

Módulo de programação baseada em blocos para a Plataforma Jabuti Edu com Blockly

Enoque Calvino Melo
Alves

Universidade Federal do
Oeste do Pará (UFOPA)
Santarém – PA, Brasil
enoque@gmail.com

Luan Ferreira Cabral

Universidade Federal do
Oeste do Pará (UFOPA)
Santarém – PA, Brasil
luanfc@gmail.com

Socorro Vânia
Lourenço Alves

Universidade Federal do
Oeste do Pará (UFOPA)
Santarém – PA, Brasil
socorro.vania@gmail.com

Raimundo Augusto
Rego Rodrigues Júnior

Universidade Federal do
Oeste do Pará (UFOPA)
Santarém – PA, Brasil
ray.augustoo@gmail.com

ABSTRACT

This work presents the Jabuti Edu Platform, a brazilian educational and robotic programming project for children and young people, and the development of a new block-based programming module using the Blockly Library. The adaptation of Blockly to Jabuti Edu was designed to make it a more attractive and interactive programming teaching tool, and to allow the children to program using a plug-in blocks visual tool and follow the results of their programs through the robot reactions.

RESUMO

Este trabalho apresenta a Plataforma Jabuti Edu, um projeto educacional brasileiro de programação e robótica para crianças e jovens e a criação de um novo Módulo de programação baseada em blocos através da Biblioteca Blockly. A adaptação da Blockly à Jabuti Edu teve a finalidade de torná-la um instrumento de ensino de programação mais lúdico e interativo, e permitir que as crianças programem utilizando uma ferramenta visual com blocos encaixáveis e acompanhem o resultado de seus programas através das reações do robô.

Categories and Subject Descriptors

K.3.1 [Computers and Education]: Computer Uses in Education

General Terms

Design, Languages.

Keywords

Educational robotics, Technology, Programming .

1. INTRODUÇÃO

A plataforma Jabuti Edu é uma iniciativa de um projeto brasileiro de robótica livre, baseada na teoria Construcionista de Papert [1], que tem como objetivo ensinar robótica e programação para crianças e jovens. Porém uma das inovações dos últimos anos, a programação baseada em blocos, ainda não havia sido implementada na plataforma.

A programação baseada em blocos é uma nova tendência na área de programação de computadores que surgiu com o objetivo de facilitar o aprendizado. O processo de desenvolvimento baseado em blocos permite que usuários programem sem precisar decorar comandos de uma linguagem de programação, ou preocupar-se com erros de sintaxe, o que o torna um modelo ideal para crianças ou pessoas que estão aprendendo a programar.

O conceito de programação baseada em blocos está se tornando bastante popular e algumas ferramentas têm sido desenvolvidas utilizando esse conceito, como por exemplo, a Blockly [2], uma

biblioteca que permite gerar editores visuais para programação em diferentes linguagens.

Este trabalho descreve como a ferramenta Blockly foi utilizada para acrescentar um ambiente de programação visual baseado em blocos dentro de um robô educacional chamado Jabuti Edu. A adaptação da Blockly à Jabuti Edu teve a finalidade de torná-la um instrumento de ensino de programação mais lúdico e interativo, proporcionando à criança a possibilidade de programar sem precisar escrever nenhuma linha de código, apenas clicando e arrastando blocos de comandos encaixáveis que representam instruções, e acompanhar a execução do programa através das reações do robô.

Para atingir o objetivo descrito, abordam-se, na seção 2 o referencial teórico que nortearam o desenvolvimento deste trabalho. Na seção 3 é descrita a plataforma educacional Jabuti Edu. Na seção 4 é apresentada a adaptação e a criação do módulo Blockly. Na seção 5 é descrita as etapas de integração do código da biblioteca com o código já existente da camada de apresentação da Jabuti Edu. Finalmente na seção 6 são tecidas algumas considerações finais.

2. REFERENCIAL TEÓRICO

2.1 Papert, o Construcionismo e a Linguagem LOGO

O uso da tecnologia como instrumento educacional advém de princípios Construcionistas de aprendizagem, onde o indivíduo constrói seu próprio conhecimento através de uma ferramenta [1]. O Construcionismo é uma teoria desenvolvida pelo educador Seymour Papert, baseada na teoria do Construtivismo Cognitivo de Jean Piaget, com quem trabalhou na Universidade de Genebra.

Em sua proposta Construcionista, Papert dá ênfase à criação de ambientes ativos de aprendizagem que permitam aos alunos comprovar, testar e executar suas ideias. Neste sentido, Papert via nos computadores e suas possibilidades um recurso valioso para a criação desses ambientes, instrumentos capazes de potencializar a aprendizagem e a criatividade das crianças.

Em busca da criação de ambientes de aprendizagem que complementasse sua teoria, na década de 60, Papert liderou um grupo de pesquisadores do *Massachusetts Institute of Technology* (MIT-USA) para criar a linguagem de programação LOGO. LOGO é uma linguagem de programação que possibilita a criança dar instruções/comandos ao computador para que ele execute as ações determinadas por ela. A criança coloca o conhecimento no computador e indica as operações que devem ser executadas para produzir as respostas desejadas. Assim, o processo de

aprendizagem ocorre por meio da realização de uma ação concreta, que resulta em um produto palpável.

Até os dias atuais, LOGO é utilizada como ferramenta de apoio ao ensino regular e por aprendizes em programação de computadores. O ambiente LOGO tradicional envolve uma tartaruga gráfica, um robô pronto para responder aos comandos do usuário. No entanto, o primeiro robô controlado pela LOGO possuía um formato hemisférico e movimentava-se lentamente pelo chão, dando pequenos passos [3].

A maioria dos comandos, pelo menos nas versões mais antigas de LOGO, refere-se a desenhar e pintar. Porém, em suas versões mais atuais são mais abrangentes, trabalhando com textos, fórmulas, Inteligência Artificial e controle das portas paralelas do computador.

2.2 Robótica educacional

O avanço das versões da linguagem LOGO, a popularização de brinquedos programáveis, como a linha LEGO Mindstorms [4], e o surgimento de computadores de placa única, como Arduino [5] e Raspberry Pi [6], possibilitaram o desenvolvimento de muitos projetos de modelos robóticos com o intuito de auxiliar o ensino de diversas áreas.

O crescimento constante da robótica e a compreensão de diversas áreas de conhecimento ligadas à ela, a tornaram um notável campo a ser explorado no quesito educacional. Seu uso é considerado uma forma moderna de aplicar a teoria de Papert em sala de aula, onde ora é tratada como metodologia de ensino, ora como um objeto de aprendizagem.

Segundo Saygin *et al.* [7], a robótica educacional tem como objetivo fomentar no aluno a investigação e a materialização dos conceitos aprendidos no conteúdo curricular, por meio da montagem de sistemas constituídos por modelos. Isso, para o autor, proporciona um aprendizado prático que desenvolve no aluno a capacidade de discutir e apresentar soluções na resolução de problemas que lhes são propostos.

A robótica permite interagir com o concreto (robô) e o abstrato (programa) em um mesmo projeto, proporcionando a oportunidade de o aluno observar a ação (movimento do robô) de seu raciocínio executado em um artefato físico [8].

2.3 A programação para crianças e jovens e o modelo de programação em blocos

Nos últimos anos, ocorreram muitas iniciativas de introduzir os conceitos de programação para crianças e adolescentes com o objetivo de estimular o raciocínio lógico e aprendizagem, principalmente em escolas. Conforme citado anteriormente, até hoje LOGO é muito utilizada por instituições de ensino, porém, para dar suporte a essas ações, novas linguagens e ambientes de programação foram desenvolvidos como a Squeak Etoys [9] e Alice [10], porém, nenhum com conceitos mais simples para que possam ser usados em um ambiente de ensino especialmente voltado para crianças ou iniciantes em programação.

Os esforços das pesquisas desenvolvidas pelo MIT em desenvolvimento de linguagens e ambientes de programação e aprendizagem para crianças e jovens deram origem ao Scratch [11], uma nova linguagem de programação inspirada na linguagem LOGO. O Scratch baseia-se no modelo de programação baseado em blocos, ou seja, permite que crianças programem aplicações interativas apenas juntando blocos gráficos

semelhantes à versão digital do Lego, sem pontuações e sintaxes das linguagens tradicionais.

Conforme observado na **Figura 1**, as linguagens de programação em blocos caracterizam-se pelo uso de blocos de construção (*building blocks*) que podem ser encaixados, formando pilhas ordenadas (*stacks*). A linguagem impede os usuários de cometerem erros de sintaxe, devido aos blocos serem programados para encaixar-se somente onde houver sentido sintaticamente.

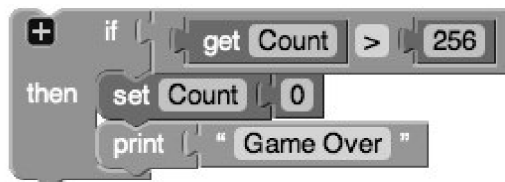


Figura 1 – Estrutura de blocos de comandos

Inspirados no modelo de blocos da linguagem Scratch, surgiram outras linguagens de programação visual e interfaces que facilitam e dão suporte ao desenvolvimento de programas seguindo o paradigma de blocos programáveis, tais como o ArduBlock [12], App Inventor [13], Swift Playgrounds [14] e Blockly.

Neste trabalho será utilizado a linguagem de programação Blockly, uma tecnologia desenvolvida pelo Google. Todo o código do Google Blockly é gratuito e open source, assim o desenvolvedor pode manipular os dados da forma que achar necessário em seus próprios projetos. Blockly será descrita com mais detalhes na seção 4 deste trabalho.

3. PLATAFORMA EDUCACIONAL JABUTI EDU

Jabuti Edu é uma iniciativa de robótica livre que pretende disponibilizar uma Plataforma para construção de robôs educacionais baseada em projetos de software e hardware abertos. Atualmente conta com um robô chamado também de Jabuti Edu, baseado no computador Raspberry Pi, e a Jabuti Edu Mini, que é um robô baseado na plataforma de prototipagem Arduino.

Inicialmente idealizada por Eloir José Rockenbach, hoje o Projeto Educacional conta com uma comunidade atuante que congrega pessoas de várias regiões do país com o objetivo de compartilhar informações, alinhar os esforços de desenvolvimento e garantir a representatividade e a aplicação da tecnologia em nível nacional.

A Comunidade Jabuti Edu (jabutiedu.org) está organizada em Núcleos, que congregam pessoas próximas geograficamente e que são responsáveis pelas ações do projeto em sua Região.

Este trabalho foi desenvolvido pelo Núcleo do Pará, com o apoio do Laboratório Mídias Eletrônicas, do Instituto de Engenharia e Geociências da Universidade Federal do Oeste do Pará (UFOPA). O Núcleo do Pará está sediado na cidade de Santarém e é coordenado pelo Professor Enoque Alves.

Trataremos aqui somente da Plataforma Jabuti Edu (Figura 2), produzida em uma impressora 3D (versão de plástico) ou em uma fresa CNC (versão MDF), embarca um microcomputador Raspberry Pi e disponibiliza uma interface WEB de programação e administração através de uma rede WIFI gerada pelo próprio robô.

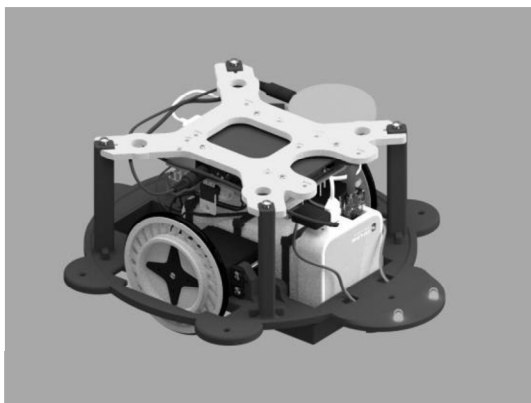


Figura 2 - Jabuti Edu

Desenvolvida usando tecnologias livres, todo seu código está licenciado sob a AGPL e o Hardware sobre a Licença de Hardware Aberto do CERN V1.2L, o que permite que qualquer pessoa possa reproduzi-la.

3.1 Componentes

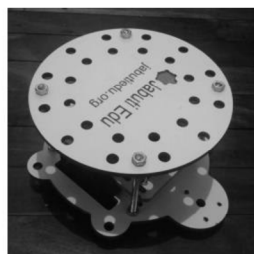
Na construção da Jabuti Edu vários componentes são utilizados em sua montagem. A seguir listamos os componentes mais comuns hoje utilizados pelos membros da comunidade Jabuti Edu. Em alguns deles fazemos referência ao fabricante e modelo da peça, mas não existe nenhuma relação comercial com a marca citada e nem a obrigatoriedade de utilizar a mesma, qualquer produto similar pode ser utilizado atualmente e no futuro na montagem da Jabuti Edu.

3.1.1 Chassi

Desenvolvido originalmente por Diego Henrique, o chassi da Jabuti Edu apresenta atualmente duas versões oficiais (Figura 3): A primeira, em plástico (PLA ou ABS), impressa em uma impressora 3D e a segunda em MDF, que pode ser construída com o uso de uma máquina de corte à laser, fresadora CNC ou até mesmo com ferramentas manuais como serra tico-tico e furadeira.



Versão em plástico



Versão em MDF

Figura 3 - Chassi da Jabuti Edu em Plástico e MDF

Atualmente a comunidade tem trabalhado em uma versão MDF encaixável, que não leva nenhum parafuso em sua montagem, permitindo que a mesma seja facilmente montada por qualquer pessoa, inclusive pelas crianças.

3.1.2 Raspberry Pi

O cérebro da Jabuti Edu é um computador Raspberry Pi® (Figura 4), rodando o sistema operacional Raspbian (uma adaptação do Linux Debian para o Raspberry). Por se tratar de um computador completo, a Jabuti Edu pode ser usada como uma estação de trabalho, bastando para isso conectar um teclado, mouse e um monitor HDMI.

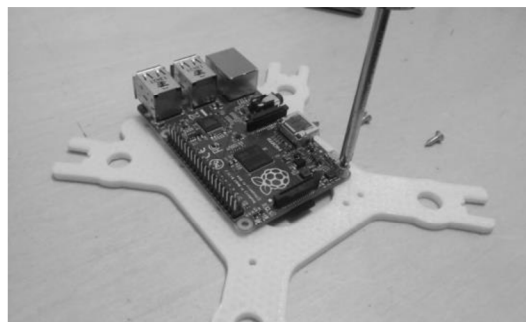


Figura 4 - Raspberry Pi

3.1.3 Roteador WIFI

O acesso remoto à Jabuti Edu é feito através de uma rede wifi gerada pelo próprio robô. Inicialmente usava-se um adaptador de rede USB para gerar uma rede ad hoc, mas por questões de performance o mesmo foi substituído por um mini roteador. Atualmente utiliza-se o roteador TP-LINK TL-wr702n (Figura 5).



Figura 5 - Mini roteador atualmente utilizado na Jabuti Edu

3.1.4 Shield

A Shield da Jabuti Edu (Figura 6) é uma placa de expansão desenvolvida especialmente para prover uma interface entre os pinos de entrada e saída do Raspberry e o hardware do robô. Esta placa possui toda a eletrônica necessária para controlar os motores, leds e demais componentes.

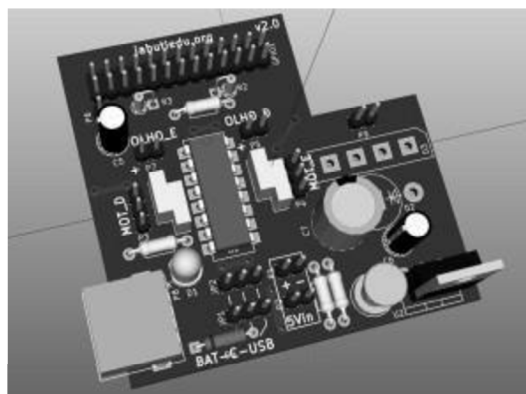


Figura 6 - Nova Shield Jabuti Edu

3.1.5 Servos motores

Os servos motores (Figura 7) são responsáveis pela locomoção da Jabuti Edu, atualmente utiliza-se servos (padrão Futaba s3003) modificados para rotação contínua. Então é necessário a alteração do motor original, retirando-se uma trava, permitindo que o mesmo possa fazer giros de 360 graus.

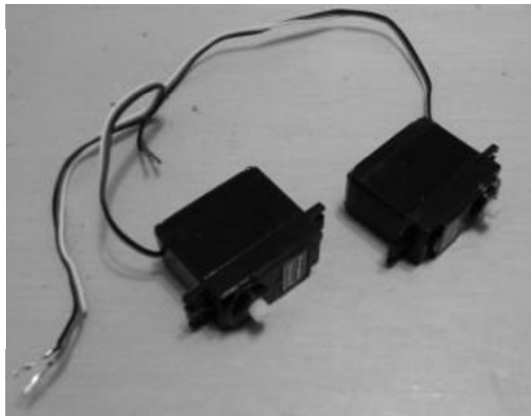


Figura 7 - Servos Motores Futaba s3003

3.1.6 Caixa de som

A Jabuti Edu tem a capacidade de falar, para isso utiliza-se uma mini caixa de som simples (Figura 8), facilmente encontrada em lojas como acessório para celular. O truque é fazer o robô falar, transformando texto em voz, através de um software TTS (Text To Speech) muito comum em sistemas Linux.



Figura 8 - Mini caixa de Som

3.1.7 Bateria

Como fonte de alimentação para a Jabuti Edu utiliza-se como bateria, carregadores portáteis para celulares, também conhecidos como Power Packs. Quanto maior a amperagem fornecida pelo carregador melhor. Atualmente utiliza-se o carregador portátil TP-LINK TL-PB10400 com duas portas USB com corrente máxima de 1A e 2A respectivamente (Figura 9).



Figura 9 - Carregador portátil TL-PB10400

3.2 Arquitetura

A Arquitetura da Plataforma Jabuti Edu é organizada em camadas (Figura 10) que são responsáveis por funções distintas. Esta separação visa divisão de responsabilidades, a independência

entre o software e o hardware do robô e a facilitação da manutenção do sistema.

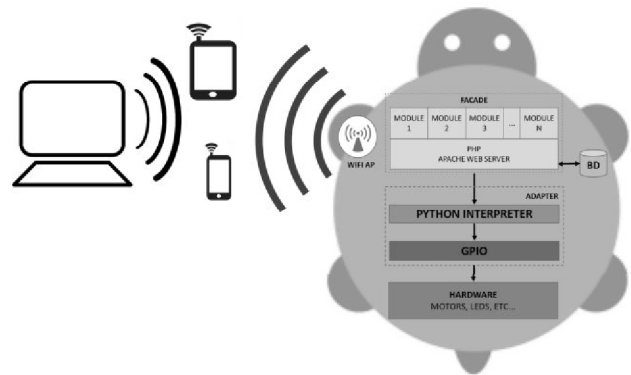


Figura 10 - Arquitetura do Jabuti Edu

A seguir serão apresentadas de forma resumida todas as camadas que compõem a arquitetura do robô.

3.2.1 Camada de Apresentação

Esta camada é responsável pela interação entre os usuários e o robô, contém todos os componentes utilizados para programar e controlar a Jabuti Edu. Envolve tanto as interfaces propriamente ditas (como objetos gráficos responsáveis por receber dados e comandos do usuário) quanto o controle da interação, cadastro e autenticação de usuários, administração do sistema, etc.

A Camada de apresentação foi programada em PHP, rodando sob um servidor web. Assim, o usuário pode acessar a Jabuti Edu de qualquer dispositivo, seja ele um computador, notebook, tablet ou celular, bastando para isso a utilização de um navegador web padrão. Um banco de dados é utilizado para armazenar os dados de cadastro dos usuários do sistema.

Toda a interação com o usuário foi implementada em módulos, onde cada módulo oferece uma forma de controlar ou programar o robô. Por exemplo, no módulo 1 (Figura 11) as crianças podem controlar a Jabuti Edu diretamente pressionando botões de comandos gráficos, neste módulo não há o uso de comandos escritos.

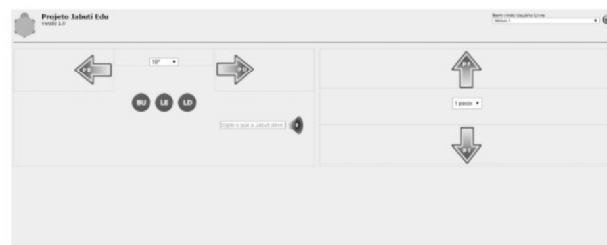


Figura 11 - Módulo 1 do Jabuti Edu

A partir do módulo 2 a programação é orientada à código, o Jabuti Edu permite que vários usuários programem ao mesmo tempo. Todo o processo é acompanhado pelo professor através de uma interface de administração da tarefa, assim que o aluno conclui sua programação o professor recebe um indicativo em sua tela. Uma vez concluída a programação o professor pode analisar o código desenvolvido pelo aluno ou selecionar o que vai rodar na Jabuti Edu selecionando o aluno na interface e disparando a execução do programa. Assim uma única Jabuti Edu pode ser utilizada por uma turma de alunos, sendo mediada pela

ação do professor na administração da tarefa e na execução dos programas desenvolvidos.

3.2.2 Camada de Adaptação

Camada intermediária que é responsável pela lógica de execução dos comandos da linguagem LOGO e tradução para os comandos de baixo nível que acionaram o hardware do robô. Para isso foi criado um interpretador de comandos em Python que recebe os comandos gerados pela camada de apresentação e converte para os comandos direcionados à GPIO (General Purpose Input/Output).

GPIO são portas programáveis de entrada e saída de dados que são utilizadas para prover uma interface entre os periféricos (motores e leds) e o microprocessador do Raspberry PI.

3.2.3 Camada de Hardware

Camada física formada pelos componentes de hardware envolvidos com a atuação do robô no mundo real. Composta por circuitos eletrônicos, sensores e atuadores é responsável pela movimentação do robô e todas as suas interações com o mundo físico.

Nesta camada encontram-se os servos motores, responsáveis pela movimentação da Jabuti Edu, os leds que representam seus olhos e a caixa de som que permite que o robô possa emitir sons e até falar. Todos estes componentes são controlados por uma placa de circuitos desenvolvida especialmente para a Plataforma Jabuti Edu.

4. ADAPTAÇÃO E CRIAÇÃO DO MÓDULO BLOCKLY

A Blockly é uma biblioteca de programação visual do Google, capaz de utilizar ferramentas gráficas para representar código, tais como: variáveis, expressões lógicas, laços e condicionais. É uma forma de representar graficamente a lógica de programação. A Blockly é compatível com as linguagens de programação: Javascript, Dart, Lua, PHP e Python.

Sua API de desenvolvimento foi utilizada em algumas versões do Scratch, e está servindo como base para a construção de sua versão 3.0, com data de lançamento ainda desconhecida [9]. A Blockly foi escolhida para este trabalho devido à sua diversidade de opções, que possibilitam facilitar a programação e a obtenção dos dados dos usuários. Dentre elas: Exportar todo o código dos blocos; criar blocos personalizados de acordo com sua preferência; converter o código para outra linguagem de programação; e editar toda a sua interface para que se adeque ao seu projeto.

Para utilizar a Blockly, basta fazer o download de seu código fonte pelo site *Google Developers*, e incorporá-lo ao seu código. A Blockly é open-source e compatível com aplicações web, Android e IOS.

4.1.1 Estrutura do Blockly

Para iniciarmos com a Blockly, devemos primeiramente demarcar a região da página onde será inserida toda a interface da Blockly, delimitando um tamanho fixo onde esta interface ocupará. Esta área é chamada de BlocklyArea. Toda a configuração e inserção da Blockly é feita usando a linguagem HTML. A interface padrão da Blockly possui alguns blocos pré-definidos, como: blocos de estruturas condicionais e de repetição (if, for e while); variáveis do tipo texto, numerais e lógicas; operações lógicas e aritméticas;

e a função print. Abaixo um exemplo da interface da Blockly (Figura 12).

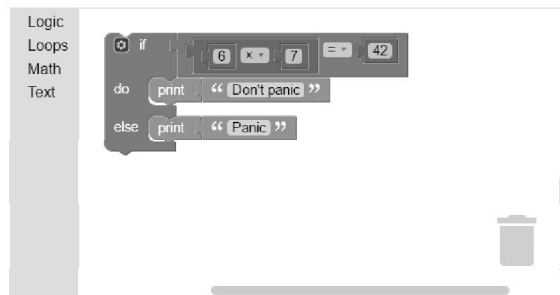


Figura 12 - Imagem ilustrativa da interface padrão da Blockly

Após delimitar a página e inserir a interface Blockly, é necessária a criação de blocos personalizados, que serão utilizados no projeto para representar os movimentos e as ações do robô.

4.1.2 Blocos Personalizados

Os blocos personalizados possibilitam a criação de novas funções à Blockly utilizando diversas linguagens, através de uma API própria da ferramenta, onde podemos colocar o trecho de código que queremos executar dentro de um bloco novo, personalizar sua aparência e incorporá-lo ao nosso código, de maneira simples e rápida. A personalização dos blocos é toda feita em uma interface disponibilizada pelo Google Developers, e utiliza a própria Blockly para criar novos blocos. A personalização dos blocos pode ser feita também alterando o código de cada bloco.

4.1.3 Estrutura dos blocos

Os blocos possuem uma estrutura que permite não somente alterar sua aparência dentro de seu código, mas também, alterar o comportamento de códigos nativos da Blockly. A estrutura da Blockly possui: Variáveis de configuração, que permitem alterar a aparência dos blocos, tais como a cor, o formato e o texto do bloco; Variável de ajuda, que permite a inserção de um link de ajuda para cada bloco; Objetos de ligação, que nos permitem restringir a ligação de blocos à outros, definindo um bloco de somente ligação no início ou fim; E também Objetos auxiliares, com os quais podemos inserir um menu drop-down ao bloco, uma figura, um botão switch, um link, um calendário, ou até mesmo um menu circular para a seleção de ângulos. A estrutura acima permite modificar a aparência do bloco, mas somente isso não é suficiente para que o bloco fique funcional, é necessária a criação de uma função block para que o bloco faça algo.

Toda e qualquer função bloco, deve ser colocada em uma variável chamada code, tal variável armazena todo o código que executará naquele bloco, juntamente com outra variável opcional, chamada option, que armazenará a opção selecionada pelo usuário naquele bloco, por exemplo, se possuíssemos um menu drop-down com diversas opções, a opção selecionada pelo usuário seria armazenada na variável option. Em resumo, os blocos possuem duas estruturas fundamentais, a estrutura onde definimos as configurações do bloco e a estrutura da função que queremos executar através daquele bloco (Figura 13).



Figura 13 - Imagem demonstrativa da estrutura dos blocos personalizados

4.1.4 Adicionando o Blockly ao conjunto de módulos

O código do Jabuti é aberto e disponibilizado no site do projeto (*jabuti.org*), permitindo que usuários do Jabuti Edu que possuem algum conhecimento em programação alterem seu código. Para a criação do módulo Blockly, precisamos modificar alguns comportamentos na execução de comandos do módulo atual do Jabuti Edu.

A Programação Baseada em Blocos foi adicionada à Plataforma Jabuti Edu como um novo Módulo, no qual a sua interface foi construída a partir da integração com a biblioteca Blockly. Após a criação da interface, foram adicionados os blocos personalizados que representam as ações do Robô.

Algumas alterações precisaram ser feitas na camada de adaptação, para que o Interpretador de comandos pudesse interagir com a Blockly. Essas mudanças no Interpretador não diziam respeito especificamente à integração com a Blockly, mas estavam relacionadas à alguns bugs que só foram evidenciados com a execução simultânea de vários código enviados pela novo módulo.

5. ETAPAS DA ADAPTAÇÃO

A criação do Módulo Blockly foi realizada em três etapas consecutivas que buscaram refinar a integração do código da biblioteca com o código já existente da camada de apresentação da Jabuti Edu.

5.1 Primeira Etapa

Na primeira etapa acreditava-se que a adaptação da Blockly seria apenas feita na camada de apresentação da Jabuti Edu. De acordo com a documentação da arquitetura isso seria o mais natural de se fazer. Nesta etapa uma tela simples foi criada para receber a Blockly. Blocos de testes foram criados para verificar a comunicação entre os blocos e as leds do robô.

Em seguida as demais funções de programação foram inseridas, tais como laços, variáveis, condicionais, comparações e funções. Das funções próprias do robô, todas estavam em funcionamento, entretanto, havia um problema a ser resolvido na execução.

O Interpretador havia sido construído para executar comandos individualmente assim que chegavam, porém o novo módulo enviava vários comandos provenientes dos diversos blocos construídos. O problema se dava que enquanto a Jabuti Edu estava executando um comando, como por exemplo, andar para frente, um novo comando chegava e era ignorado pelo Interpretador. Logo o Interpretador precisava ser reescrito.

5.2 Segunda Etapa

Nesta etapa do processo de criação do módulo Blockly, a interface não foi alterada, nem a programação por trás dos blocos personalizados. Nesta versão, observou-se a necessidade de alterar o comportamento de algumas funções próprias do Jabuti Edu referentes à execução de comandos.

Observou-se que o Interpretador era chamado de forma individual para cada comando gerado na camada de apresentação da Plataforma Jabuti Edu. Isso causou um problema quando o Blockly foi integrado, pois os comandos foram enviados todos de uma só vez e o Interpretador não estava pronto para isso, fazendo com que alguns desses comandos fossem perdidos durante a execução do programa criado no novo Módulo.

A Solução foi adaptar o Interpretador para enfileirar os comandos recebidos antes de executá-los, dividindo a responsabilidade entre o recebimento do comando e sua execução (Figura 14). O Interpretador foi dividido internamente em dois elementos: O enfileirador (*Queuer*), que recebe os comandos advindos da camada de apresentação e os coloca em uma fila para serem posteriormente executados; e o executor (*Runner*) que percorre a lista de comandos e a executa em sequência.

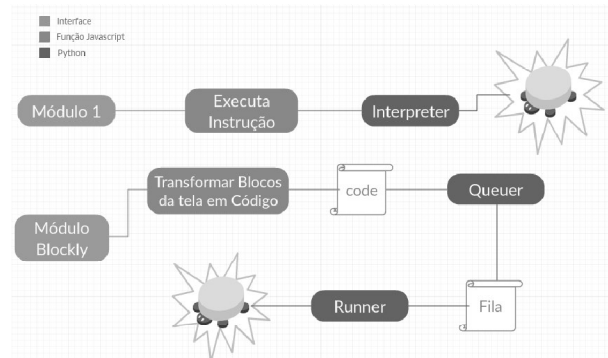


Figura 14 - Imagem demonstrativa do processo de execução nos dois módulos

5.3 Terceira Etapa

Nesta etapa foi criado um botão para parar um código em execução, isso se fez necessário a partir do momento que um programa criado no novo módulo poderia ficar muito tempo em execução, por ter o mesmo um loop demorado ou mesmo por ter entrado em loop infinito.

Uma vez disparado um código o Interpretador o executa continuamente até que o mesmo chegue ao fim, mas em algum momento o usuário pode querer para a execução do código, e isso era impossível a partir da interface Web, pois o código estava sendo executado pelo Interpretador que foi escrito em Python.

A solução encontrada até o momento foi a execução de um novo processo do Interpretador que pegará o id do processo em execução e o interromperá, parando de forma abrupta a execução do código enviado pelo módulo Blockly.

Esta solução não nos agrada, pois acreditamos não ser a melhor maneira de implementar a interrupção da execução dos programas criados pelos usuários. Em vista disso, estamos trabalhando atualmente na camada de adaptação para desenvolver uma solução melhor. Segue abaixo, a Figura 15 que demonstra a interface da versão após à terceira etapa.

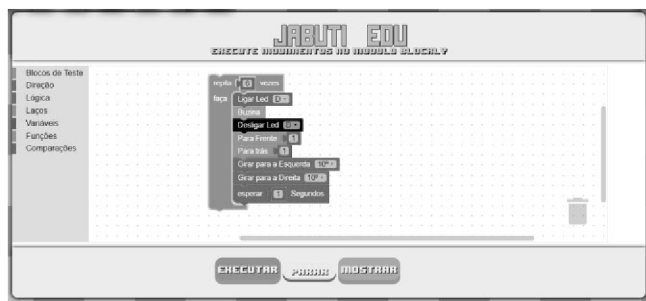


Figura 15 - Interface do módulo Blockly em sua versão finalizada

6. CONSIDERAÇÕES FINAIS

A programação visual é um campo promissor a ser explorado, principalmente quando se trata de aprendizado de programação. Quando estamos aprendendo a programar, temos que lidar com uma série de conceitos, como sintaxe da linguagem, estrutura do código e regras de escrita, que às vezes até nos prendem diante do entendimento do principal, que é a lógica por trás da programação. O uso de uma ferramenta que torne essa etapa de ensino mais fácil e divertida é uma excelente maneira para auxiliar futuros programadores, e atrelada à um robô que permita que o resultado de seus códigos sejam representados visualmente, é ainda melhor. A Jabuti Edu integrada à interface do Blockly proporciona uma experiência de aprendizagem lúdica e construtiva ao mesmo tempo, capaz de estimular não somente a criatividade e interesse pela computação, mas também auxiliar no aprendizado de outras habilidades, como lógica, matemática, e até mesmo a escrita.

O desenvolvimento do presente trabalho gerou não só a união de duas ferramentas de programação e robótica voltadas para o ensino, mas também propôs alternativas que aprimorem ainda mais o uso do Jabuti Edu. A camada de adaptação foi melhorada e alguns de seus bugs corrigidos.

Alguns requisitos ainda precisam ser atendidos, como o botão de parada de execução. Outras pequenas melhorias também precisam ser alcançadas, como a velocidade na execução dos códigos. A Plataforma Jabuti Edu está em constante evolução e muito ainda temos que melhorar.

A biblioteca Blockly é uma ótima ferramenta de desenvolvimento de linguagens visuais baseadas em blocos, é possível adaptá-la à várias situações e ferramentas. Isso reduziu bastante o tempo de desenvolvimento de um Módulo de Programação em Blocos para a Plataforma Jabuti Edu, e com certeza pode ser utilizada por outros projetos que tenham tal necessidade.

7. REFERENCIAS

- [1] Papert, S. (1980). Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, Inc., New York, NY, USA.
- [2] BLOCKLY. Site Oficial para desenvolvedores. Disponível em: <<https://developers.google.com/blockly/>>. Acessado em: Outubro 2016.
- [3] Papert, S. (1985). Logo: computadores e educação. Editora Brasiliense, São Paulo.
- [4] LEGO Mindstorms.. Site Oficial. Disponível em: <www.lego.com/en-us/mindstorms >. Acessado em: Outubro 2016.
- [5] ARDUINO. Site Oficial. Disponível em: <www.arduino.cc >. Acessado em: Outubro 2016.
- [6] RASPBERRY Pi. Site Oficial. Disponível em: <www.raspberrypi.org>. Acessado em: Outubro 2016.
- [7] Saygin, C., Yuen, T., Shipley, H., Wan, H., and Akopian, D. (2012). Design, development, and implementation of educational robotics activities for k-12 students.
- [8] Saleiro, M., Carmo, B., Rodrigues, J. M., and du Buf, J. H. (2013). A low-cost classroom-oriented educational robotics system. In Social Robotics, pages 74–83.
- [9] SQUEAK ETOYS. Site Oficial. Disponível em: <<http://www.squeakland.org/>>. Acessado em: Outubro 2016
- [10] ALICE. Site Oficial. Disponível em: <www.alice.org>. Acessado em: Outubro 2016.
- [11] SCRATCH. Site Oficial. Disponível em: <<https://scratch.mit.edu/>>. Acessado em: Outubro 2016.
- [12] ARDUBLOCK. Site Oficial. Disponível em: <<http://blog.ardublock.com/>>. Acessado em: Outubro 2016.
- [13] APP INVENTOR. Site Oficial. Disponível em: <appinventor.mit.edu>. Acessado em: Outubro 2016.
- [14] SWIFT PLAYGROUNDS. Site Oficial. Disponível em: <www.apple.com/swift/playgrounds/ >. Acessado em: Outubro 2016.
- [15] PHP. Disponível em: <<http://php.net/>>. Acessado em: Outubro 2016.
- [16] Springer.Apellido1, N1. & Apellido2, N2. (2007) Formato de Referencias. En Memorias del Décimo Taller Internacional de Software Educativo TISE 2007, 1-3 Diciembre, Santiago, Chile pp. 20-21.