementação de uma interface para dispositivos móveis, tais como *smartphones* e *tablets*, o que permitirá uma facilidade ainda maior de aplicação em ambiente escolar. Outra ideia é estender a aplicação de modo a permitir o uso de outros tipos de robôs capazes de desenhar, tais como carrinhos com uma caneta acoplada a um servomotor. Isso é possível graças à arquitetura modular do Madrugada, que permite utilizar as mesmas camadas de *software*, com exceção da *firmware*, para diferentes construções de *hardware*.

[4]https://github.com/helioh2/madrugada-robot

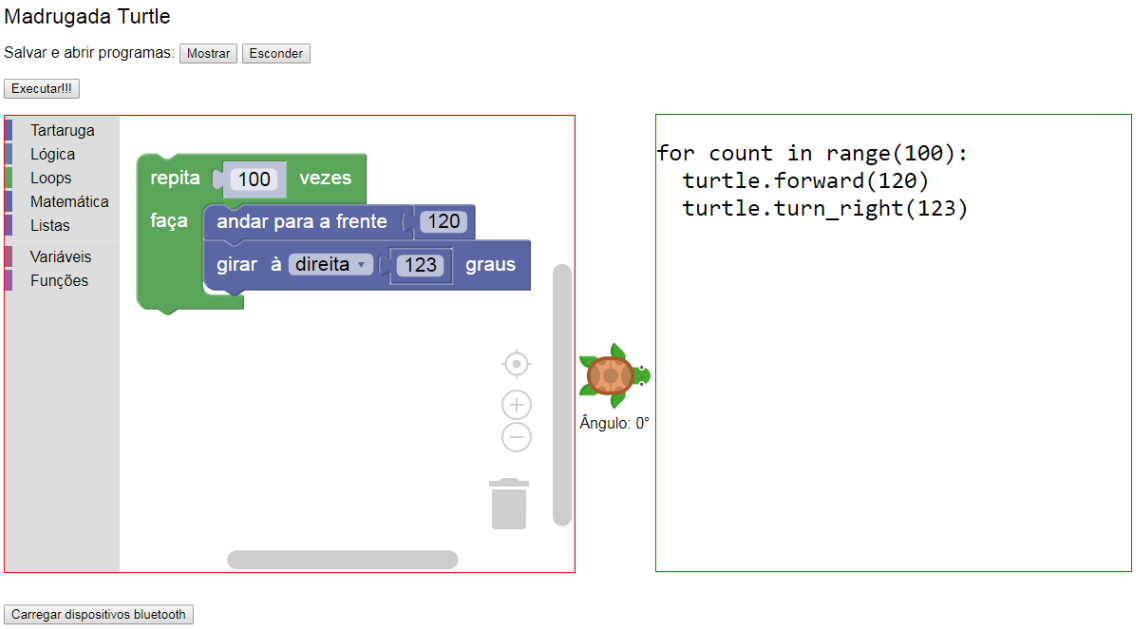

Figura 6. Interface gráfica do Madrugada.

Considerações Finais

O desenvolvimento deste protótipo foi um primeiro passo no sentido de criar uma ferramenta integrada de *software* e *hardware* que utiliza o conceito de robótica educacional no ensino de programação, em especial na Educação Básica. Uma vez que trata-se de um robô que desenha em uma lousa, um objeto muito familiar aos alunos, cremos que será possível realizar atividades lúdicas e motivadoras que despertem o interesse dos alunos e os instiguem à construção do conhecimento, conforme o propósito original da linguagem Logo. Além disso, o robô baseado no *Makelangelo* é uma excelente ideia a ser desenvolvida em projetos do movimento *maker*, tais como *hackerspaces*, podendo ser replicado facilmente com componentes de baixo custo.

Agradecimentos

Agradecemos à PROGRAD - COPEFOR da Universidade Federal do Paraná pelo apoio financeiro recebido por meio do programa Licenciar.

Referências

Makelangelo (2019). The makelangelo art robot: What is a makelangelo. Disponível em `http://www.makelangelo.com/`. Último acesso em 06 de julho de 2019.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E. (2010). The scratch programming language and environment. *Trans. Comput. Educ.*, 10(4):16:1–16:15.

Papert, S. (1972). Teaching children to be mathematicians versus teaching about mathematics. *International Journal of Mathematical Education in Science and Technology*, 3(3):249–262.