*INTERMATHS, VOL. 4, NO. 2 (**2023**)*, **112–134**
https://doi.org/10.22481/intermaths.v4i2.13983

Ensino de Matemática com a Robótica: Uma Proposta de Recurso Didático

Mathematics Teaching Using Robotics: A Proposal of a Teaching Resource

Carlos Alberto dos Santos Lima [a], **Cleusiane Vieira Silva** [a]

[a] Universidade Estadual do Sudoeste da Bahia, Jequié - BA, Brasil

* Autor Correspondente: cleusiane.vieira@uesb.edu.br

Resumo: Este artigo é fruto de uma pesquisa já concluída, em que se investigou como a robótica pode ser utilizada como recurso didático aliado a metodologia de resolução de problemas e a ideia de gamificação para o ensino e a aprendizagem de matemática. No presente trabalho temos por objetivo relatar o processo de criação de um Robô Educacional, para ser utilizado como recurso didático nas aulas de matemática. A proposta apresentada contempla a construção deste Robô, dividida em duas partes: mecânica e a construção do algoritmo do projeto. A parte mecânica trata da descrição dos componentes do Robô Educacional e sua montagem, já na parte da construção do algoritmo são expostos os códigos que permitem o funcionamento do Robô. A utilização da robótica educacional poderá auxiliar o aluno a desenvolver: lateralidade, noções de espaço, conceitos geométricos, raciocínio lógico, interpretação e análise de dados, criticidade e tomada de decisões. Na pesquisa citada, o Robô Educacional foi utilizado para estimular os discentes a resolver uma lista de tarefas matemáticas como forma revisar o conteúdo estudado no ano letivo anterior. Ressaltamos que, neste caso, o Robô Educacional associado a gamificação foi um recurso didático capaz de instigar os discentes na realização das tarefas propostas nas aulas de matemática e proporcionar um ambiente de aprendizagem por meio do trabalho colaborativo. Por outro lado, como resultado do processo de construção do Robô Educacional, obtemos um recurso didático de baixo custo e que pode ser confeccionado por professores de todos os níveis de ensino e alunos de graduação.

Palavras-chave: Robótica Educacional; Ensino de Matemática; Recurso Didático.

Abstract: This article is the result of an already complete research, which investigated how robotics can be used as a teaching resource combined with problem-solving methodology and the idea of gamification for teaching and learning mathematics. In this work we aim to report the process of creating an educational robot, to be used as a teaching resource in Mathematics classes. The proposal presented includes the construction of this robot, divided into two parts: mechanics and the construction of the project algorithm. The mechanics part deals with the description of the components of the educational robot and its assembly, while in the algorithm construction part the code that allow the Robot to function are exposed. The use of educational robotics can help the student to develop, notions of laterality, space, geometric concepts, logical reasoning, interpretation and analysis of data, criticality and decision making. In the aforementioned research, the educational robot was used to encourage students to solve a list of mathematical tasks as a way to review the content studied in the previous academic year. We emphasize that, in this case, the Educational Robot associated with gamification was a teaching resource capable of encouraging students to carry out the tasks proposed in mathematics classes and providing a learning environment through collaborative work. On the other hand, as a result of the construction process of the educational robot, we obtain a low-cost teaching resource that can be created by teachers of all educational levels and undergraduate students.

keywords: Educational Robotics; Teaching Mathematics; Teaching Resource.

Submetido em: 24 de Novembro de 2023 Aprovado em: 20 de Dezembro de 2023 Publicado em: 30 de Dezembro de 2023
ISSN 2675-8318

1 Introdução

Os recursos didáticos exercem um papel de suma importância no ensino e na aprendizagem de matemática ao auxiliar o aluno na compreensão do conteúdo abordado em sala de aula, além de tornar as aulas dinâmicas e lúdicas. Silva *et al.* [1] salientam que a ludicidade torna esse processo de construção do conhecimento prazerosa e descontraída quebrando o paradigma das aulas apenas expositivas e monótonas.

Para o professor é um grande desafio despertar, manter a atenção e a curiosidade dos alunos, de forma que ocorra uma aprendizagem matemática significativa. Assim, criar estratégias ou aprimorar as já existentes é uma maneira de minimizar este problema. Cruz [2] ressaltou o elevado nível de dependência dos adolescentes em *smartphone*, que dentre os 264 entrevistados, 73% têm vício moderado e 20% grave.

Dentre as estratégias de ensino temos inúmeros materiais pedagógicos que podem auxiliar nas aulas de matemática, segundo Souza [3] qualquer material que o professor utilizar em sala de aula para ensinar matemática pode ser entendido como um recurso didático, tais como: revista, tv, jornais, computadores, jogos, multimídia entre outros.

Percebe-se que a nova geração de alunos está voltada para as novas tecnologias. Por isso, os educadores precisam acompanhar a revolução tecnológica, a fim de, pertencer à mesma realidade dos alunos. Já que conforme Braz e Vilela [4], o que faz efeito para a aprendizagem hoje pode não fazer diferença amanhã. Além disso, de acordo com Sacomano *et al.* [5] estamos na quarta era da revolução industrial, chamada de indústria 4.0. Nesse sentido, para entender qual é o impacto que os recursos didáticos, construídos com as ferramentas atuais desta indústria, têm no ensino de matemática, neste trabalho foi relatada a produção de um robô com propósito educacional.

2 A Construção do Robô Educacional

A inspiração para construir o Robô Educacional nasceu durante as aulas do curso de licenciatura em Matemática da Universidade Estadual do Sudoeste da Bahia (UESB), a partir das componentes curriculares: Linguagem de Programação (LP); Introdução à Ciência da Computação (ICC); Banco de Dados e Informática Aplicada na Educação. Notou-se nestas componentes um potencial para a produção de recursos didáticos que podem ser aplicados nas aulas de matemática.

Nas componentes curriculares, Linguagem de Programação (LP); Introdução à Ciência da Computação (ICC) e Banco de Dados obtemos o conhecimento inicial para escrever os códigos de programação do Robô Educacional. Já na componente curricular Informática Aplicada na Educação, uma das linguagens de programação que conhecemos e aprendemos foi o LOGO. O programa LOGO é interessante pois faz uso do computador e da lógica de programação para fazer uma tartaruga andar pelo cenário, desenhando figuras geométricas de acordo com o interesse do usuário. Este programa de computador, o LOGO, fica apenas na atmosfera digital. Não podemos manipular concretamente a tartaruga, seria interessante se ela fosse "palpável". Isso só seria possível se fizéssemos o uso da robótica para unir o abstrato ao concreto.

O Robô Educacional foi construído com materiais de baixo custo e de fácil aquisição, tais como: uma vasilha com tampa, um Arduino Pro Mini, que atua como o controlador do robô; temos dois motores de passos, que exerce a função de locomoção, duas rodas de borracha e uma roda esférica. Um suporte para quatro baterias 18650* para alimentação do projeto, um módulo *bluetooth* para conexão com o *smartphone* e duas abraçadeiras de cano para a fixação dos motores. O objetivo da escolha deste material foi alcançar os profissionais da educação que desejam construir e utilizar esse recurso educacional. A seguir descreveremos os passos para a construção do Robô Educacional.

2.1 Parte Mecânica

Para construir o Robô Educacional foi utilizada uma vasilha plástica com tampa, que tinha 19 cm de diâmetro e 10 cm de altura (essas medidas são as medidas mínimas para realizar o projeto). A tampa da vasilha foi a base do Robô onde todos os elementos mecânicos e eletrônicos foram fixados. A vasilha teve a função de proteção dos componentes sendo, também, o corpo do Robô. Observe a Figura 1 a seguir.

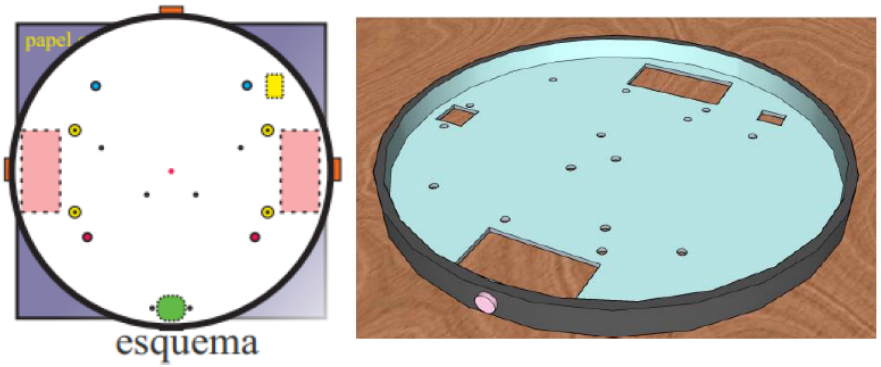
Figura 1. Vasilha utilizada para construção do robô

Fonte: fotografia realizada pelos autores

Para montar a base do robô construímos um desenho esquemático, Figura 2, com todas as medidas, posições dos furos e cortes necessários. Fixamos com uma fita adesiva o desenho esquemático sobre a tampa e transportamos as medidas para a superfície plástica (o que pode ser feito utilizando papel-carbono), furamos todos os pontos marcados no desenho e recortamos todos os retângulos.

*As baterias de Lítio 18650, são nomeadas assim por terem 1.8 cm de diâmetro por 6.5 cm de comprimento. Estas baterias são as mesmas utilizadas em *notebook*.

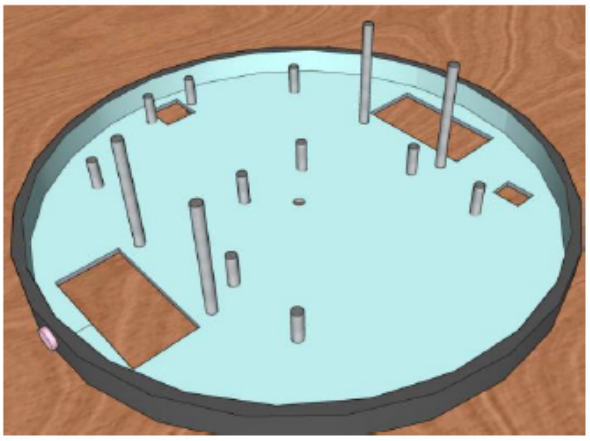
Figura 2. Processo de marcação e furos para fixação dos componentes do robô



Fonte: figura construída pelos autores

Para montar a base do robô, Figura 3, necessitamos de alguns parafusos com porcas e arruelas. Esses materiais são de baixo custo e podem ser encontrados em casas de materiais de construções. As quantidades e medidas desses parafusos são: 6 parafusos com porcas de 5 cm, para fixar os motores; 2 parafusos com porcas de 2 cm, para fixar a *protoboard*; 2 parafusos com porcas de 2 cm, para fixar o porta bateria; 4 parafusos com porcas de 2 cm, para fixar os drives; 2 parafusos com porcas de 2 cm, para fixar a roda esférica; 3 parafusos de 2 cm, para fixar o corpo do robô na base, e, por fim, 24 arruelas.

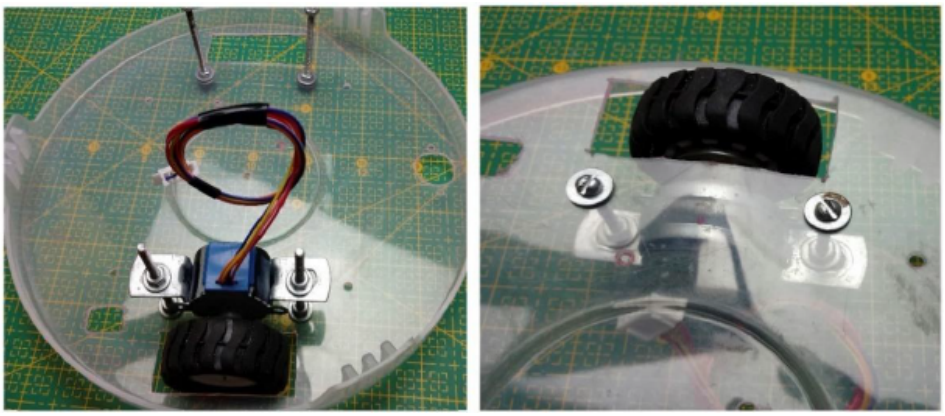
Figura 3. Anexação dos parafusos para fixação dos componentes do robô



Fonte: figura construída pelos autores

Fixamos os quatro parafusos principais que serviram para sustentar os dois motores de passos e as rodas do robô. Em seguida, rodas e motores foram colocados, conforme Figura 4 a seguir.

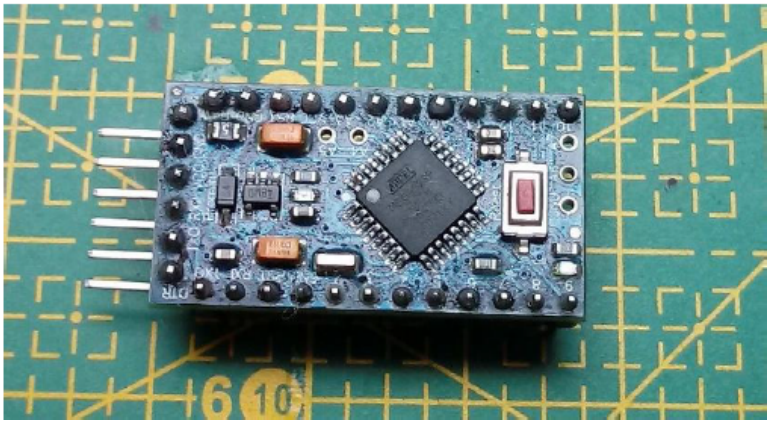
Figura 4. Parafusando os motores e as rodas na base do robô



Fonte: fotografia realizada pelos autores

O passo seguinte na montagem do robô foi acoplar o Arduino, que é uma placa de circuito impresso com microcontrolador, sendo ele capaz de controlar vários atuadores[†] como motores de passos e sensores.

Figura 5. Arduino Pro Mini



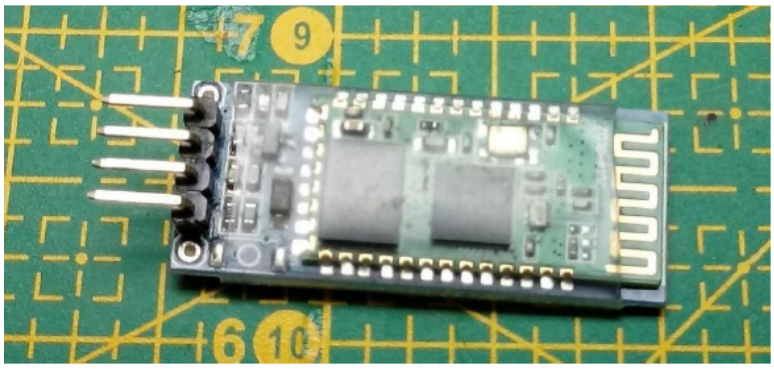
Fonte: fotografia realizada pelos autores

Para programar o Arduino, Figura 5, usamos o Arduino IDE, um programa de computador em que escrevemos os códigos em C/C++.

O módulo *bluetooth*, Figura 6, foi o componente eletrônico utilizado para conectar o Arduino remotamente ao celular ou *notebook*.

[†]Os atuadores têm a função de converter um sinal (geralmente elétrico) em ação de algum tipo, geralmente singular [6]

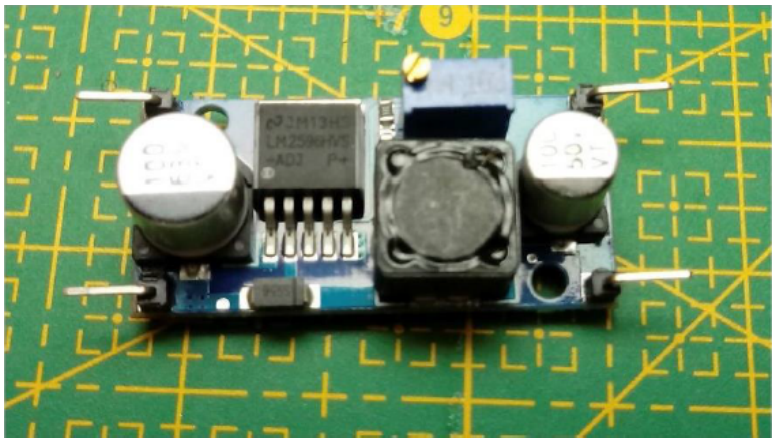
Figura 6. Módulo *bluetooth*



Fonte: fotografia realizada pelos autores

A minifonte, Figura 7, tem a função de converter a voltagem de 14 volts fornecida pelas baterias, para uma tensão menor de 5 volts suportada pelo Arduino, motor de passo, servo motor e o módulo *bluetooth*.

Figura 7. Mini fonte para convenção de energia



Fonte: fotografia realizada pelos autores

O motor de passo e seu *driver* foram os responsáveis pela tração do robô, Figura 8, sendo que este componente faz o sistema deslocar sob a superfície.

Figura 8. Motor elétrico de passo com suporte

Fonte: fotografia realizada pelos autores

Nesse projeto foi essencial usar motores desse tipo, pois existia a necessidade conseguirmos precisão de movimento.

Para o projeto ter autonomia energética, ou seja, ter um tempo maior de execução, foram utilizadas células de bateria de lítio 18650 (Figura 9), pois elas possuem capacidade de armazenamento de energia elevada e, também, são recarregáveis.

Figura 9. Bateria de lítio 18650

Fonte: fotografia realizada pelos autores

Foram utilizadas quatro baterias, Figura 9, ligadas em série por meio do suporte de bateria, a voltagem máxima fornecida por esse sistema é de 14.8 volts.

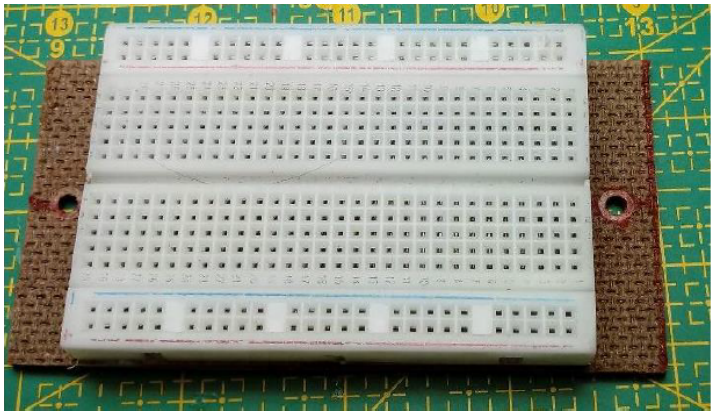
Um par de rodas de borracha, uma para cada motor de passo e uma roda pequena foram utilizadas para sustentação da base do robô e facilitar a sua locomoção. Veja Figura 10 abaixo.

Figura 10. Rodas usadas para locomoção do robô

Fonte: fotografia realizada pelos autores

Para facilitar a montagem de todos os componentes eletrônicos de modo que não tivéssemos a necessidade de utilizar solda, usamos a *protoboard* (Figura 11), um acessório que fornece a praticidade de fazer conexões apenas fixando pinos e fios.

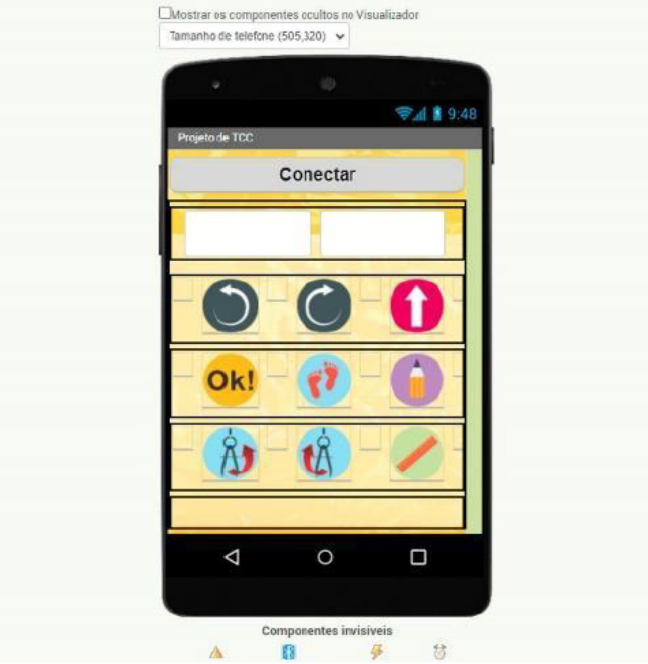
Figura 11. *Protoboard* para construção do circuito eletrônico



Fonte: fotografia realizada pelos autores

Para controlar o Robô foi usado um aplicativo de *smartphone* construído por meio da plataforma de desenvolvimento Mit APP Inventor. Esse site é gratuito e fornece uma ferramenta de fácil utilização para a elaboração de *App*. Veja na Figura 12, a seguir, a tela de iniciação do aplicativo no celular.

Figura 12. Aplicativo utilizado para controlar o robô

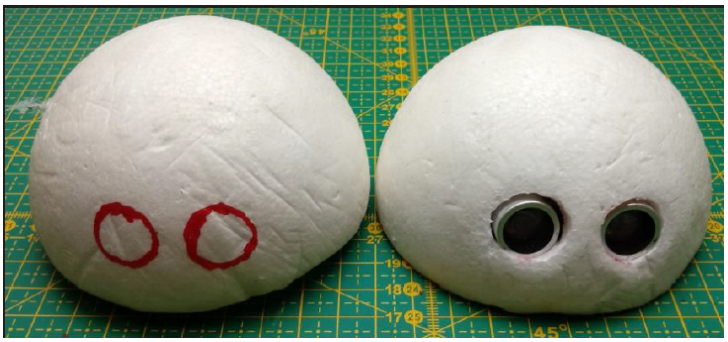


Fonte: fotografia realizada pelos autores

Depois de conectar o celular ao Robô Educacional, para movê-lo, digite na caixa de diálogo a distância em centímetros que ele deve percorrer e escolha a direção nos botões de giro ou de seta. Assim, o Robô será controlado com movimentos de acordo com o interesse do operador.

A cabeça do robô foi feita com a metade de uma bola de isopor. Esta bola de isopor é oca e pode se encontrada em papelaria ou casa de decorações. Dentro dela colocamos um sensor de distância para ser os "olhos" do robô e um servo motor para fazer a cabeça girar para os lados, conforme Figura 13 a seguir.

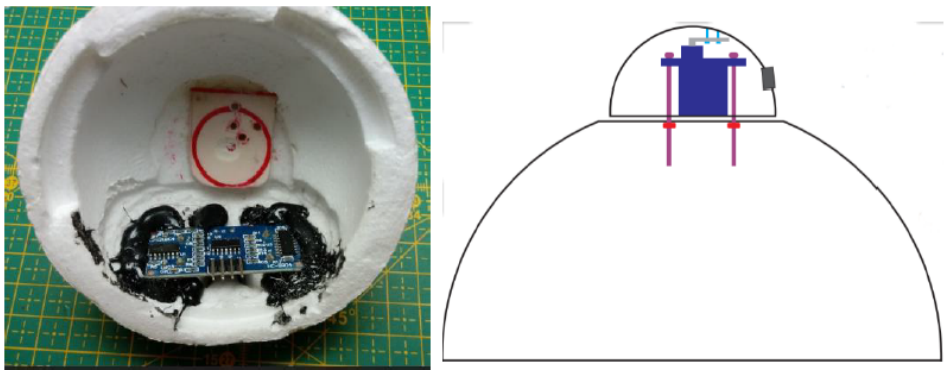
Figura 13. Montagem da cabeça do robô



Fonte: fotografia realizada pelos autores

Observe que esses dois furos têm dimensões suficiente para inserirmos o sensor sendo fixado com cola quente apenas pela borda, como mostrado na Figura 14.

Figura 14. Pequena chapa na cabeça do robô e sistema mostrando a cabeça anexada ao corpo por dois parafusos



Fonte: fotografia e esquema realizados pelos autores

Para fixar a cabeça do robô no corpo foi preciso colocar, usando supercola, um pequeno pedaço de chapa de 4 cm $\times$ 4 cm no centro da bola, como mostrado na Figura 14. Usamos esse procedimento para parafusar o servo motor no centro da bola, caso não tivesse esta chapa, o parafuso ficaria apenas no isopor, podendo soltar-se imediatamente. A aparência do Robô Educacional[‡] se assemelha com a apresentada na Figura 15 a seguir:

Figura 15. Resultado final do Robô Educacional

Fonte: fotografia e esquema realizados pelos autores

Na seção a seguir descreveremos o projeto de programação do Robô Educacional, o que envolve a linguagem de programação C/C++.

2.2 Desmembrando o algoritmo do projeto

A peça fundamental para a construção de qualquer máquina que exija previsibilidade e exatidão em seus movimentos é a escolha certa dos atuadores, eles transformam

[‡]As etapas da montagem do Robô Educacional, com todos os componentes incluindo a cabeça e sua funcionalidade, estão disponíveis em vídeo, gravado pelo primeiro autor, no Youtube cujo endereço eletrônico é https://www.youtube.com/playlist?list=PLJ_XBkc0KIrKb_jZf7ZOwQGTblmYuD2CN.

energia elétrica em mecânica. Dessa forma, quando se trata de movimentos controlados e precisos são utilizados os motores de passo e o servo motor. Os motores de passo atuam com a deslocação em seu eixo principal dando pequenos passos em ângulos menores que um grau por vez. Enquanto o servo motor trabalha diretamente com ângulos que variam de 0 a 180º graus. Sendo assim, o desenvolvimento do algoritmo que comandará a máquina deverá ser escrito em função dos atuadores ou sensores que compõem esta máquina.

Em projetos de robótica para iniciantes o Motor de Passo 28BYJ-48 é muito utilizado por ter um custo relativamente baixo e ser funcional, pois ele já vem com o seu próprio drive de controle e, além disso, tem um baixo consumo de energia em relação aos outros. O passo deste motor é de 0,088º a cada pulso, isso significa que para seu eixo girar 360º é preciso de $360°/0,088° = 4090$ passos ou pulsos. Note que, com o controle de cada passo temos altíssima precisão de movimento, visto que o motor não gira descontroladamente igual ao motor DC[§] normal. Dessa maneira, o algoritmo foi desenvolvido convertendo esses passos/pulsos em distâncias (centímetros). Para desenvolver um algoritmo foi necessário um propósito bem definido, com todas as metas e problemas que nos propomos alcançar e resolver. Na construção desse Robô para aplicação didática em aulas de matemática, foi necessário que ele tivesse movimentos precisos em seu deslocamento retilíneo ou angular. Contudo, foi preciso saber quantos centímetros as rodas do Robô percorrerem na superfície ao completar uma volta completa de 360º. Para isso, usamos a seguinte fórmula algébrica: Perímetro $= 2 \times \pi \times$raio. Note que, é preciso uma função para converter a distância (centímetros) para passos, para isso, foi usada regra de três simples. Sabemos que quando o motor executa 4090 passos o robô percorrerá o valor do perímetro. Diante dessa informação foi possível prever quantos passos o motor precisaria dar para percorrer uma determinada distância exata. Veja a seguir como foi deduzida a principal função do código:

$$\begin{aligned} &4090 \rightarrow \text{perímetro} \\ &\text{passos} \rightarrow \text{distância} \\ &\text{passos} = (4090/\text{perímetro}) \times \text{distância} \end{aligned} \tag{1.2.1}$$

Observe que o valor do perímetro dependerá das dimensões da roda que for utilizada no projeto. Note que, esse valor é calculado apenas uma vez, dessa forma, no lugar da variável perímetro você pode substituir pelo valor numérico que já foi obtido. Assim, a função passos dependerá apenas da distância. Essa função será aplicada para fazer o Robô se movimentar tanto para frente quanto para trás em linha reta, percorrendo distâncias precisas.

Para o Robô rotacionar em seu próprio eixo com ângulos precisos foi necessário deduzir outra fórmula. Primeiramente as duas rodas precisam girar em sentido contrário entre si, Figura 16, gerando assim uma rotação em seu eixo.

[§]Um motor DC é um tipo de máquinas elétricas que converte energia elétrica de corrente contínua em energia mecânica. Os tipos mais comuns dependem das forças produzidas por campos magnéticos.

Figura 16. Rotação do robô

Fonte: Construção dos autores

Note que quando essas rodas girarem farão um círculo, sendo necessário calcular o perímetro dele também. É preciso utilizar a medida do raio sendo a metade da distância entre as duas rodas pois, "a localização do robô é representada por um ponto que fica no ponto médio entre as suas duas rodas" [7, pp. 26]. A partir daí deduzimos a seguinte regra de três: quando o Robô girar 360^o ele percorrerá um perímetro completo em torno de si próprio. Assim temos que:

$$\begin{aligned} &360^o \quad \rightarrow \text{ perímetro} \\ &\text{ângulo} \rightarrow \text{ rotação} \\ &\text{rotação} = (\text{perímetro} \times \text{ângulo})/360 \end{aligned} \tag{1.2.2}$$

O perímetro nessa função dependerá da distância entre as duas rodas, assim, igualmente na fórmula anterior, só precisamos calcular uma única vez depois de estabelecida a distância delas. Desse modo, a aplicação rotação ficará em função apenas do ângulo. Contudo, observe que o valor obtido da função rotação não está em passos ou pulsos, assim, precisamos converter o valor da rotação em passos. Observe que temos dois valores de perímetros: o primeiro obtido da circunferência presente na roda que está acoplada no motor; e o segundo retirado da rotação que as rodas produzem em torno do eixo central do robô. O perímetro da roda está diretamente ligado com os passos ou pulsos. O segundo perímetro está em conexão com a rotação do robô e ambas são distâncias em centímetros. Diante dessas informações, temos as seguintes relações:

$$\begin{aligned} &4090 \quad \rightarrow \text{perímetro roda} \\ &\text{passos} \rightarrow \text{rotação} \\ &\text{passos} = (4090 \times \text{rotação}) \,/\, \text{perímetro roda} \end{aligned} \tag{1.2.3}$$

Veja que nessa nova fórmula o perímetro já pode ser calculado e acrescentado como constante. Esse fato ocorre porque o comprimento das rodas não varia. Dessa maneira, a função passos fica em função apenas da rotação. Essas funções funcionaram da seguinte forma: quando o usuário informar o valor do ângulo a função rotação fornecerá uma certa distância em centímetro, que pode ser, menor, maior ou igual ao comprimento do perímetro do Robô. A função passo receberá o valor em centímetro da rotação

e transformará em passos ou pulsos precisos, fazendo os motores rotacionarem com exatidão ao ângulo desejado. Está ocorrendo que as rodas estão se movimentando a distância do arco gerado por um determinado ângulo. Ou seja, percorrendo um arco resultará em um ângulo. Dessa forma, trabalhamos com as duas fórmulas em conjunto.

$$\text{rotação} = (\text{perímetro} \times \text{ângulo}) / 360$$
$$\text{passos} = (4090 \times \text{rotação}) / \text{perímetro\_roda} \tag{1.2.4}$$

Como já foram obtidas as principais fórmulas do programa, vamos agora construir um algoritmo que permite fazer as entradas de dados quando solicitado pelo usuário. Essas informações serão comunicadas por meio de um aplicativo de celular via *bluetooth*.

Inicialmente precisamos importar quatro bibliotecas que são: *AccelStepper.h*; *MultiStepper.h*; *stdio.h*; *Servo.h*. A primeira e a segunda são utilizadas para manipular o motor de passo, a terceira é específica para escrever textos de comando e a última para comandar o servo motor. Caso fôssemos trabalhar com mais módulos seria preciso adicionar bibliotecas específicas para eles. Em nosso caso, serão utilizados apenas essas:

```
\#include $<$AccelStepper.h$>$

\#include $<$MultiStepper.h$>$

\#include $<$stdio.h$>$

\#include $<$Servo.h$>$
```

Para usar o servo motor é preciso criar um objeto do tipo Servo. Nesse exemplo, chamamos esse objeto de *myservo*, contudo, pode-se nomear segundo a preferência do programador. Por meio do *myservo* podemos controlar o posicionamento da alavanca do servo motor e até da sua velocidade. Esse recurso será importante para fazer a cabeça do Robô girar para os lados, ou criar outros mecanismos como, por exemplo, um braço para manipular algum objeto.

Para o motor ter maior precisão podemos dividir ainda mais seus passos, para isso, vamos definir algumas constantes, *FULLSTEP* e *HALFSTEP*.

```
//Exemplos:
    \#define FULLSTEP 4

    \#define HALFSTEP 8
```

Nas linhas de códigos a seguir foram definidas as portas digitais do Arduino em que cada conexão do motor foi ligada. Veja que o motor 1 é ligado nas portas digitais (2, 3, 4, 5) e o motor 2 é ligado nas portas (6, 7, 8, 9). Note que podemos ligar os motores em outras portas, contudo, precisaríamos trocar também a numeração referente a entrada que foi utilizada nas linhas abaixo.

```
    \#define motorPino1 2  //  Pino do driver IN1

    \#define motorPino2 3  //  Pino do driver IN2

    \#define motorPino3 4  //  Pino do driver IN3

    \#define motorPino4 5  //  Pino do driver IN

```

```
9          \#define motorPino5 6  //  Pino do driver2 IN
10
11         \#define motorPino6 7  //  Pino do driver2 IN
12
13         \#define motorPino7 8  //  Pino do driver2 IN3
14
15         \#define motorPino8 9  //  Pino do driver2 IN4
```

No projeto são utilizados dois motores de passo, dessa forma, precisam ser criados dois objetos do tipo 0 e, além disso, comunicar os parâmetros dos pinos e da precisão. O objeto *stepper* é vinculado ao motor 1 e o stepper2 ao motor 2. Com esses objetos será possível acessar as funções da biblioteca *Accelstepper*.

```
1           AccelStepper stepper (HALFSTEP,motorPino1,motorPino3,motorPino2,motorPino4);
2           AccelStepperstepper2 (HALFSTEP,motorPino5,motorPino7,motorPino6,motorPino8);
```

Nas linhas a seguir foram definidas algumas variáveis, dentre estas, a velocidade e a *distRoda* precisam de um pouco de atenção. A variável *distRoda* receberá o valor da distância das duas rodas que estão montadas no Robô. Esse valor influencia no cálculo do perímetro das fórmulas mostradas anteriormente. A variável velocidade pode receber um parâmetro de até 2000 passos, contudo, nos testes realizados pelos pesquisadores a velocidade de 500 passos forneceu melhores resultados.

```
1          String leeCadena; int  velocidade = 500, velocidade1, velocidade2;
2
3          int  direcao,direcao1,direcao2,direcao3=1;
4
5          int  distRoda = 16, raio, passo, reta, seta, cont=0, corrente, desligado,
               ligado;
6
7          int angulo, angulo1, direita, esquerda, servo, cima, baixo;
8
9          int codigoSerial = 0, protocolo, contat=0;
10
11         long pass, dist, comArco1, comArco2, comArco3, comArco4, rotacao,compcen1,
               compcen2;
```

Todas as linhas dos códigos dentro do setup são lidas apenas uma vez quando o Arduino é ligado, dessa maneira, ele foi utilizado para configurar inicialmente o programa. O *serial.begin* possibilita analisar o código funcionando pelo monitor serial da IDE. O *stepper.setMaxSpeed()* configura o motor 1 para ter uma velocidade de até 1000 passos, da mesma forma que o *stepper2.setMaxSpeed()*. O *myservo.attach()* indica para o Arduino o pino que o servo motor está conectado na placa. Se o servo motor estiver em outro pino digital é preciso informar esse valor para o *myservo.attach()*. Por sua vez, o *myservo.write()* é usado para comunicar o valor do ângulo para o servo motor executar. Nesse caso, toda vez que o Arduino for ligado, o servo motor irá para a posição de 90º. Como ele fica preso na cabeça do Robô Educacional, então por meio dessa configuração o Robô iniciará o movimento com a cabeça voltada para frente.

```
1  void setup(){
2
3           Serial.begin(9600);
4
5           stepper.setMaxSpeed(1000);
6
7           stepper2.setMaxSpeed(1000);
8
9           myservo.attach(10);
10
11          myservo.write(90);}
```

Todas as linhas dentro do *loop* são lidas uma após a outra. Para organizar todo o código foram utilizadas algumas funções específicas para cada funcionalidade do Robô. A função *comuSerial()* possibilitou a comunicação do Arduino com o celular via *bluetooth*. O *controleSerial()* é um menu que faz a escolha das ações informadas para o Robô. A função *motorControle_reta()* trabalha com o deslocamento do Robô em linha reta, tanto para frente, quanto para trás. Já o *motorControle_centro()* controla a rotação em grau do Robô em torno de si, em sentido horário ou anti-horário. O *servoMotor()* controla as ações do servo motor.

```
1    void loop(){
2
3        comuSerial();
4
5        controleSerial();
6
7        motorControle\_reta();
8
9        motorControle\_centro();
10
11       servoMotor();}
```

Observe que na função *comuSerial()* o bloco condicional *if()* depende dos dados recebidos e tratados no *while()*. Inicialmente o *while()* avalia se tem dado chegando ou não. Quando a comunicação avaliada é tida como verdadeira entrará dentro do *while()*. O *delay()* pausa o *loop* por dois segundos, em seguida o primeiro caractere de texto é lido pelo *Serial.read()* e armazenado na variável C que é do tipo *char*. A variável *leeCadena* é um vetor do tipo *string*, logo cada caractere recebido em C é colocado em uma posição dentro de *leeCadena*. Quando *leeCadena* é maior que zero a condição é verdadeira e entrará dentro do *if()*. O serial *println(leeCadena.toInt())* converterá o texto armazenado em um número inteiro e apresentará no monitor serial da IDE. Em seguida a variável *codigoSerial* recebe todo valor de *leeCadena.toInt()* já convertido.

```
1  void comuSerial(){
2        while (Serial.available()) {
3         delay(2);
4     char c  = Serial.read();
5     leeCadena += c;}
6        if (leeCadena.length()$>$0){
7         Serial.println(leeCadena.toInt());
8         codigoSerial = leeCadena.toInt();}}
```

O aplicativo instalado no celular enviará dois tipos de códigos: o primeiro será um número com mais de quatro dígitos, esses são para escolher as funcionalidades do Robô. Já os números menores ou iguais a quatro dígitos são ações que ele fará. Tais como se movimentar a determinada distância ou girar em determinado ângulo, segundo a escolha do operador. Quando o número recebido for maior que quatro dígitos, então entrará na condição *if()* e a partir daí o código Serial será avaliado pelo *switch()* e determinará as funções que o robô realizará. O case 10000 indica a direção para frente; case 10001 direção para trás; case 10002 rotação no sentido horário; case 10003 rotação no sentido anti-horário (ver anexo 1).

Se o número recebido tiver a quantidade de dígito menor ou igual a quatro, significa que esse dado não é para selecionar as opções do menu, logo não entrará no *if()* passando assim para o *else{}*. Dessa forma, a variável passo receberá o número em centímetro que está dentro do *codigoSerial*. Em seguida o *codigoSerial* é zerado para processar novos

valores. Note que dentro do *else* o passo também é informado com a distância que o Robô irá percorrer ou o ângulo que ele irá girar:

```
1 else{passo = codigoSerial; codigoSerial = 0; dist = passo; angulo = dist;\}}
```

Observe que os valores recebidos nas variáveis passo, *dist* e *angulo* não estão convertidos para pulsos ou passos que o motor precisa para funcionar. Assim, se a função *motorControle_Centro()* for selecionada no *switch()* a função usará o valor que está guardada dentro da variável angulo e a partir daí será convertido pelas funções (*rotacao* e *pass*). Perceba que nas fórmulas é usado a palavras *long* e a letra L nos números inteiros que estão multiplicando uma variável, fizemos isso para forçar uma conversão mais precisa. De posse do passo já convertido e armazenado em *pass* são usados os objetos *stepper.moveTo* e *stepper2.moveTo* para informar a quantidade de passos que o motor deverá rotacionar. Em *stepper.setSpeed()* são comunicados a direção e a velocidade do motor. O comando *while()* força o motor a executar todos os passos que foram dados. Ao sair do comando de repetição a função *stepper.setCurrentPosition()* resetará todos os passos usados dentro da biblioteca. Em seguida as variáveis usadas são zeradas. Observe que é trabalhado com os dois motores em conjunto. Esse bloco de código utiliza o valor fornecido pelo usuário em grau, transforma em passo e faz o robô girar no ângulo e direção desejados (ver anexo 2).

Essa função controla o movimento para frente ou para atrás do robô através dos valores passados pela variável distância (*dist*). Veja que esse valor ainda não está convertido para passos ou pulsos e por meio da função *pass* esse número será convertido.

Depois que *pass* recebe o total de passos, o *stepper.MoveTo()* informa para o motor a quantidade de pulso que ele precisa dar; por sua vez, as funções *stepper.setSpeed()* informam a direção e a velocidade. O comando de repetição *while()* assegura que o motor dará todos os passos fornecidos. Ao sair do *while()* a função *stepper.setCurrent.Position()* anula todas as entradas na biblioteca para receber novos valores. Em seguida algumas variáveis são reiniciadas.

Note que novamente a função trabalha com os dois motores ao mesmo tempo: o *stepper* e o *stepper2* (ver anexo 3).

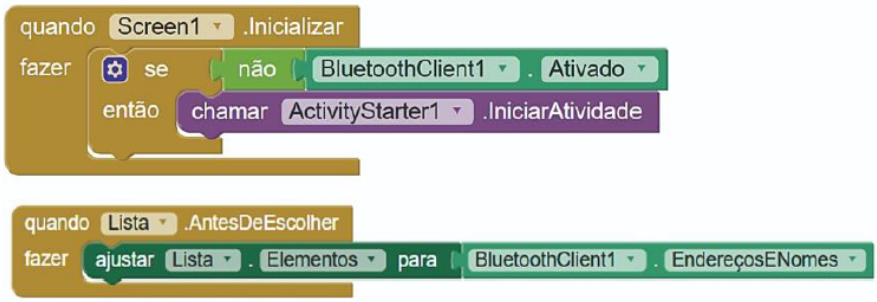
A função *servoMotor()* só funcionará quando a função *motorControle_centro()* estiver em execução, isso pois, dentro dessa função a variável *contat* receberá o valor de 100. Quando *contat* for verdadeira entrará na condição *if()* e em seguida será passada o ângulo de 180º para o servo motor através da função *myservo.write()*, em seguida pausa, e é informado o ângulo de 90º. Ao final do código a variável *contat* é zerada para repetir o processo, caso a função rotação for acionada novamente. Esse movimento do servo motor condiz com o movimento da cabeça do Robô toda vez que ele for rotacionar (ver anexo 4).

O controle remoto é um aplicativo para *smartphone* desenvolvido na plataforma gratuita chamada MIT App Inventor. Esse recurso de desenvolvimento de aplicativos pode ser utilizado por iniciantes em robótica, pois a estrutura de códigos é feita em blocos de encaixe. Uma das vantagens desse site é que ele possui muitos artigos e tutoriais para o aprendizado. Para usar a plataforma basta pesquisar por Mit App Inventor e fazer um cadastro utilizando apenas um e-mail. Ao abrir o site são disponibilizadas todas as ferramentas necessárias para criar um aplicativo. Contudo, o objetivo desse artigo é

explicar a lógica por trás do código, o que por sua vez ampliará o desenvolvimento e aperfeiçoamento do mesmo para futuros projetos.

No primeiro bloco de código, quando o aplicativo é iniciado é verificado se o *bluetooth* está ativo (Figura17), se ele não estiver, então aparecerá uma mensagem na tela pedindo para ligar o *bluetooth* (Figura 17). Por sua vez, antes dele ser ligado, uma lista com todos os dispositivos próximos será apresentada para ser escolhido e pareado.

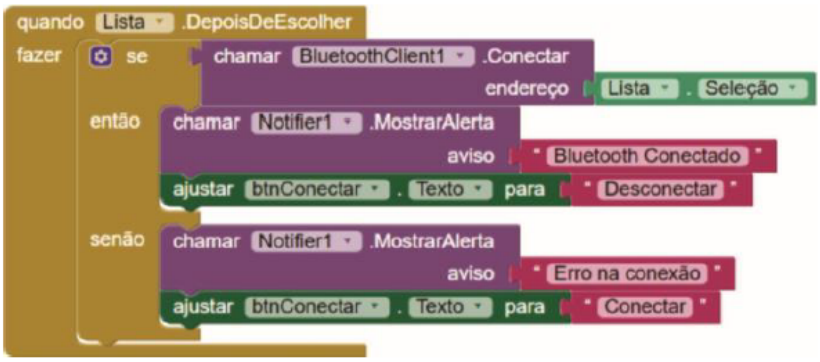
Figura 17. Verificação de ativação do *bluetooth*



Fonte: figura gerada pelos autores no Mit App Inventor

Quando o dispositivo for escolhido pelo usuário, o *bluetooth* tentará conectar e parear, se for conectado, será mostrada na tela do aplicativo (Figura 18) a mensagem que foi conectado. Caso não for conectado, mostrará uma mensagem de alerta ou erro, Figura 18.

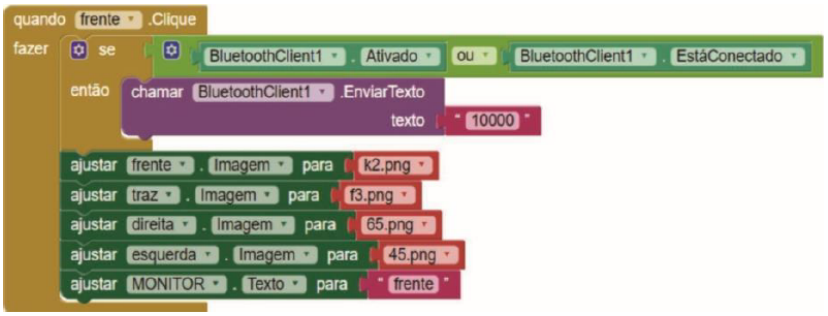
Figura 18. Mensagem de erro de conexão



Fonte: figura gerada pelos autores no Mit App Inventor

Esse bloco de código pertence à funcionalidade dos botões. Quando ele é clicado, verificará se o *bluetooth* está ativo, se estiver ativo, então será enviado um texto para o *bluetooth* que está no Arduino. Observe que esse botão envia a *string* 10000. Quando esse número chegar no Arduino será tratado e verificado que é um código para selecionar uma função do Robô, como por exemplo se movimentar para frente. Dessa forma, esse código (Figura 19) serve para acessar opções do menu.

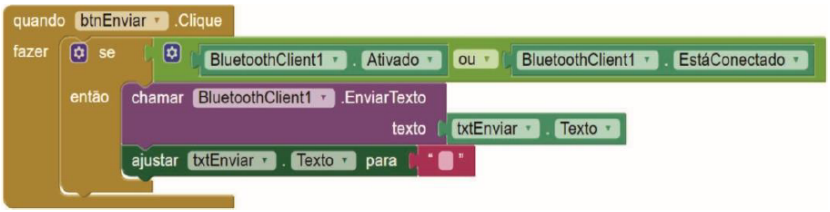
Figura 19. Código de acesso às opções do Menu



Fonte: figura gerada pelos autores no Mit App Inventor

Depois de escolhidas a direção e/ou a rotação que o Robô fará por meio dos botões citados anteriormente, é preciso informar a distância ou o ângulo. Para isso, basta digitar por meio do teclado do celular, na caixa de texto, o comando e clicar no botão enviar. O código verificará se o *bluetooth* está ligado, se estiver, então enviará todo o texto digitado. Números menores que 10000 não são classificados como protocolo para selecionar as opções do menu. Observe, na Figura 19, que esses parâmetros podem ser mudados tranquilamente.

Figura 20. Mudança de parâmetro



Fonte: figura gerada pelos autores no Mit App Inventor

Os códigos aqui apresentados são abertos ao aperfeiçoamento por parte daqueles que desejarem construir o Robô Educacional.

3 Aplicação do Robô Educacional

O Robô Educacional foi utilizado, como recurso didático, em uma investigação cujo objetivo era revisar conteúdos matemáticos dos anos finais do Ensino Fundamental II e foi associado às metodologias de Gamificação e Resolução de problemas. Na ocasião, os pesquisadores utilizaram, além dos dois Robôs Educacionais, um tabuleiro com o aspecto de uma trilha, Figura 20, que deveria ser percorrido pelos Robôs sob o comando de duas equipes de alunos. Foi utilizado ainda um dado com dimensões específicas, como pode ser visualizado na figura 20, também construído com o propósito do experimento, que servia para ditar o número de "passos" a ser percorrido pelos Robôs.

Figura 21. Mudança de parâmetro

Fonte: fotografia realizada pelos autores

Para fazer os Robôs se movimentarem sobre o tabuleiro, os discentes tinham que resolver corretamente, 24 situações problemas de matemática, propostas em lista elaborada pelo professor regente da turma e os pesquisadores. As situações problemas foram resolvidas alternadamente pelas equipes de alunos. Neste experimento foi percebido que as duas equipes não estavam competindo entre si. Os seus objetivos eram fazer os Robôs se movimentarem e chegarem no destino final, independentemente se um estivesse na frente do outro. Isso foi satisfatório, pois resolver as questões de matemática fazia sentido para eles, o desafio de ambas as equipes não era vencer o jogo, e sim, solucionar os problemas matemáticos. Durante o experimento, foi possível observar como o fator lúdico e diferente conseguiu transcender, em algumas ocasiões, a aversão a matemática em prazer, levando os alunos a pensar, raciocinar e interagir entre si, e com o docente para solucionar os problemas propostos.

4 Conclusão

A concepção do Robô Educacional, com materiais de baixo custo, proporcionou aos autores oferecer um recurso didático a profissionais da educação, de modo especial, a professores de matemática, uma forma de instigar os discentes ao aprendizado e propor diversas estratégias de ensino.

Ao Relatar todo o processo de construção do Robô Educacional, incluindo a abertura dos códigos de programação, este trabalho oportuniza que esse processo sirva como fonte de pesquisa e ensino para professores, uma vez que, o Robô Educacional pode ser aperfeiçoado, tanto na parte mecânica com a adição de braços e outros mecanismos de movimento, quanto no âmbito da programação.

Ressaltamos que outras aplicações do Robô Educacional podem ser implementadas de acordo com objetivo do professor, como a adaptação de uma caneta ou lápis na base do robô para a construção de figuras geométricas, sendo dados medida dos lados, ângulos, etc.

Contribuições

Todos os autores contribuíram substancialmente na concepção e/ou no planejamento do estudo; na obtenção, análise e/ou interpretação dos dados; na redação e/ou revisão crítica; e aprovaram a versão final a ser publicada.

Fontes de financiamento

Não há.

Orcid

Carlos Alberto dos Santos Lima https://orcid.org/0009-0004-0276-3244
Cleusiane Vieira Silva https://orcid.org/0000-0002-7156-2276

Referências

1. J. L. S. da Silva, J. R. Evangelista, R. B. dos Santos e P. M. Mendes, "Matemática Lúdica Ensino Fundamental e Médio", *Educação em Foco*, no. 06, pp. 26-36, 2013.
2. F. A. D. Cruz, "O Impacto do Uso de Mídias Tecnológicas (Tecnologia Móvel-Internet) na Qualidade de Vida de Adolescentes", Thesis Dissertation in Educação, Universidade Federal de São Paulo, São Paulo, 2014.
3. S. E. de Souza, "O Uso de Recursos Didáticos no Ensino Escolar". In: *I Encontro de Pesquisa em Educação, IV Jornada de Prática de Ensino, XIII Semana de Pedagogia da UEM: Infância e práticas Educativas*, Maringá, PR, 2007.
4. E. J. Braz and D. P. Vilela, "Robótica Educacional Como Auxilio no Ensino de Matemática: relatos de uma professora", *Revista Ensin@ ufms*, Três Lagoas, vol. 1, no 05, pp. 149-163. 2020.
5. J. B. Sacomano, R. F. Gonçalves, M. T. da Silva, S. H. Bonilla and W. C. Sátyro, *Indústria 4.0: Conceitos e Fundamentos*, 1st ed, São Paulo: Editora Edgard Blücher Ltda. 2018.
6. F. Brito, *Sensores e atuadores*, 1st ed, São Paulo: Editora Érica, 2017.
7. F. P. de Paiva, "Filtro de Kalman aplicado à Localização de Robôs Móveis utilizando o sensor Laser Rangefinder 2D", Trabalho de conclusão de Curso, Engenharia Elétrica, Universidade Federal de Juiz de Fora, Juiz de Fora, 2017.

Editora-científica: Ana Paula Perovano. Orcid iD: https://orcid.org/0000-0002-0893-8082

ANEXO 1

```
void controleSerial(){
 if(leeCadena.length()>4){
switch(codigoSerial){
 case 10000:
 seta = 0;
 direita = 0;
 esquerda = 0;
 servo = 0;
 direcao = -1;
 direcao1 = 0;
codigoSerial = 0;
break
case 10001:
seta = 0;
direita = 0;
esquerda = 0;
servo = 0;
direcao = 1;
direcao1 = 0;
codigoSerial = 0;
break;
case 10002:
seta = 0;
direita = 0;
esquerda = 0;
servo = 0;
direcao  =  0;
direcao1 =  1;
direcao2 = -1;
codigoSerial = 0;
break;
case 10003:
seta = 0;
direita = 0;
esquerda = 0;
servo = 0;
direcao  =  0;
direcao1 = -1;
direcao2 =  1;
codigoSerial = 0;      break;}}
else{ passo = codigoSerial;    codigoSerial = 0; dist = passo;
   angulo = dist; } }
```

ANEXO 2

```
void motorControle_centro() {
      if(  ((direcao1!=0)&&(direcao2!=0)) ){
      if(  ((direcao1!=0)&&(direcao2!=0)) && ( passo > 0) ){
    contat=100;
    rotacao =  long (( ( 51L * (angulo))/360)+1);
    pass =   long ( (4096L * rotacao)/14);
    stepper.moveTo(pass);
    stepper2.moveTo(pass);
    stepper.setSpeed(direcao1 * velocidade);
    stepper2.setSpeed(-direcao2 * velocidade);
     delay(500);
 while ( (direcao1 * stepper.currentPosition() != pass) && (
    direcao2 *   stepper2.currentPosition()!= pass)){
  stepper.runSpeed();
  stepper2.runSpeed();}}
  stepper.setCurrentPosition(0);
  stepper2.setCurrentPosition(0);
  passo = 0;
  pass =  0;
  angulo = 0;}
leeCadena = ""; }
```

ANEXO 3

```
 void motorControle_reta() {
  if(direcao!=0){
              if(  ((direcao == -1) || (direcao == 1 ) ) &&
                 passo > 0 ){
           contat = 100;
       pass =  ( long ( (4096L * 1)/13.8 ) ) * dist;
      stepper.moveTo(pass);
      stepper2.moveTo(pass);
      stepper.setSpeed(direcao * velocidade);
      stepper2.setSpeed(-direcao * velocidade);
      delay(500);
 while ( (direcao * stepper.currentPosition()!= pass) && (
    direcao * stepper2.currentPosition()!= pass)){ stepper.
    runSpeed(); stepper2.runSpeed();}}
       stepper.setCurrentPosition(0);
       stepper2.setCurrentPosition(0);
       passo = 0;
       pass  = 0;}
      leeCadena = ""; }
```

ANEXO 4

```
void servoMotor(){
       if(contat == 100 ){
              myservo.write(180);
              delay(500);
              myservo.write(90);
              delay(500);
              myservo.write(0);
              delay(500);
              myservo.write(90);
              contat = 0;}}
```