

Apostila

ROBÓTICA (ROBÔ SEGUIDOR DE LINHA)

José Marcelo Traina Chacon

Sérgio Luiz Moral Marques

2024

Robótica	3
Para que serve a Robótica	4
Robô seguidor de linha.....	4
Simulador sBotics	5
SBotics Instalação	5
Escolhendo o robô.....	7
Escolhendo a Arena.....	9
Movimentação na arena sBotics	10
Área de Programação	10
Funções do R-Educ	12
Primeiro programa	14
Comandos básicos	15
Exercício 1.....	19
Comandos básicos de R-Educ	20
Principais funções.....	24
Carregar Arena	25
Robô seguidor de linha.....	27
Exercício 2.....	29
Exercício 3.....	29
Referência Bibliográfica	30

Robótica

O que é robótica?

Robótica é a ciência que estuda as tecnologias associadas a concepção e construção de robôs. Os robôs são mecanismos automáticos que utilizam de circuitos integrados para realizarem atividades e movimentos humanos simples ou complexos. A robótica tem grande aplicação em diversas áreas desde a produção industrial, medicina até atividades domésticas.

O conceito sobre robótica surgiu no início do século XX, na obra “O Mentiroso” do autor de ficção científica Isaac Asimov,

Foi o autor quem criou a palavra ‘robótica’ e foi também ele quem apresentou, no campo da ficção-científica, as Leis da Robótica.

O termo foi popularizado apenas em 1950 por conta do livro “Eu, Robô”, do mesmo autor. A obra levantou diversas discussões sobre a relação entre homens e máquinas.

A ideia da criação de máquinas é antiga. Na Grécia Antiga, acreditava-se que os gregos e romanos já desenvolviam diferentes tipos de máquinas capazes de realizar movimentos automatizados.

O que é um robô?

Os Robôs são mecanismos automáticos que realizam trabalhos e movimentos humanos. São controlados por humanos e providos de sistemas eletrônicos.

A palavra robô apareceu pela primeira vez em um teatro, na peça “Robôs Teatrais de Rossum”, do tcheco Karel Čapek. Isso foi há 100 anos, mas já na naquela época, o robô era símbolo de desenvolvimento tecnológico e grandeza.

Para que serve a Robótica

A robótica tem grande aplicação em diversas áreas desde a produção industrial, até atividades domésticas. Desde a Primeira Revolução Industrial, robôs e outros equipamentos são utilizados para aumentar a produtividade das empresas.

Mas ao contrário do que pensam, a robótica não existe apenas para uso de grandes empresas.

Tarefas cotidianas como limpar a casa, por exemplo, se tornaram menos exaustivas com a ajuda de novas tecnologias e dispositivos que realizam atividades automatizadas.

Robô seguidor de linha

Robôs seguidores de linha basicamente são robôs pré-programados cuja função é, através da leitura de sensores, detectar onde existe um caminho em uma superfície e seguir este caminho.



Simulador sBotics

O sBotics é uma plataforma de simulação dos níveis 1 e 2 da prova prática estadual da Olímpiada Brasileira de Robótica. Na abordagem comum desta prova utilizase kits e robótica para simular o resgate de uma vítima em um ambiente de desastre. O sBotics oferece uma alternativa para aqueles que desejam testar seus conhecimentos de robótica e programação.



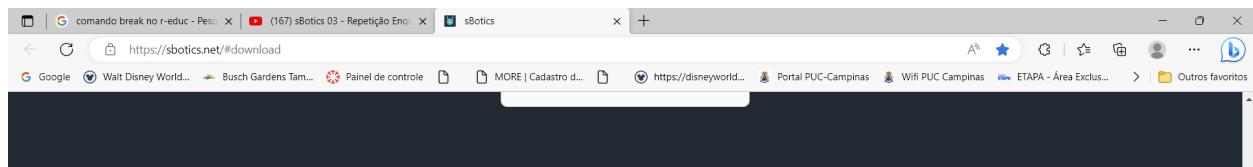
SBotics Instalação

Instalação

Entre no site: <https://sbotics.net/#download>

Clique em Baixar sBotics

Escolha o seu sistema operacional para instalação



Baixar o Simulador sBotics

Baixe aqui o Inicializador (Launcher) sBotics, que é utilizado para automaticamente pesquisar e atualizar o Simulador sBotics.



Windows

Linux

macOS

[Baixar para Windows 32](#)

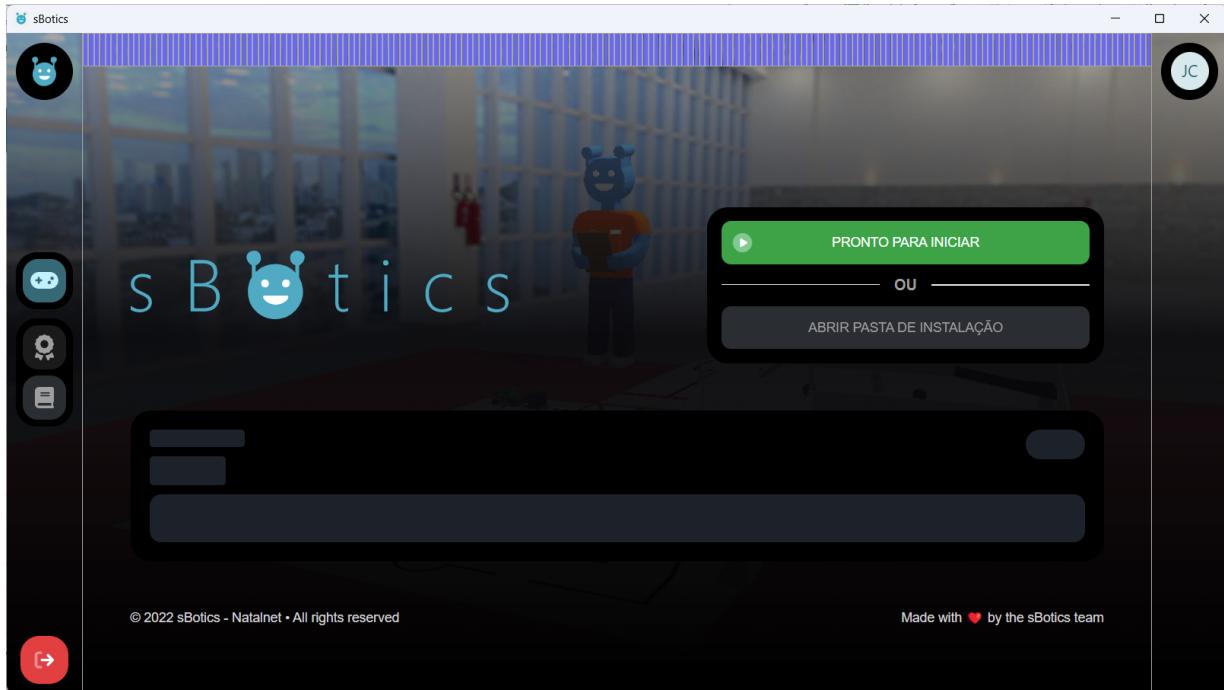
[Baixar para Windows 64](#)

Ao baixar sBotics você concorda com nossos [Termos de Serviço](#).

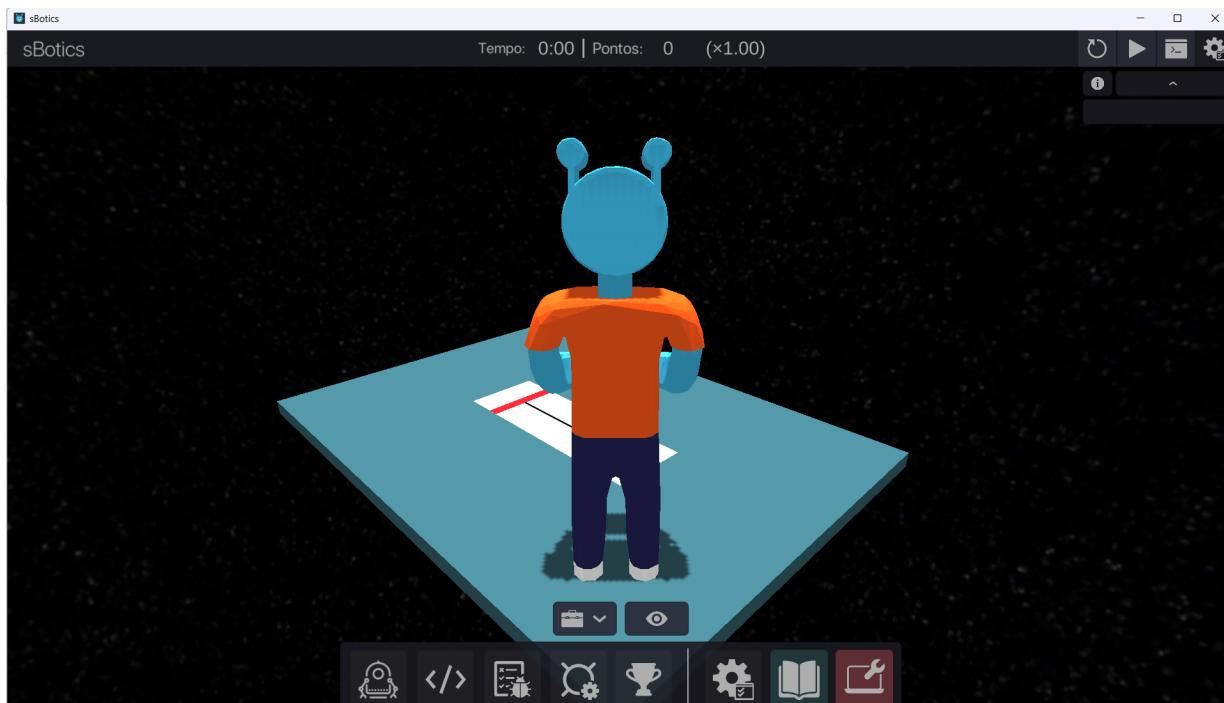
Crie uma conta com o seu email particular e salve uma senha para acesso.

Após o cadastro será inicializado a tela de abertura do Sbotics

Clique em Pronto para iniciar

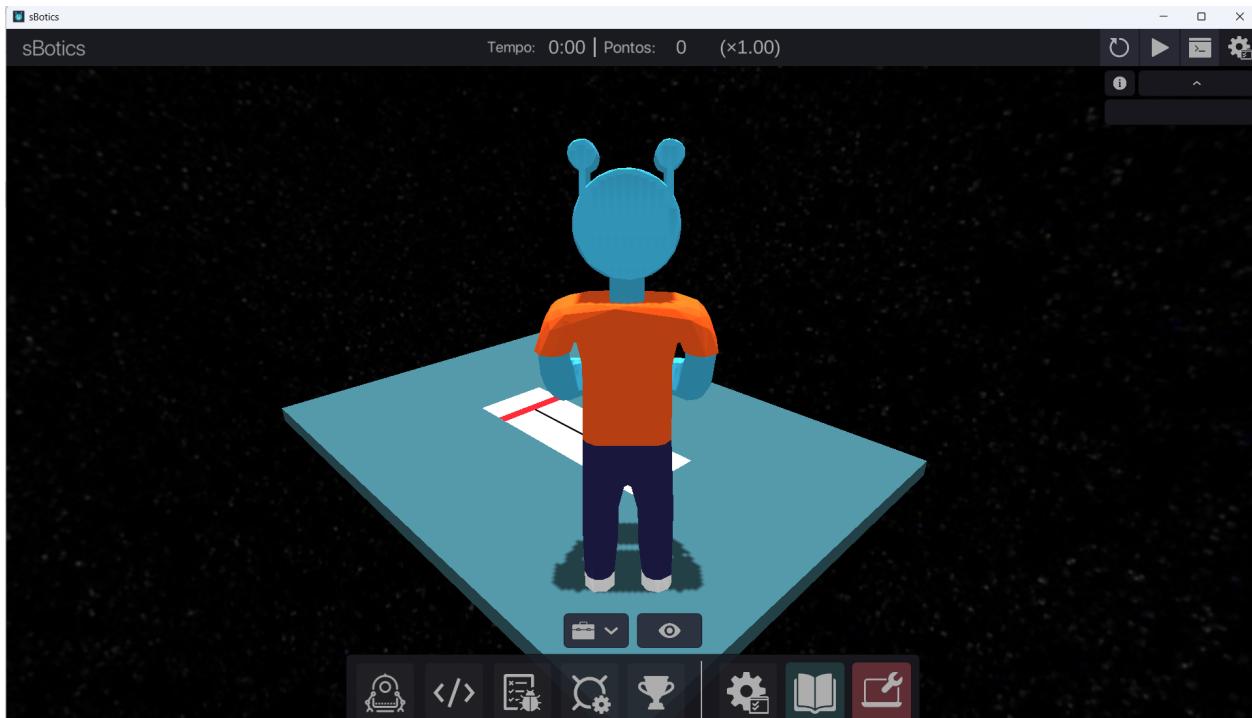


A tela do Simulador aparecerá



Escolhendo o robô

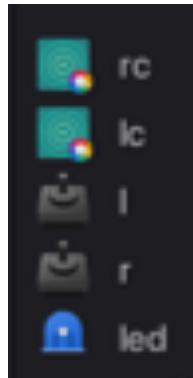
Clique em robôs



1. Tipos de robôs pré-definidos
2. Robô escolhido para a aula

Este robô possui:

3. 1 led
4. 2 sensores de luz / cor
5. 2 servomotores
6. Nomes dos componentes do robô escolhido:



- Sensor de cor direito
- Sensor de cor esquerdo
- Servomotor esquerdo
- Servomotor direito
- led

1

2

3

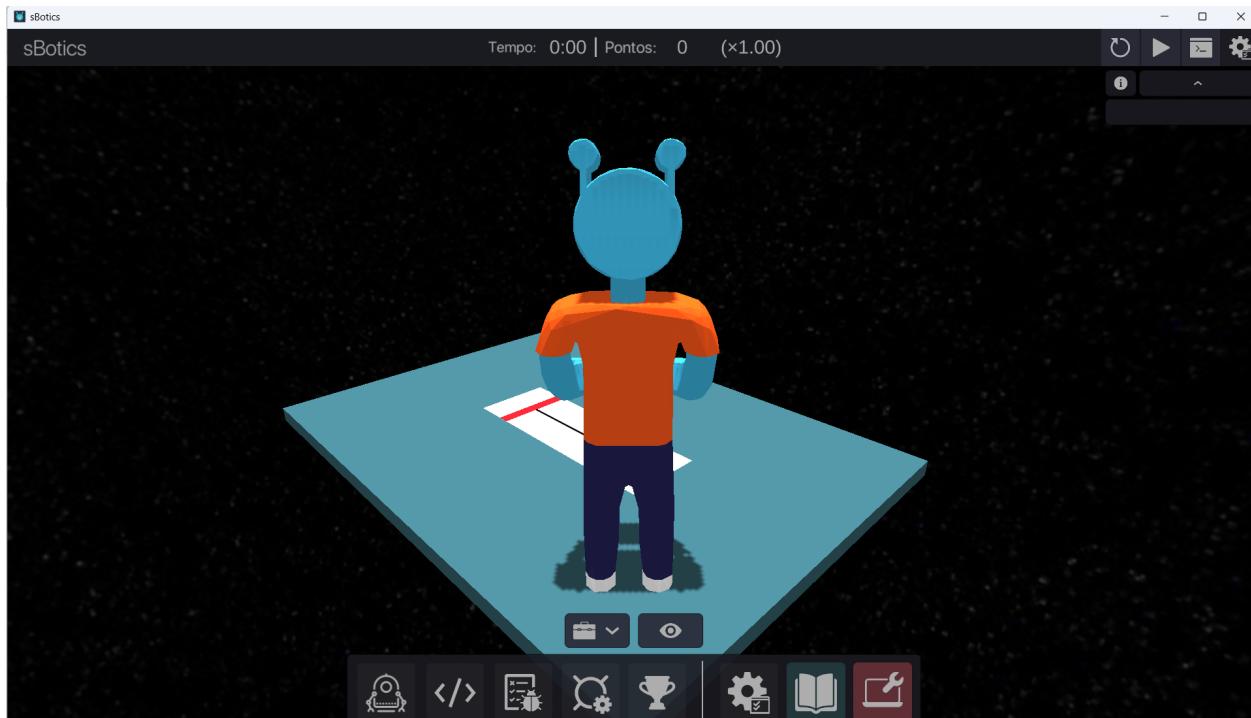
4

5

6

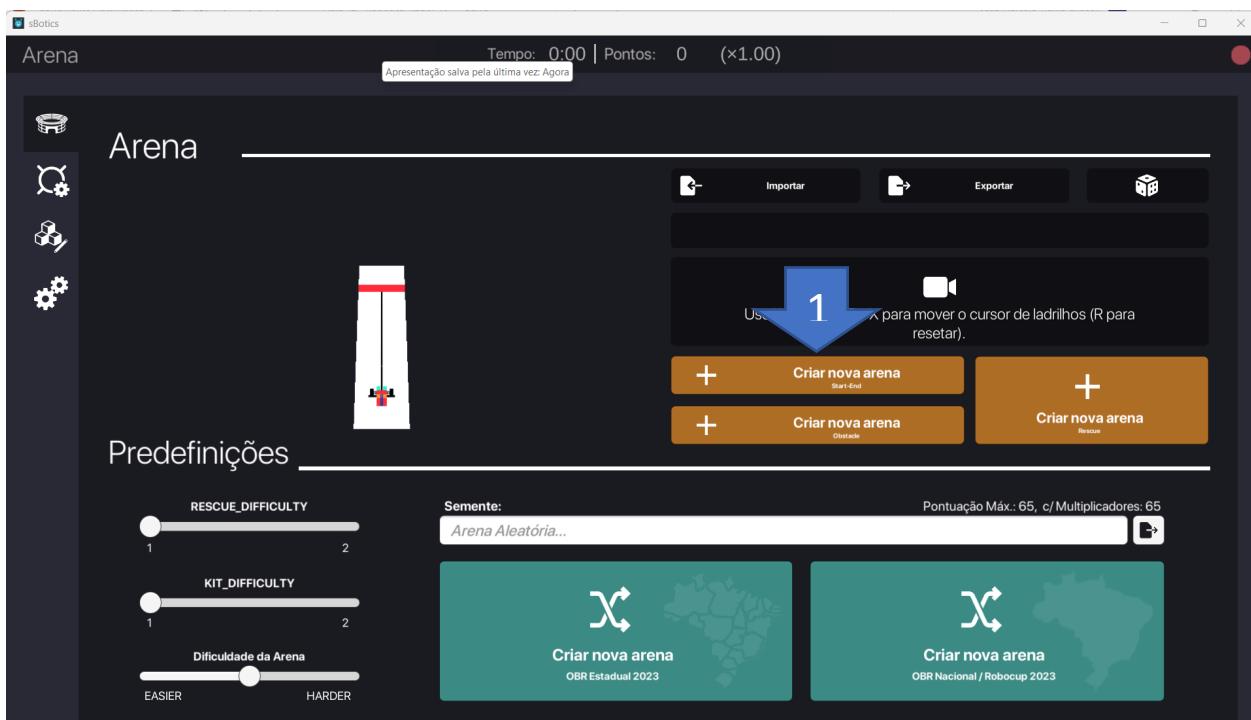
Escolhendo a Arena

Clique em Arena



1. Selecione Criar Nova Arena Start-End

Pressione a tecla ESC para retornar a Arena



Movimentação na arena sBotics

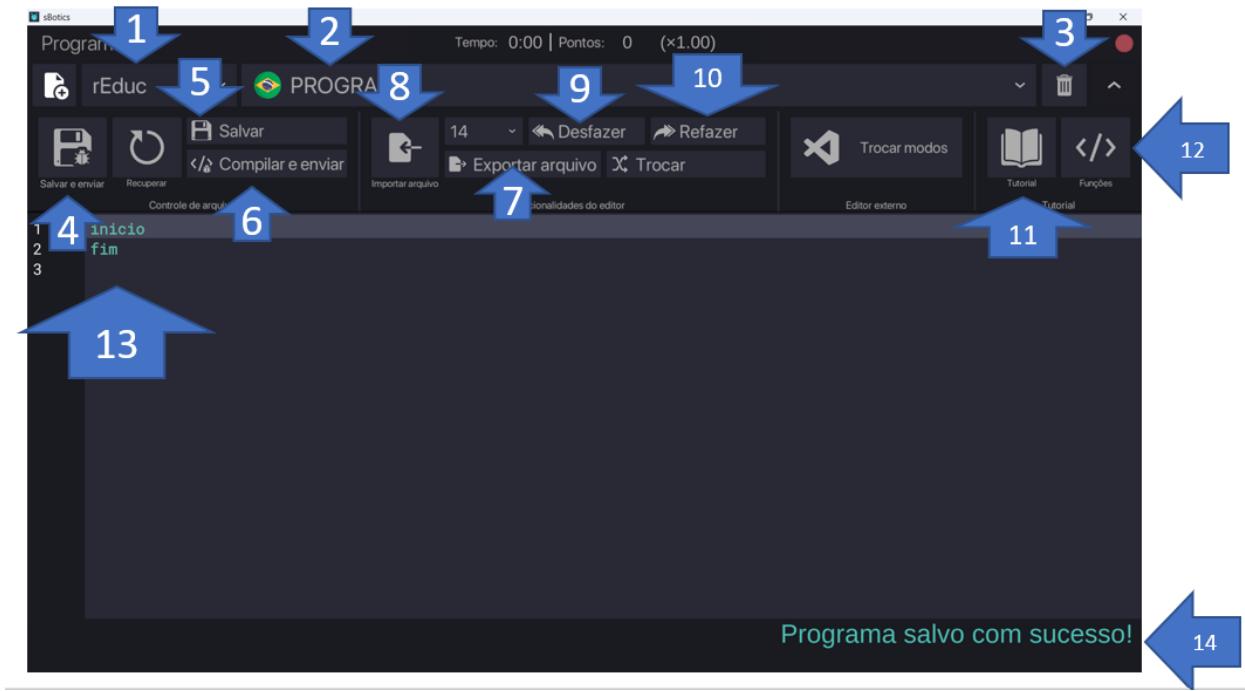
- Q – Vira para esquerda ↡
- E – Vira para direita ↠
- W – Aproxima 🔍
- S – Afasta 🔎
- A – Move para a esquerda ←
- D – Move para a direita →
- Z – Sobe ↑
- X – Desce ↓
- C – Subir o olhar ↗
- V – Baixar o olhar ↘
- R – Aproxima do robô 🤖

Área de Programação

1. Clique em programação



1. Linguagem escolhida para programar o sBotics
2. Arquivo de Programa carregado no Editor
3. Excluir arquivo
4. Salvar e enviar arquivo para o robô
5. Salvar o arquivo
6. Compilar o arquivo e enviar para o robô
7. Exportar arquivo de programa
8. Importar programa externo
9. Desfazer
10. Refazer
11. Tutorial
12. Funções
13. Área de programação
14. Mensagens



Funções do R-Educ

1. Clique em Tutorial



A screenshot of a web browser displaying the 'Introdução' page of the sBotics Docs website. The URL in the address bar is https://docs.sbotics.net/sbotics/sBotics_Simulator/introduction. The page content includes a warning message about the text being written in C# and a note about method names. On the right side, there's a sidebar with tabs for 'sBotics Simulator' (selected), 'Funções' (highlighted with a blue arrow labeled '1'), and other options like 'Mais' and 'Ferramentas'.

sBotics Simulator

- Introdução
- Instalação

sBotics na Robótica

- Criação de Robôs
- Sistema de Componentes
- Classe de Componentes
- Classes de Apoio

Mudanças

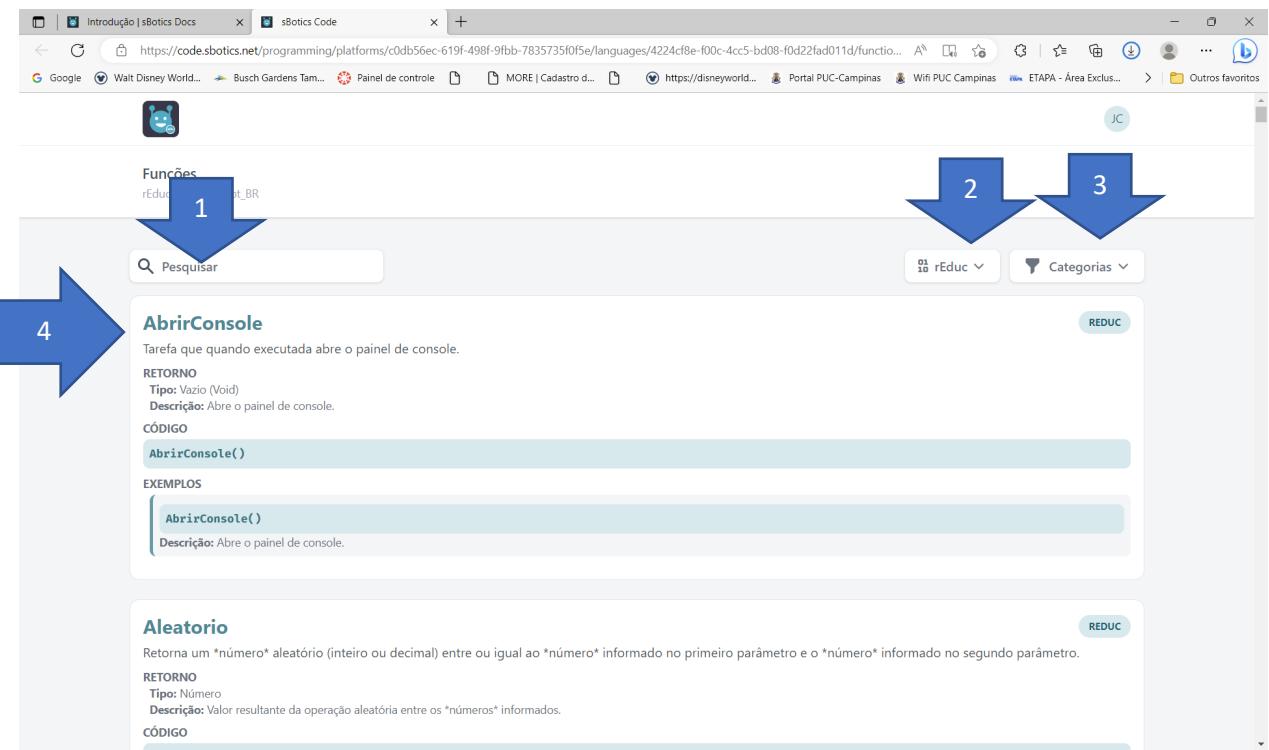
- Programação
- Visão do sBotics

Introdução

Aviso, esse texto foi escrito pensando em C#. Alguns dos métodos podem possuir nomes diferentes em rEduc/BlockEduc.

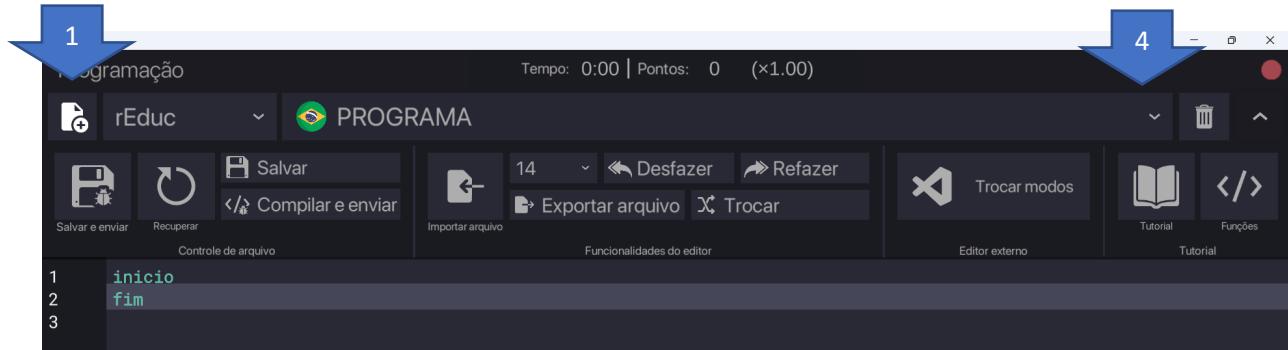
Nós do sBotics estamos reprojetando o funcionamento dos robôs para um sistema baseado em componentes e de forma síncrona, fazendo com que a programação fique mais lógica e concisa. Antes, comandos na programação poderiam interagir com tudo, desde a movimentação do robô até a realização de cálculos. Ora, qual o por quê do robô fazer uma operação "modulo" (resto de divisão, %) quando você realizava `bot.Modulo(5, 10);` e não uma biblioteca matemática? Agora, estamos separando tudo, fazendo com que as coisas tenham mais lógica e sejam mais intuitivas. Classe do robô apenas interage com robô, classe tempo com o tempo e assim por diante. Métodos que realizavam implementações complexas ao aluno, removidas.

1. Pesquisar um comando
2. Escolher a linguagem
3. Selecionar uma categoria
4. Sintaxe e explicação da função



Primeiro programa

1. Clique em novo programa



2. Digite o nome do programa: Aula1

3. Clique em novo



4. Escolha o programa **Aula1**

Comandos básicos

AbrirConsole

Tarefa que quando executada abre o painel de console.

EscreverLinha

Tarefa que adiciona uma linha no console com o *texto* informado no primeiro parâmetro.

Esperar

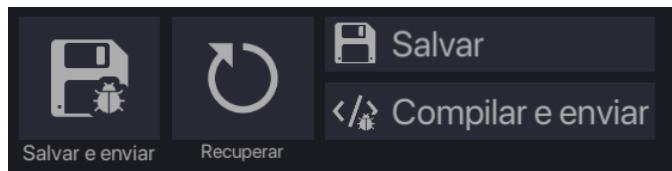
Tarefa que pausa a execução do programa e espera um determinado tempo para voltar, representado pelo *número* no primeiro parâmetro em milissegundos.

Digite o programa abaixo

```
inicio
    AbrirConsole()
    EscreverLinha("Início")
    Esperar(500)
fim
```

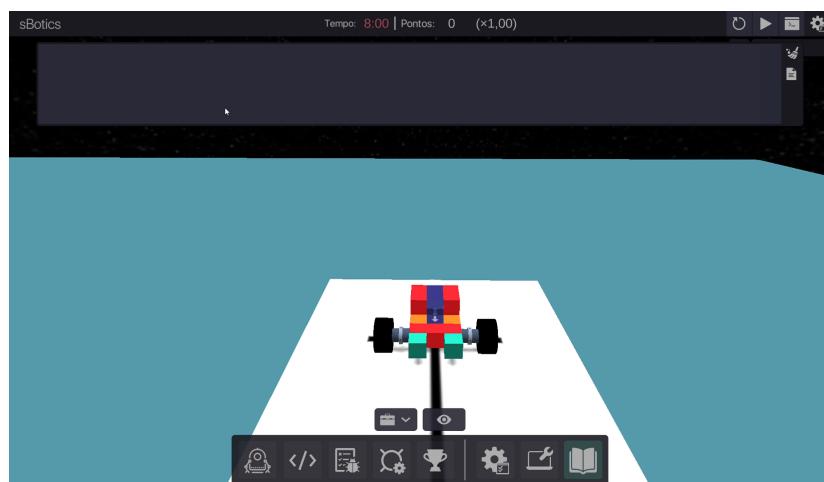
Clique em salvar e enviar

Clique em Compilar e enviar



Aperte a tecla ESC para retornar a Arena

Execute o programa carregado no Robô



Vamos escrever uma função para ligar o led em cor branca

DesligarLuz

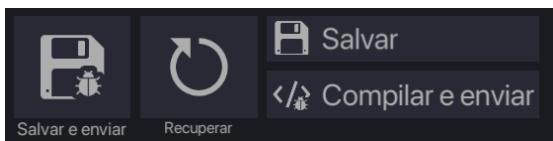
Tarefa que desliga o componente de luz de nome informado pelo *texto* no primeiro parâmetro.

Digite o código abaixo

```
1  tarefa ligarledbranco{
2      EscreverLinha("Ligar led branco")
3      LigarLuz("led", 255, 255, 255)
4      Esperar(500)
5  }
6
7  inicio
8      AbrirConsole()
9      EscreverLinha("Início")
10     Esperar(500)
11
12     ligarledbranco()
13 fim
```

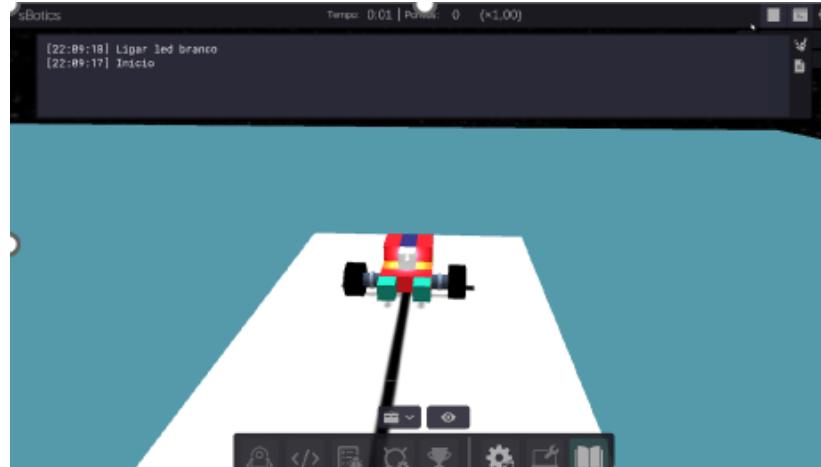
Clique em salvar e enviar

Clique em Compilar e enviar



Aperte a tecla ESC para retornar a Arena

Execute o programa carregado no Robô



Função para virar a direita

TravarMotor

Tarefa que trava ou destrava pelo *booleano* no segundo parâmetro a rotação livre do motor de nome informado pelo *texto* no primeiro parâmetro.

Motor

Tarefa que aplica uma força e velocidade do *número* informado pelo segundo parâmetro no motor de nome informado *pelo* texto no primeiro parâmetro.

Escrever a função

```
tarefa direita{
    TravarMotor("r",falso)
    Motor("r",1000)
}
```

Chamar a função no programa principal

```
EscreverLinha("Virar para a direita")
direita()
Esperar(3000)
```

Clique em salvar e enviar

Clique em Compilar e enviar



Aperte a tecla ESC para retornar a Arena

Execute o programa carregado no Robô



Observação o robô não para de virar para a direita após 3000 milissegundos

Desenvolva a função para virar à esquerda

Função para ir para frente

```
tarefa frente{
    TravarMotor("r",falso)
    TravarMotor("l",falso)
    Motor("r",0+1000)
    Motor("l",0+1000)
}
```

Desenvolva a função para atras

Função para parar

```
tarefa parar{
    TravarMotor("r",verdadeiro)
    TravarMotor("l",verdadeiro)
}
```

Exercício 1

Desenvolva o programa principal para:

- Parar o robô
- Ir para a esquerda por 3000 milissegundos
- Ir para frente por 6000 milissegundos
- Ir para atras por 2000 milissegundos
- Desligar o led

```
EscreverLinha("Desligar led")
DesligarLuz("led")
Esperar(500)
```

- Mensagem de fim
- Clique em salvar e enviar
- Clique em Compilar e enviar



Aperte a tecla ESC para retornar a Arena

Comandos básicos de R-Educ

Operadores Lógicos e Relacionais

A linguagem Reduc possui os seguintes operadores lógicos e relacionais:

e	→ e lógico
ou	→ ou lógico
!	→ não lógico
==	→ igual
!=	→ diferente
>	→ maior que
<	→ menor que
>=	→ maior ou igual que
<=	→ menor ou igual que

Controladores de fluxo

A linguagem R-Educ possui as seguintes estruturas controladoras de fluxo: se, enquanto, farei, repita, teste, para e sair. A seguir apresentaremos como deve ser escrita cada uma dessas estruturas.

Se

A estrutura se da linguagem R-Educ representa o comando de seleção se-então e se-senão-então presente em todas as linguagens de programação. Esta estrutura ao ser utilizada permite selecionar quais os comandos que serão executados caso uma determinada condição seja satisfeita.

se (condicao) entao{

comandos

} senao {

outros comandos

}

Enquanto

A estrutura enquanto da linguagem R-Educ representa um comando de repetição condicional. A ideia básica é fazer com que a execução de uma série de comandos seja repetida enquanto uma determinada condição for satisfeita. Quando a condição não for mais satisfeita a repetição é finalizada.

```
enquanto (condicao) farei {
```

```
    comandos
```

```
}
```

Farei

A estrutura farei da linguagem R-Educ representa um comando de repetição condicional assim como o enquanto apresentado anteriormente. A única diferença entre esta estrutura e a anterior é que nesta primeira sequência de comandos é realizada e, em seguida, a condição é avaliada, caso a condição seja satisfeita os comandos são novamente executados.

```
farei {
```

```
    comandos
```

```
} enquanto (condicao)
```

Repita

A estrutura repita da linguagem R-Educ representa um comando de repetição contada. A utilização desta estrutura faz com que um conjunto de comandos seja executado determinado número de vezes. Abaixo apresentamos como deve ser utilizada esta estrutura para que um conjunto de comando seja executado 5 vezes.

```
repita 5 vezes {
```

```
    comandos
```

```
}
```

Teste

A estrutura teste da linguagem R-Educ representa um comando de seleção. A utilização desta estrutura faz com que o valor de uma variável do tipo texto fornecida seja avaliado e dependendo de seu valor uma determinada sequência de comandos é realizada. No código abaixo a variável testada foi chamada de variavel e caso o seu valor seja igual a valor1 a sequência de comandos denominada no código como comandos1 será executada, caso seu valor seja igual a valor2 o conjunto comandos2 será executado. Caso seu valor não seja nenhum dos apresentados a sequência de comandos denominada de comandos3 será executada.

```
teste ( variavel ) {
    caso valor1: comandos1
    caso valor2: comandos2
    outros: comandos3
}
```

Para

A estrutura para da linguagem R-Educ representa um comando de repetição contada. A utilização desta estrutura faz com que um conjunto de comandos seja executado determinado número de vezes. A quantidade de vezes que a sequência de comandos é realizada vai depender do valor da variável de controle que vai assumir valores desde um limite superior até um limite inferior, percorrendo-os de acordo com um determinado passo informado na estrutura. No código abaixo a variável de controle foi chamada de variavel, o limite inferior e superior de valor1 e valor2 respectivamente e o passo de x.

```
para variavel de valor1 ate valor2 passo x farei {
    comandos
}
```

Para utilizar essa estrutura é necessário definir a variável utilizada previamente.

Interromper

Interrompe a execução de uma repetição.

Interromper()

Tarefas

Um programa escrito em R-Educ pode conter tarefas, conjunto de comandos nomeados que também são chamados de funções, procedimentos ou sub-rotinas. Para que esta estrutura seja utilizada elas devem ser declaradas antes do início do programa. A declaração de uma tarefa com nome "funcao" por exemplo, é feita como segue:

```
tarefa funcao{
```

```
    comandos
```

```
}
```

Ao declarar uma tarefa a sequência de comandos contida nela não será executada. Para que a execução de seus comandos seja realizada é necessário que o seu nome seja chamado. Para que a sequência de comandos inserida na tarefa função seja executada, por exemplo, é necessário que seja digitado o seguinte comando:

```
funcao()
```

Funções

A linguagem R-Educ é uma linguagem dinâmica. Dessa forma inúmeras funções podem ser cadastradas, estas podem requisitar ou não parâmetros, possuir ou não retornos. O cadastro de funções permite que operações específicas do dispositivo robótico em uso seja efetuada e outras estruturas presentes em outras linguagens de programação sejam realizadas. As funções cadastradas e suas

formas de operação podem ser visualizadas diretamente no ambiente de programação.

Principais funções

Led

DesligarLuz

Tarefa que desliga o componente de luz de nome informado pelo *texto* no primeiro parâmetro.

LigarLuz

Tarefa que liga o componente de luz de nome informado pelo *texto* no primeiro parâmetro, nas cores vermelho, verde e azul passadas respectivamente pelos *números* no segundo, terceiro e quarto parâmetros.

Leitura / cor

Colorido

Retorna um *booleano* que representa se a cor lida pelo sensor de cor de nome informado pelo *texto* no primeiro parâmetro é diferente de preto.

Cor

Retorna um *texto* que representa o nome da cor lida pelo sensor de cor de nome informado pelo *texto* no primeiro parâmetro.

CorAzul

Retorna um *número* que representa a tonalidade de azul lida pelo sensor de cor de nome informado pelo *texto* no primeiro parâmetro.

CorVerde

Retorna um *número* que representa a tonalidade de verde lida pelo sensor de cor de nome informado pelo *texto* no primeiro parâmetro.

CorVermelho

Retorna um *número* que representa a tonalidade de vermelho lida pelo sensor de cor de nome informado pelo *texto* no primeiro parâmetro.

Movimentação / Básico

Motor

Tarefa que aplica uma força e velocidade do *número* informado pelo segundo parâmetro no motor de nome informado *pelo* texto no primeiro parâmetro.

TravarMotor

Tarefa que trava ou destrava pelo *booleano* no segundo parâmetro a rotação livre do motor de nome informado pelo *texto* no primeiro parâmetro.

Som

EstaTocando

Retorna um *booleano* que representa o estado de funcionamento do componente buzzer de nome informado pelo *texto* no primeiro parâmetro.

PararSom

Tarefa que para o som do componente buzzer de nome informado pelo *texto* no primeiro parâmetro.

TocarFrequencia

Tarefa que toca uma frequência em hertz informada pelo *número* no segundo parâmetro no componente buzzer de nome informado pelo *texto* no primeiro parâmetro.

TocarNota

Tarefa que toca uma nota musical informada pelo *texto* no segundo parâmetro no componente buzzer de nome informado pelo *texto* no primeiro parâmetro.

Tempo

Esperar

Tarefa que pausa a execução do programa e espera um determinado tempo para voltar, representado pelo *número* no primeiro parâmetro em milissegundos.

Texto / Console

AbrirConsole

Tarefa que quando executada abre o painel de console.

Escrever

Tarefa que define o conteúdo no console para o *texto* informado no primeiro parâmetro.

EscreverLinha

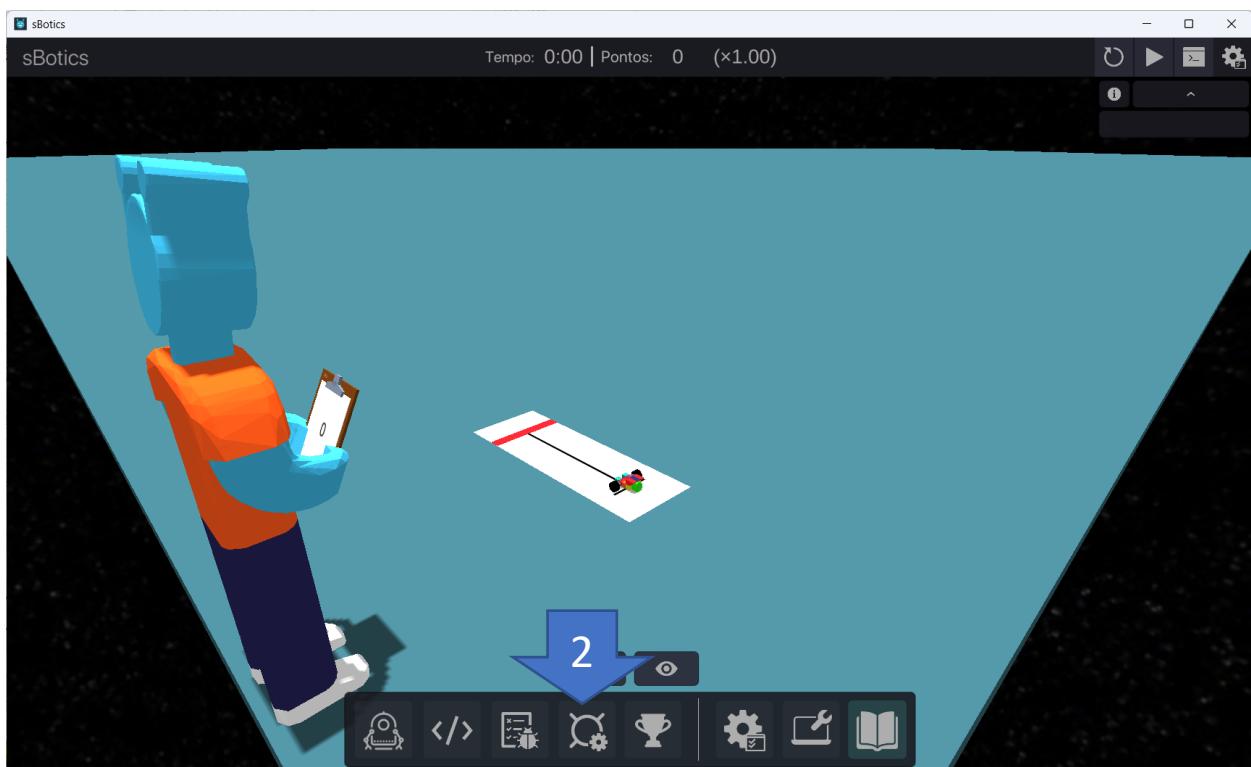
Tarefa que adiciona uma linha no console com o *texto* informado no primeiro parâmetro.

LimparConsole

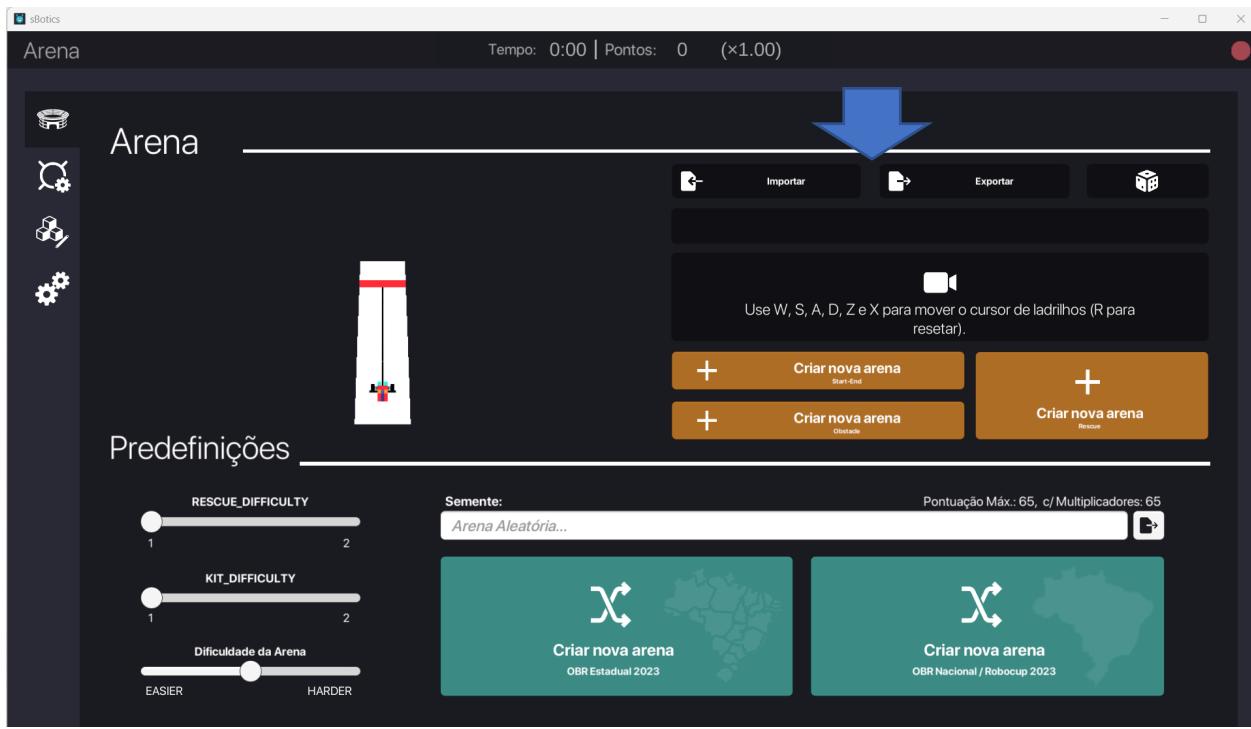
Tarefa que quando executada limpa o painel de console.

Carregar Arena

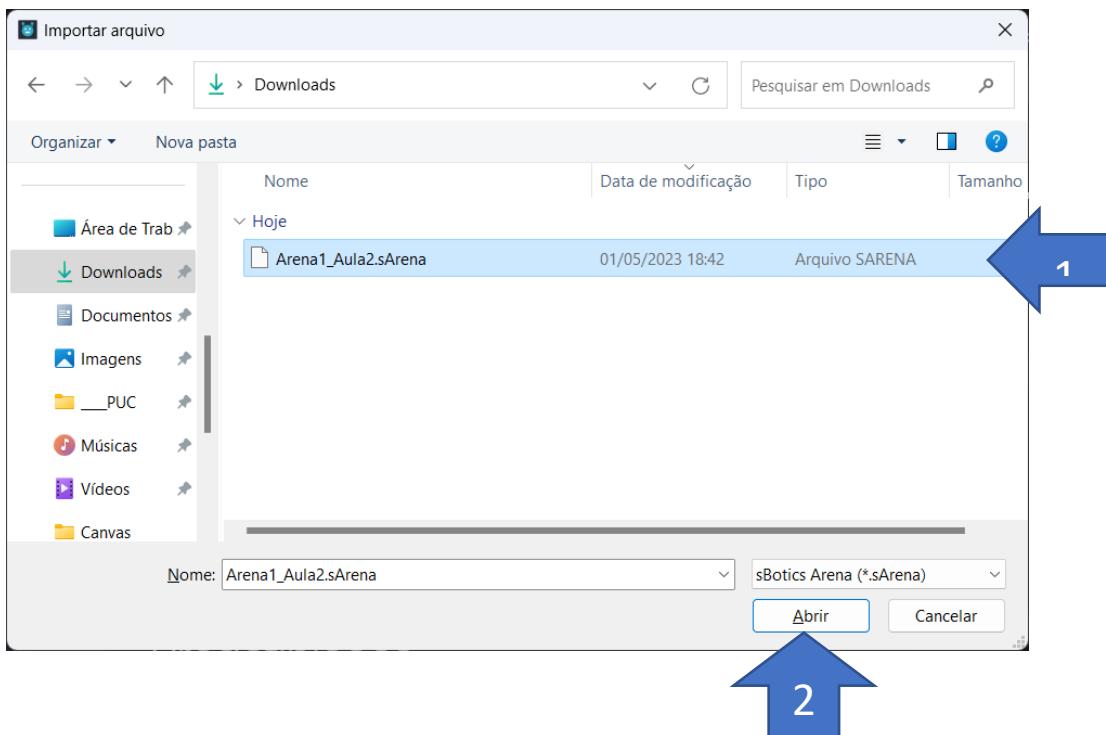
1. Baixe o arquivo minha **Arena1_Aula2.sArena** do Canvas
2. Clique em arena



Clique em Importar



1. Selecione o arquivo importado do Canvas
2. Clique em Abrir
3. Pressione ESC



Robô seguidor de linha

Teste o programa seguidor de linha abaixo

MELHORIAS: Implemente as curvas a direita

```
tarefa parar{  
    TravarMotor("r",verdadeiro)  
    TravarMotor("l",verdadeiro)  
}  
  
tarefa esquerda com numero velocidade {  
    TravarMotor("r",falso)  
    Motor("r",0+velocidade)  
}  
  
tarefa frente com numero velocidade {  
    TravarMotor("r",falso)  
    TravarMotor("l",falso)  
    Motor("r",0+velocidade)  
    Motor("l",0+velocidade)  
}  
  
inicio  
    AbrirConsole()  
enquanto (verdadeiro) farei {  
    parar()  
    se(Colorido("rc") e Colorido("lc")) entao {  
        frente(300)
```

```
Escrever("Ir para frente")
}

senao se (Colorido("rc")) entao {
    Escrever("virar esquerda")
    esquerda(200)
}

fim
```

Exercício 2

Utilizando os comandos vistos em aula, e utilizando a arena **minha Arena1_Aula2.sArena** baixada do Canvas, faça um programa que:

- Escreva “Início de programa”, acenda o led verde.
- Siga as linhas, virando a direita, esquerda e seguindo em frente (utilizando os sensores).
- Ao detectar que está no fim da pista (linha vermelha): pare o robô, escreva “Fim de programa”, desligue o led.

Exercício 3

Baixe a arena **minha Arena2_Aula2.sArena** do Canvas e teste o programa da atividade 1.

Verifique os problemas apresentados e relate nesta atividade.

Proponha possíveis soluções para o problema.

Implemente estas soluções e teste o novo programa.

Referência Bibliográfica

SBOTICS, Equipe. **SBotics Docs**. Disponível em:
https://docs.sbotics.net/sbotics/sBotics_Simulator/changes_in_the_sBotics_view.
Acesso em: 01 fev. 2024.

SBOTICS, Equipe. **Funções rEduc Simulation pt_BR**. Disponível em:
[https://code.sbotics.net/programming/platforms/c0db56ec-619f-498f-9fbb-7835735f0f5e/languages/4224cf8e-f00c-4cc5-bd08-f0d22fad011d/functions?category=.](https://code.sbotics.net/programming/platforms/c0db56ec-619f-498f-9fbb-7835735f0f5e/languages/4224cf8e-f00c-4cc5-bd08-f0d22fad011d/functions?category=)
Acesso em: 01 fev. 2024.