UMA METODOLOGIA PARA ROBÔ SEGUIDOR DE LINHA

Vitor dos Santos Silva - 3º ano do Ensino Médio Integrado ao Ensino Técnico em Automação Industrial[1]

Derek Vieira Silva – 1º ano do Ensino Médio Integrado ao Ensino Técnico em Automação Industrial[1]

Guilherme Fortunato Miranda – 1º ano do Ensino Médio Integrado ao Ensino Técnico em Automação Industrial[1]

Nicolas Gabriel Bomfim Souza Santos – 1º ano do Ensino Médio Integrado ao Ensino Técnico em Automação Industrial[1]

Augusto Campos – 1º ano do Ensino Médio Integrado ao Ensino Técnico em Automação Industrial[1]

Pedro Eduardo dos Santos – 2º ano do Ensino Médio Integrado ao Ensino Técnico em Automação Industrial[1]

Tutor: Vera Lúcia da Silva[1], Professores Colaboradores: Wagner Roberto Garo Júnior[1], Masamori Kashiwagi[1] e Raphael Antônio de Souza[1]

verals@ifsp.edu.br, wagner.garo@ifsp.edu.br, masamori@ifsp.edu.br, raphael@ifsp.edu.br

[1] INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO - IFSP
Suzano – SP

Categoria: ARTIGO BÁSICO

Resumo: O robô analisado tem como principal objetivo resolver todos os desafios propostos pela Olimpíada Brasileira de Robótica (OBR). A tarefa hipotética é seguir um caminho com diversas irregularidades, atravessando redutores (lombadas), partes falhas do percurso (*gaps*), virar corretamente nas curvas, desviar de obstáculos, subir rampas, e finalmente, salvar as vítimas debilitadas por um acidente, transportando-as à área de resgate, de forma totalmente autônoma. Desafios como esses, quando resolvidos corretamente, demonstram a crescente capacidade dos robôs de ajudar os humanos em situações de risco, ou mesmo em situações cotidianas. O robô utiliza-se principalmente de dois sensores de cor e um sensor de distância ultrassônico. É desenvolvido com o kit Lego Mindstorms EV3 e programado através da plataforma LeJOS, que possibilita a programação do bloco EV3 na Linguagem Java. O robô, em sua fase de testes, conseguiu resolver corretamente todos os desafios propostos para as salas 1 e 2 da OBR.

Palavras Chaves: Robótica, Linguagem Java, Lego, OBR, Sensor de Cor, Sensor Ultrassônico.

Abstract: *The robot which was analyzed has as its main purpose solving all the challenges proposed by the Brazilian Olympics of Robotics (OBR). The hypothetical task is to take a path with many irregularities, coming across speed bumpers and gaps in the route, turn correctly on curves, dodge obstacles, go up ramps, and finally, save the impaired victims from an accident, transporting them to the rescue area, in a completely automatic way. Challenges like these, when correctly solved, demonstrate the growing capacity of robots to help the humans in risky situations, or even in the daily life. The robot mainly uses two color sensors and an ultrasonic distance sensor. It is built with the Lego Mindstorms EV3 kit and programmed through the interface Lejos, that makes it able to program the EV3 brick with the language Java. The robot, in its test phase, managed to solve correctly the proposed challenges for the rooms 1 and 2 from the OBR.*

Keywords: *Robotics, Java Language, Lego, OBR, Color Sensor, Ultrasonic Sensor.*

1 INTRODUÇÃO

Para o desenvolvimento desse trabalho foi utilizado um robô seguidor de linha desenvolvido pelos alunos. O robô seguidor de linha "é um projeto bem famoso entre os apaixonados por robótica" [Candido, 2018] cujo processo de desenvolvimento não será descrito com detalhes. Nas seções seguintes serão descritos os métodos utilizados para a resolução das tarefas, que foram cuidadosamente selecionados e preparados para atingir um bom desempenho, através de múltiplos testes e experiências com diferentes métodos e lógicas. Os métodos aqui descritos foram os que obtiveram os melhores resultados na fase de teste. A seção 2 apresenta a metodologia utilizada no trabalho em geral. A seção 3 apresenta o método utilizado para seguir a linha. A seção 4 apresenta o método utilizado para fazer curvas. A seção 5 apresenta como é feito o desvio de obstáculos. A seção 6 apresenta como são superados os *gaps*. A seção 7 apresenta como é realizada a rampa. A seção 8 apresenta como é identificada a sala 3.

2 MATERIAIS E MÉTODO

O robô utilizado para os testes e, que possibilitou a avaliação da precisão do método adotado, foi projetado com dois sensores de cor próximos ao chão, cujo retorno no modo de refletância era um valor entre 0 e 1 (0 – Nenhuma reflexão, 1 – Reflexão total). No modo de cor, o sensor retornava um valor inteiro que correspondia a uma cor preestabelecida.

Utilizou-se também um sensor de distância ultrassônico, cujo valor de retorno era em metros.

O controlador utilizado foi EV3, que é "muito mais do que um simples brinquedo. Ele é, na verdade, um kit de iniciação em robótica" [Veras, 2014].

A linguagem de programação utilizada foi Java, que é uma linguagem de programação orientada a objetos [Horstmann, Cornell, 2010]. A orientação a objetos é um recurso muito útil que permite que características do robô sejam manipuladas de maneira mais prática e eficiente.

Foi utilizada para a programação o IDE Eclipse, que permite o desenvolvimento de aplicações Java. No entanto, este ambiente "também pode ser usado para o desenvolvimento de aplicações utilizando outras linguagens de programação tais como PHP, C++, etc." [Vinicius, 2009].

3 SEGUIR A LINHA

Nessa tarefa, o robô tem como objetivo seguir a linha, que consiste em uma faixa preta construída com fita isolante, ao longo de uma placa branca de MDF. Para isso, são utilizados os valores lidos pelos sensores, que são manipulados pela técnica de controle de processos PID (Proporcional Integral Derivativo), e seu valor após essa manipulação é aplicado diretamente nos motores. Com essa técnica, o robô é capaz de seguir uma linha reta, bem como curvas suaves, conforme Figura 1.

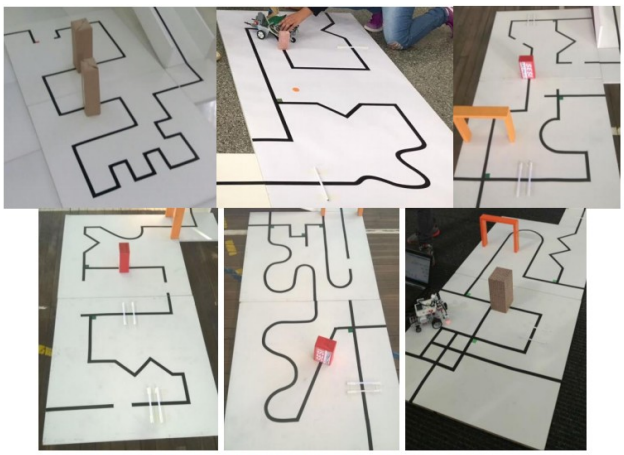

Figura 1 - Pistas

O valor do PID é calculado através da multiplicação do erro no instante atual por um valor constante, acrescido da multiplicação da integral do erro do instante inicial até o atual por um valor constante, acrescido da multiplicação da derivada do erro no instante atual por um valor constante. Valor esse definido pela seguinte fórmula:

$$ValorPID = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)\,dt$$

Onde K_p é a constate de proporcionalidade, (ou ganho), K_i é a constante de integração, K_d é a constante de derivação e e(t) é o valor do erro no instante. O erro é calculado através da subtração do valor do sensor de cor esquerdo, no modo de refletância, do sensor de cor direito, também no modo de refletância.

Após definida uma velocidade inicial dos motores, e realizados os procedimentos pré-código (que não serão abordados nesse artigo), o código entra em um *loop* constante, no qual o valor do PID é calculado, somado à velocidade inicial de um motor, e subtraído do outro.

No programa, esse ajuste é calculado através da multiplicação do erro no instante atual pela constante de proporcionalidade, acrescido do somatório dos erros anteriores multiplicado pela constante de integração, acrescido da diferença do erro no instante atual e do erro no instante anterior multiplicada pela constante de derivação, definido pela seguinte fórmula:

$$ValorPID = K_p e(t) + K_i \sum_0^t e(t) + K_d \left[e(t) - e(t_{-1}) \right]$$

3.1 AS CONSTANTES

Na fórmula anteriormente citada pode-se perceber a presença de três valores constantes, K_p, K_i e K_d. Esses valores foram escolhidos de maneira experimental, observando a resposta do robô aos seus diferentes valores. No início do experimento, as constantes K_i e K_d foram definidas como 0, e foram-se associando valores arbitrários à constante K_p. O código era executado, e observava-se o comportamento do robô com aquele valor definido para K_p. Caso o robô estivesse variando muito na linha, diminuía-se o valor da constante, e caso estivesse demorando muito para corrigir-se, aumentava-se o valor. Após encontrado um valor razoável para K_p, associava-se um valor à K_d, e executava-se o mesmo procedimento que foi utilizado para definir K_p, e por fim, fazia-se o mesmo com K_i.

3.2 AÇÃO *ANTI WIND-UP*

O valor do PID depende de uma integral (ou somatório), e portanto, esse termo da equação pode sobrepor-se aos outros, pois cresce consideravelmente a cada vez que o código é executado, pois este consiste de uma soma, portanto, é necessário aplicar medidas para evitar o chamado "Efeito *Wind-Up*".

Existem múltiplas medidas que podem ser adotadas para não ter o desempenho do robô afetado. Uma delas é a de reiniciar o termo integral a cada *n* vezes que o código for executado, por exemplo, reiniciando-a a cada 100 vezes que o PID for calculado.

3.3 O CÓDIGO

Segue, Figura 2, um exemplo de código do PID para o robô seguidor de linha, utilizando o sensor de cor no modo de refletância (pseudocódigo):

```
Motor motorDireito;
Motor motorEsquerdo;
Sensor sensorDireito;
Sensor sensorEsquerdo;
float Kp = 2000f, Ki = 10.5f, Kd = 2000f; //Valores escolhidos pelo usuário
float P = 0, I = 0, D = 0, ValorPID;
int resetIntegral = 0;
float erro, erroAnterior = 0;

float velocidadeBase = 180; //Graus por segundo
motorDireito.Velocidade = velocidadeBase;
motorEsquerdo.Velocidade = velocidadeBase;

motorDireito.Ativar();
motorEsquerdo.Ativar();

while(true){
    if(resetIntegral>100){
        resetIntegral = 0;
        I = 0;
    }
    erro = SensorDireito.ValorAtual() - SensorEsquerdo.ValorAtual();
    P = erro;
    I += erro;
    D = erro-erroAnterior;
    ValorPID = Kp*P+Ki*I+Kd*D;
    erroAnterior = erro;
    resetIntegral += 1;
    motorEsquerdo.Velocidade = VelocidadeInicial-ValorPID;
    motorDireito.Velocidade = VelocidadeInicial+ValorPID;
}
```

Figura 2 - Código exemplo

Os valores associados para as constantes podem ser alterados pelo usuário, bem como o valor da velocidade inicial dos motores, e a quantidade de vezes que o PID precisa ser calculado para que o termo integral seja reiniciado.

4 CURVAS

O PID pode ser utilizado para seguir a linha e fazer curvas suaves, porém, curvas de 90° e encruzilhadas precisam de sua própria função para serem realizadas.

A Figura 3 ilustra como um robô deve se comportar em cada tipo de curva, indicando os caminhos a serem seguidos.

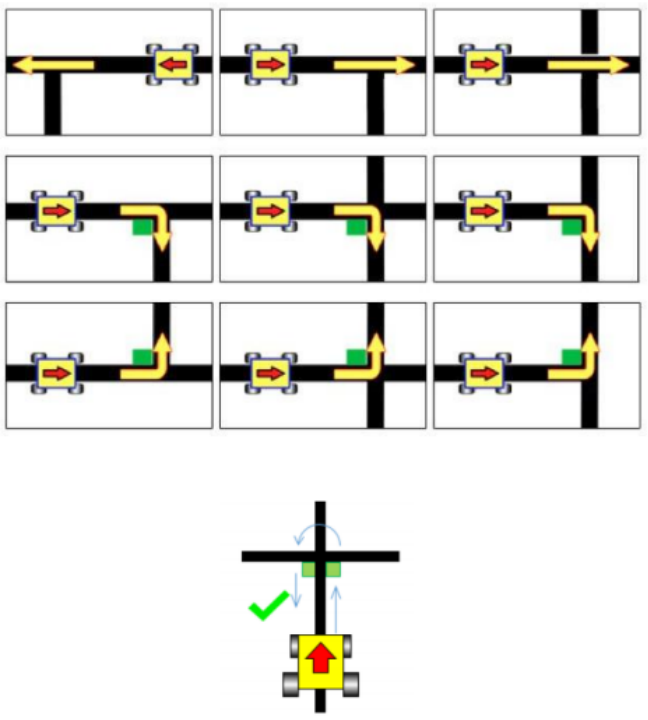

Figura 3 - Curvas (Fonte: OBR 2019)

4.1 IDENTIFICAÇÃO DAS CURVAS

As curvas são denotadas de duas maneiras, ou com uma marcação verde, ou pelo fato de não haver outro caminho para o robô seguir a linha que não seja virar, conforme Figura 4.

Figura 4 - Exemplo de curva

4.1.1 IDENTIFICAÇÃO DAS CURVAS PELO ROBÔ

Levando em consideração que os sensores funcionam no modo de refletância, eles são capazes de detectar a linha preta ou a marcação verde quando a sua refletância diminui, já que quando segue a linha normal seu foco está majoritariamente na placa branca, e uma alteração bruta ocorre quando seu foco muda para uma das fitas.

Ao seguir a linha normalmente, o valor x lido pelo robô deve, em média, ser maior do que sua leitura quando está sobre uma curva, pois o branco, mesmo que parcial, reflete melhor do que o preto ou o verde, portanto, quando a refletância ficar menor do que uma constante REFLETÂNCIA_PRETO, sabe-se que o robô está sobre uma situação não convencional. A constante REFLETÂNCIA_PRETO pode ter valores diferentes para cada sensor, os quais são encontrados através de testes.

O robô para quando uma situação anômala é identificada, e ambas as suas velocidades são definidas com o mesmo valor. Então, caso apenas um dos sensores esteja em situação anômala, chama-se uma função externa, na qual o modo do sensor é trocado de refletância para cor.

A cor é lida, e caso seja verde, o robô deve iniciar a curva. Caso não seja, logo é preto, o robô anda um pouco mais para a frente e efetua outra leitura. Caso leia verde, deve seguir em frente (verde após preto). Caso leia branco, deve checar se há caminho em sua frente, o que pode ser realizado ligando um dos motores por um tempo determinado, de modo que um dos sensores fique completamente sobre onde seria a linha e realize uma leitura. O robô retorna ao seu estado anterior e analisa a leitura. Caso tenha lido preto, segue em frente, mas caso tenha lido branco, inicia o procedimento de curva.

4.1.2 BECO SEM SAÍDA

De um mesmo modo, pode ser criada outra função externa para caso ambos os sensores detectem anomalia. Similarmente, o modo do sensor é trocado para cor. Efetua-se uma leitura, e caso essa leitura resulte verde, o robô inicia o procedimento da volta de 180°, caso seja preto, o robô continua seu caminho.

4.2 PROCEDIMENTOS

Após identificada uma curva o robô deve realizar os procedimentos para contorná-la de forma apropriada.

4.2.1 CURVA 90°

Após identificada a curva o robô deve mover-se para frente o suficiente para que seu centro fique alinhado com a linha para a qual vai virar. Uma vez alinhado, os motores se ligam em sentidos opostos, por exemplo: Caso esteja virando para a direita, o motor direito deve ser ligado no modo reverso, e o esquerdo no modo normal, para que o robô gire em sentido horário. Os motores permanecem ligados até que o sensor do lado para o qual o robô está girando detecte a cor preta, e então, após um pequeno *delay* (para que haja tempo de o robô retirar o sensor de cima da linha e colocá-la entre os sensores), ele para.

4.2.2 CURVA 180º

Caso seja identificado um Beco Sem Saída, o robô deve ligar os motores em sentidos reversos, e continuar até que o sensor do lado para o qual está girando detecte a cor preta, e então há um *delay* para que o robô se alinhe, e os motores são parados.

4.2.3 PROCEDIMENTO PID ZERO

Após realizar o procedimento das curvas o robô retorna a ser controlado pelas ações do PID. Anteriormente foi definida uma velocidade para o PID, de modo que fosse corrigindo sua trajetória enquanto se movimentava. Porém, nesse momento é preciso que ele se encaixe novamente na linha, mas parado onde está. Portanto, define-se o modo do sensor novamente como refletância e utiliza-se o controle pelo PID (não o mesmo de anteriormente, embora possa se utilizar das mesmas constantes) com a velocidade inicial de ambos os motores definida como 0, para que ele se ajuste sem de deslocar, movimentando-se apenas onde se encontra. Após o valor do erro ficar menor do que um valor especificado pelo usuário (o que indicaria que o robô está corretamente sobre a linha), o robô retorna ao seu código principal, para que continue seguindo a linha.

5 OBSTÁCULOS

Há dois tipos de obstáculos, conforme Figura 5, que devem ser levados em conta ao posicionar o sensor de distância ultrassônico. O ideal é que o sensor seja colocado o mais baixo o possível, para poder ser utilizado para encontrar as vítimas na sala 3 (que não será abordada nesse artigo) e os obstáculos.

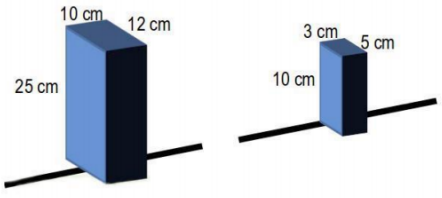

Figura 5 - Obstáculos

É criada uma variável que armazenará a leitura do sensor, e definido um limiar de ativação da função de obstáculo (Ex: 3cm), portanto, quando o valor da leitura do sensor atingir um valor menor do que o limiar, é iniciada a função de obstáculo.

Uma vez identificado o obstáculo, o robô deve parar, virar aproximadamente 90º para a direita (pode-se utilizar a esquerda mediante adaptação), andar para a frente por um tempo (apenas o suficiente para quando virar novamente para a esquerda não estar alinhado com o obstáculo), e então virar para a esquerda, andar para frente (o suficiente para passar o obstáculo), virar-se novamente para a esquerda, andar para frente (de modo que seu centro esteja razoavelmente alinhado com a linha), e então virar-se para a direita novamente, de modo que os sensores fiquem aproximadamente em cima da linha, e então executar o PID ZERO (Citado na subseção 4.2.3).

Os passos para o robô desviar do obstáculo estão ilustrados na Figura 6 abaixo.

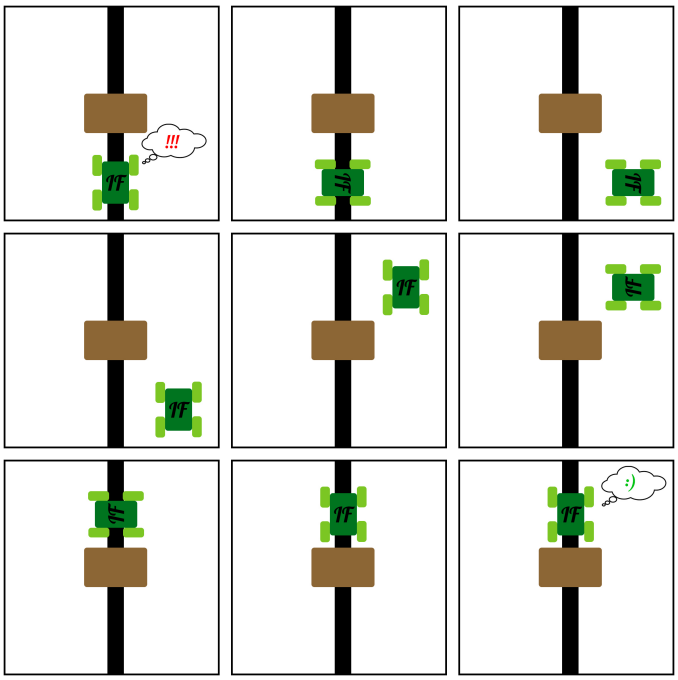

Figura 6 - Desvio de obstáculo

6 GAPS

Os *gaps* consistem em uma pequena parte do circuito pela qual o robô deve seguir reto, sem, no entanto, haver uma linha que o guie e explicite isso. Essa parte do desafio simboliza partes do caminho em que podem haver possíveis falhas. *Gap* vem do inglês e significa “lacuna”.

O procedimento para essa parte do desafio é simples. Caso a refletância dos sensores seja acima do normal (no caso, uma refletância alta, devido à superfície branca), detecta-se que não há linha preta, portanto o robô deve ter a velocidade de ambos os motores definidas como iguais e constantes, fazendo com que o robô ande reto para a frente. Assim que detectado que a refletância voltou a seu valor normal, o robô volta para o código principal (PID seguidor de linha).

7 RAMPA

A rampa é uma parte do percurso que possui uma inclinação de 10 a 20 graus, e que pode causar dificuldades dependendo da montagem estrutural do robô. Como o robô utilizado para esse trabalho não dispunha de acelerômetro para identificar sua inclinação, foi utilizado um método alternativo.

Foi identificado pelo grupo que na rampa não haviam curvas, portanto isso podia ser utilizado para detecção da rampa. O

robô então possui um sistema de contagem, que mede por quanto tempo não houve nenhuma curva. No circuito normal, curvas são muito comuns, e ocorrem com frequência, portanto a única situação em que há uma grande linha reta sem curva alguma é a rampa.

Quando a contagem passa de um certo valor, sabe-se que o robô está em uma linha reta há muito tempo, logo essa linha deve ser a rampa. Vale pontuar, no entanto, que esse sistema está sujeito a falhas, caso a pista possua poucas curvas.

8 DETECÇÃO DE SALA 3

Após detectar a rampa, o robô segue linha apenas, sem a possibilidade de encontrar curvas ou obstáculos, portanto essas funções são desativadas. O robô então segue até uma variação de refletância ser detectada por ambos os sensores, proveniente da fita prateada que se encontra no início da sala 3.

Quando a fita é detectada, o robô anda para a frente por um determinado tempo para que entre na sala 3 por completo, e assim que termina esse tempo, ele para. Em outro artigo será abordada a metodologia para resolver os desafios da sala 3.

9 RESULTADOS

Após diversas análises da performance do robô através de métodos visuais realizadas em laboratório, observou-se que os métodos apresentados nesse trabalho eram muito efetivos, apesar de diversas variações às quais o robô estava suscetível. Revelou-se, no entanto, a necessidade de uma boa calibragem desses métodos para que o robô responda de uma maneira adequada.

10 CONCLUSÕES

Com a produção desse trabalho pôde-se concluir que os desafios propostos pela OBR, embora não sejam resolvidos facilmente, possuem uma solução tangível aos alunos que dispuserem de equipamentos adequados.

A robótica móvel apresenta muitos desafios e existirão muitas soluções para resolvê-los. Este trabalho apresentou uma metodologia para resolver os desafios da OBR, utilizando programação orientada a objetos e Linguagem Java, técnicas de controle PID e a utilização de sensores de cor e refletância.

REFERÊNCIAS BIBLIOGRÁFICAS

Candido, Gradimilo. ROBÔ SEGUIDOR DE LINHA COM SENSOR INFRAVERMELHO E PWM. 2018. Vida de Silício. Disponível em: <https://portal.vidadesilicio.com.br/robo-seguidor-de-linha-sensor-infravermelho-e-pwm/>. Acesso em: 23 jul. 2019.

Horstmann, C. S.; Cornell, G. Core Java: fundamentos. 8. ed. São Paulo: Person Prentice Hall, 2010. v. 1

Olimpíada Brasileira de Robótica. Manual de Regras e Instruções Etapa Regional / Estadual. 2019. Disponível em: <http://www.obr.org.br/manuais/OBR2019_MP_ManualRegionalEstadual.pdf>. Acesso em: 04 jul. 2019.

Veras, Leonardo. LEGO MINDSTORMS EV3. 2014. EXAME. Disponível em: <https://exame.abril.com.br/tecnologia/lego-mindstorms-ev3/>. Acesso em: 23 jul. 2019.

Vinicius, Caio. Trabalhando com a IDE Eclipse. 2009. Oficina da Net. Disponível em: <https://www.oficinadanet.com.br/artigo/1571/trabalhando_com_a_ide_eclipse>. Acesso em: 23 jul. 2019.

Observação: O material multimídia deste trabalho encontra-se disponível em: www.mnr.org.br/mostravirtual.