ENSINO DE FÍSICA COM ROBÓTICA AUMENTADA

Roteiro de montagem e programação de um robô utilizando as plataformas Lego Mindstorm NXT e Arduíno e descrição dos desafios de Cinemática utilizando pistas virtuais.

Manual do Professor

Gilberto de Oliveira Viana
Prof. Dr. Rafael João Ribeiro
Prof. Me. Gregory Vinicius Conor Figueiredo

Sumário

1. INTRODUÇÃO

Física e robótica estão presentes no cotidiano de todas as pessoas, mesmo sem ser percebida, pois existem equipamentos automatizados em todos os lugares, desde uma máquina de lavar até um celular de última geração. Mas onde entra o termo robô quando falamos em máquina de lavar ou em celular? Robô é um termo utilizado para descrever equipamentos que realizam atividades de forma autônoma ou programada, então a máquina de lavar pode ser considerada um “robô simples” que lava a roupa de maneira autônoma. O celular, por sua vez, passou por diversos processos automatizados durante sua fabricação.

Nos dias atuais dificilmente se encontra um produto que não foi passou por processos automatizados. Mas a Física, onde entra nessa história? A resposta é: em tudo. A máquina de lavar tem motores que transformam energia elétrica em energia mecânica, tem engrenagens utilizadas para aumentar o torque, ou aumentar e diminuir a velocidade de rotação do cesto de roupa; no celular existem componentes que só foram criados graças as descobertas da Física moderna como, por exemplo, o diodo túnel, além, é claro, das aplicações do eletromagnetismo.

Os dois exemplos do parágrafo anterior são simples, basta olhar ao redor, analisar, e ver que em tudo existe um fenômeno físico associado. Muitos desses fenômenos são de difícil compreensão por se tratar de algo abstrato ou que não são facilmente visíveis. Ao propor atividades utilizando o conceito de robótica aumentada, interação de robôs reais com cenários virtuais, busca-se facilitar esse entendimento, fornecendo ao aluno a oportunidade de manipular os objetos utilizados e analisá-los até que o conceito seja compreendido. O robô é um objeto real interagindo com ambientes virtuais durante os desafios.

Para concluir cada desafio será necessário compreender os conceitos físicos associados a ele. O uso da robótica aumentada pode facilitar a realização e a visualização de alguns experimentos ao dispensar o uso de uma grande estrutura e equipamentos caros.

Nos desafios apresentados neste trabalho serão analisados alguns conceitos de cinemática. Esta etapa é um começo para que, em trabalhos futuros, possam ser criados ambientes que envolvam outros assuntos mais complexos.

2. ROBÔ UTILIZANDO A PLATAFORMA ARDUÍNO

O robô utilizado nos desafios que serão descritos é mostrado na figura 1. Esse robô pode ser montado utilizando a plataforma Arduíno. Algumas partes são fundamentais no funcionamento do robô: o Arduíno visto na figura 2 é responsável por receber os sinais dos sensores, a programação e controle dos motores do robô; O sensor, mostrado na figura 3, é o “olho” do robô, é através dele que o robô “enxerga” os objetos da pista; a ponte H recebe os sinais do Arduíno e envia comandos aos motores recebem que fazem o robô se mover; a bateria fornece energia para o funcionamento de todo o sistema.

No Arduíno podem ser programados vários parâmetros, entre eles os que serão utilizados nos desafios:

- Velocidade do giro das rodas através de pulsos PWM enviados ao motor;
- Aceleração;
- Direção da rotação das rodas;
- Tempo de funcionamento e parada dos motores.

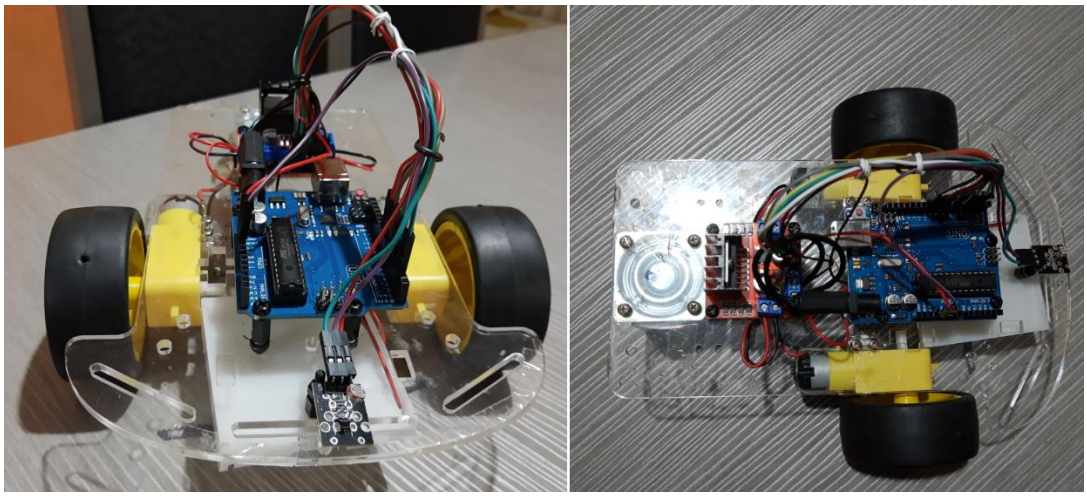


Figura 1 Robô utilizado nos desafios Fonte: Autoria Própria

Figura 2 Arduíno UNO Fonte: Autoria Própria

A plataforma Arduino, desenvolvida na década de 2000, surgiu do objetivo de estudantes e pesquisadores em viabilizar um sistema de fácil montagem de projetos eletrônicos destinados a processos de automação, sem a necessidade de conhecimentos avançados acerca da composição de placas e seus devidos comandos e funções. Dessa forma em 2005, Massimo Banzi e sua equipe criaram uma plataforma de código aberto, de baixo custo, capaz de proporcionar de modo mais fácil projetos práticos na área da eletrônica (SOUZA, 2013), que pode ser arranjada de muitas formas conforme o uso a que se destina e que já traz acoplados microprocessadores e programas pré-adaptados, sensores e botões, para acionamento de bombas hidráulicas, portões, sistemas de iluminação, motores e outras funcionalidades. Essa plataforma é chamada de Arduíno (RIBEIRO et al, 2017). Na figura 3 é possível ver o sensor de luminosidade utilizado para fazer a leitura da pista durante os desafios. Esse sensor é formado por um divisor de tensão composto por um LDR (Light Dependent Resistor), que tem sua resistência elétrica variável de acordo com a intensidade de luz recebida, e um resistor. Dessa forma a tensão de saída do divisor irá variar de acordo com a luminosidade que incide sobre o LDR. Ao ligar a saída do divisor de tensão em uma entrada analógica do Arduíno ele irá fazer a leitura desse valor. Dessa forma pode-se fazer uma leitura, calibrada, de 0 a 100 onde zero corresponde ao preto e cem corresponde ao branco. Pode ser feito também uma calibração digital, onde o nível zero corresponde ao preto e o nível 1 corresponde ao branco.

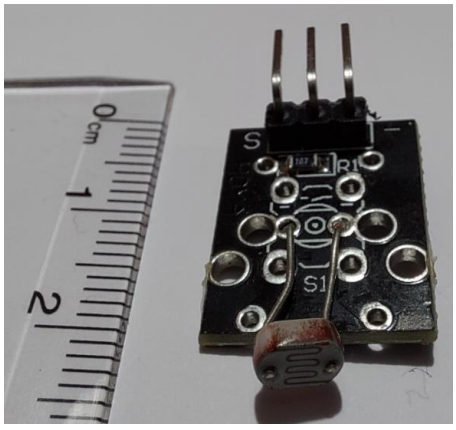


Figura 3 Sensor de luz LDR. Fonte: Autoria própria

3. MONTAGEM DO ROBÔ ARDUINO

O robô utilizado neste projeto foi desenvolvido utilizando o Arduino UNO, e um kit composto por chassis de acrílico motores de corrente contínua com redutor e rodas de borracha, que pode ser adquirido no mercado especializado em produtos para

robótica. Existem no mercado diversos kits de chassis e qualquer um deles pode ser utilizado. A descrição neste trabalho utiliza o modelo 2WD, mas o mesmo princípio pode ser utilizado para montagem de outros modelos. A lista de materiais utilizadas para montagem é:

- 1 placa Arduíno UNO;
- 1 Sensor de luz LDR;
- 1 Driver Ponte H L298N, para 2 motores;
- 8 Parafusos Plásticos em Nylon M3 x 12mm;
- 8 Espaçadores em Nylon M3 x 6mm - Fêmea x Fêmea
- 8 Porcas sextavadas M3
- 1 Kit Chassi contendo motor C.C. com redutor e rodas de borracha, por exemplo o kit comercial 2WD;
- 1 Suporte para 4 pilhas ou conector para bateria de 9V.

3.1 Montagem do chassi

O primeiro passo é montar os motores com redutores e as rodas no chassi do robô. A figura 4 mostra o kit e o chassi já montado. Geralmente o kit vem com as instruções de montagem.

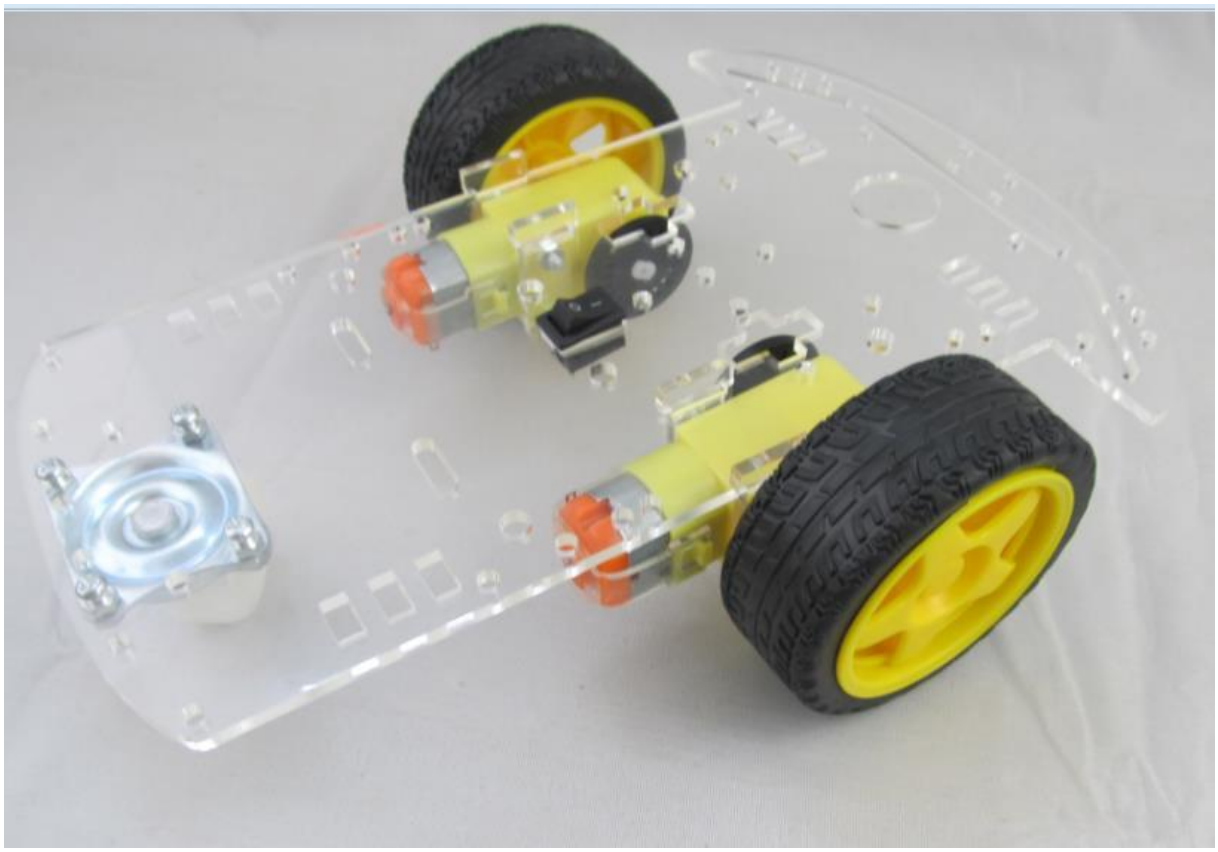


Figura 4 Chassi 2WD Fonte: Autoria própria

3.2 Definir layout

Após a montagem do chassi, verificar o layout de montagem das placas e sensor sobre o chassi e a posição do suporte de pilhas ou da bateria de 9V no chassi do robô. A figura 5 mostra um exemplo de layout que pode ser adotado na montagem. Nesta etapa, dependendo do layout, pode ser necessário fazer novas furações no chassi para fixar as placas. Apesar de ser uma tarefa simples deve-se atentar para a segurança a fim de evitar acidentes.

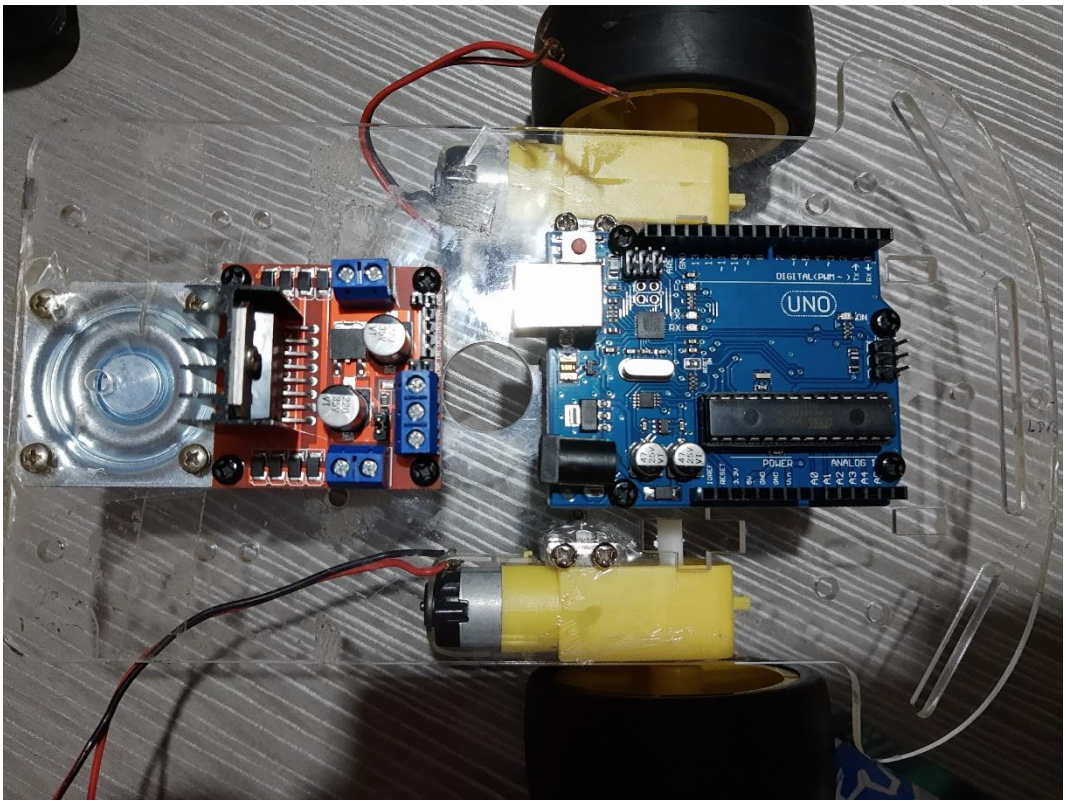


Figura 5 Layout de montagem das placas sobre o chassi. Fonte: Autoria própria

3.3 Instalar os componentes no chassi

A próxima etapa consiste em fixar os componentes no chassi. A figura 6 mostra os parafusos, porcas e espaçadores utilizados na fixação e a figura 7 mostra as placas com os espaçadores e parafusos prontos para serem instalados no chassi. O suporte de pilhas, mostrado na figura 8, também deve ser fixado utilizando os parafusos e espaçadores, conforme mostra a figura 9.

Figura 6 porcas, espaçadores e parafusos M3. Fonte: Autoria própria

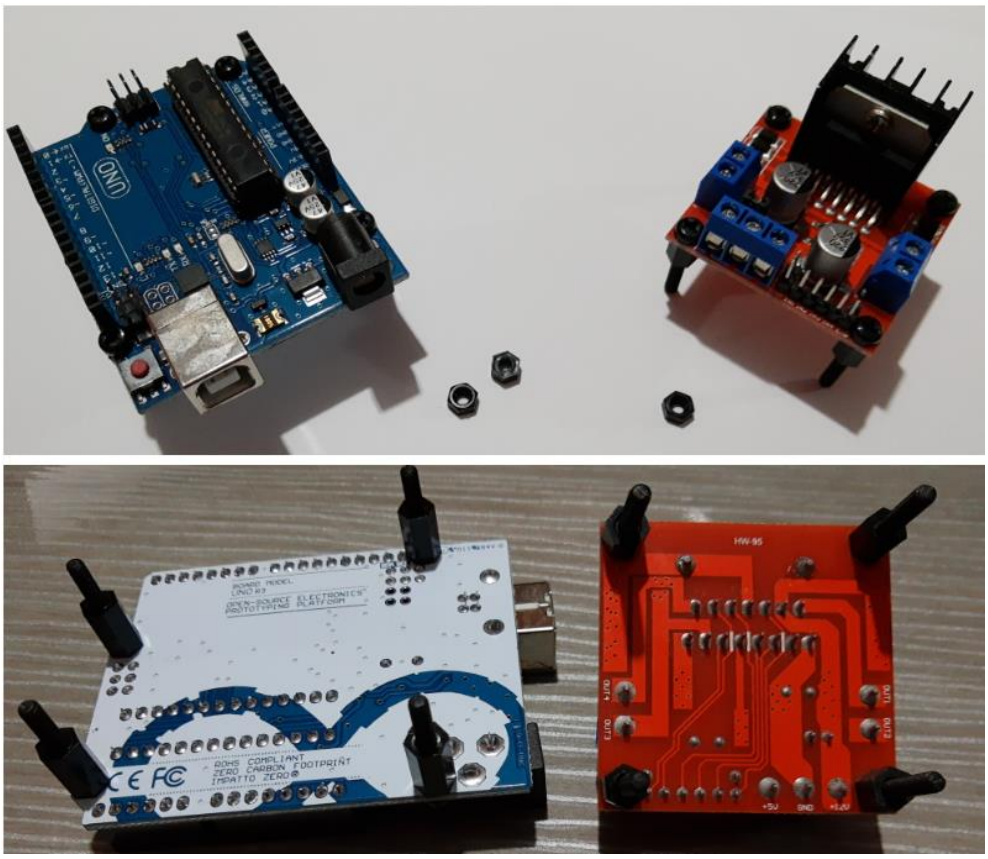


Figura 7 Placas com parafusos e espaçadores. Fonte: Autoria própria

A figura 8 mostra as placas fixadas no chassi

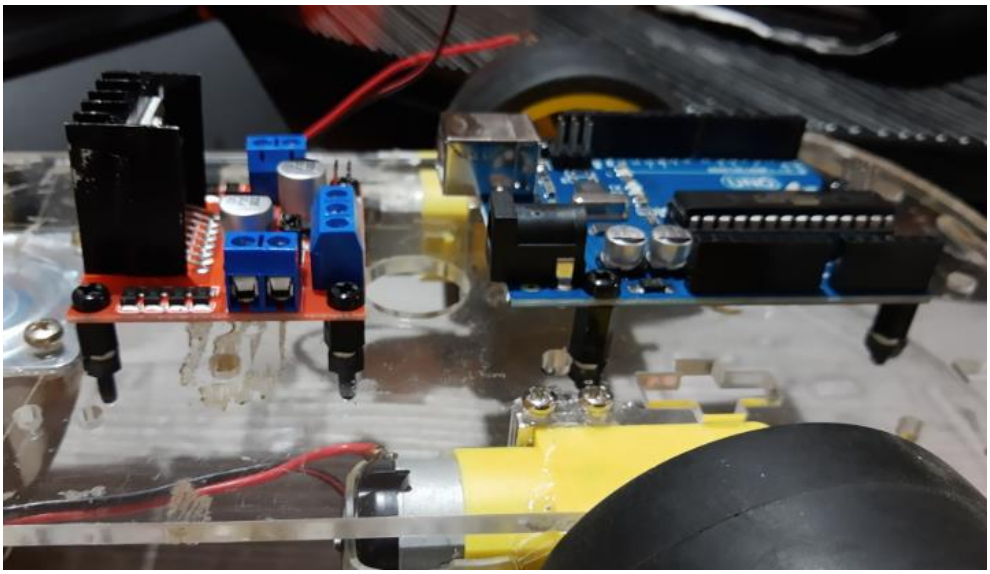


Figura 8 Detalhe das placas fixadas no chassi. Fonte: Autoria própria

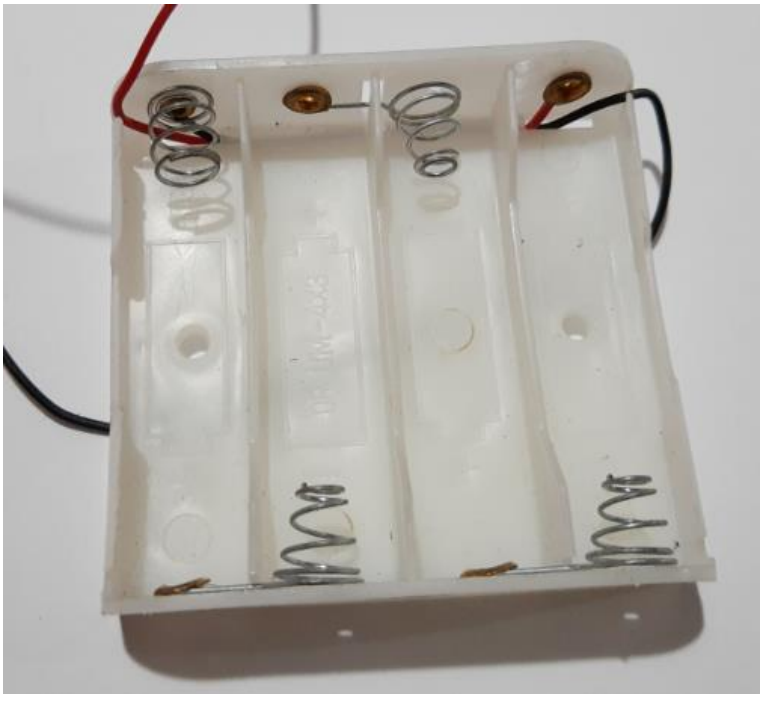


Figura 9 suporte de pilhas. Fonte: Autoria própria

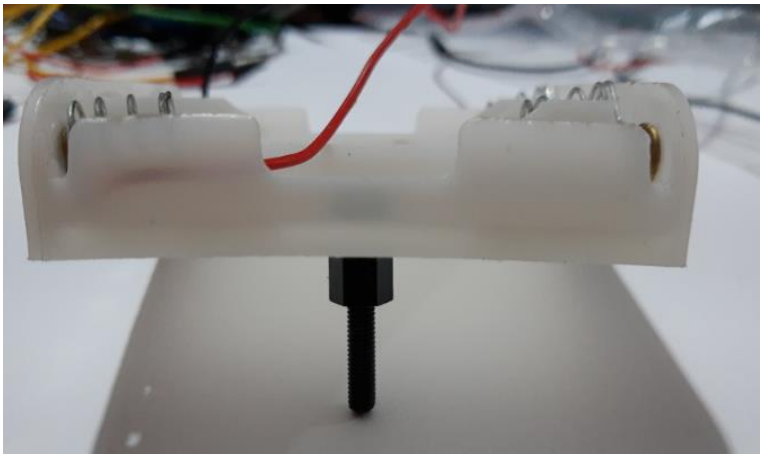


Figura 10 Detalhe da fixação do suporte de pilhas. Fonte: Autoria própria

3.4 fazer a instalação elétrica do robô

A figura 11 mostra as conexões do driver ponte H L298, A figura 12 mostra o diagrama elétrico do robô. A tabela 1 mostra um resumo das ligações a serem efetuadas

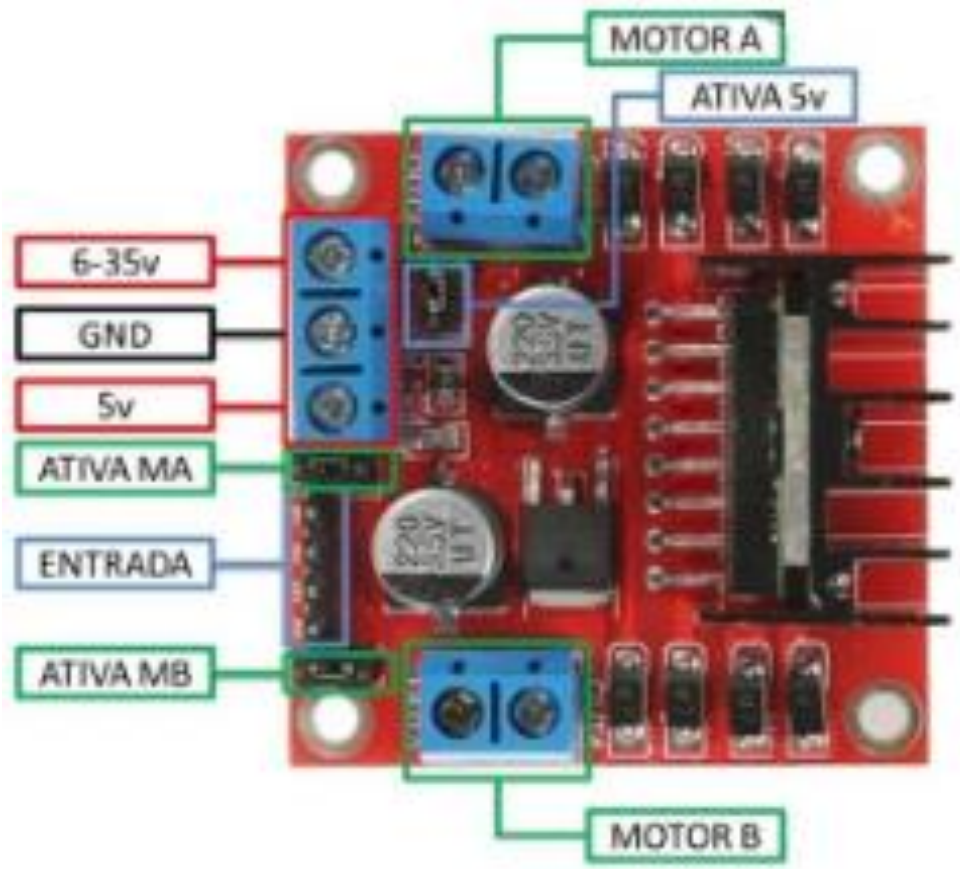


Figura 11 conexões ponte H para dois motores

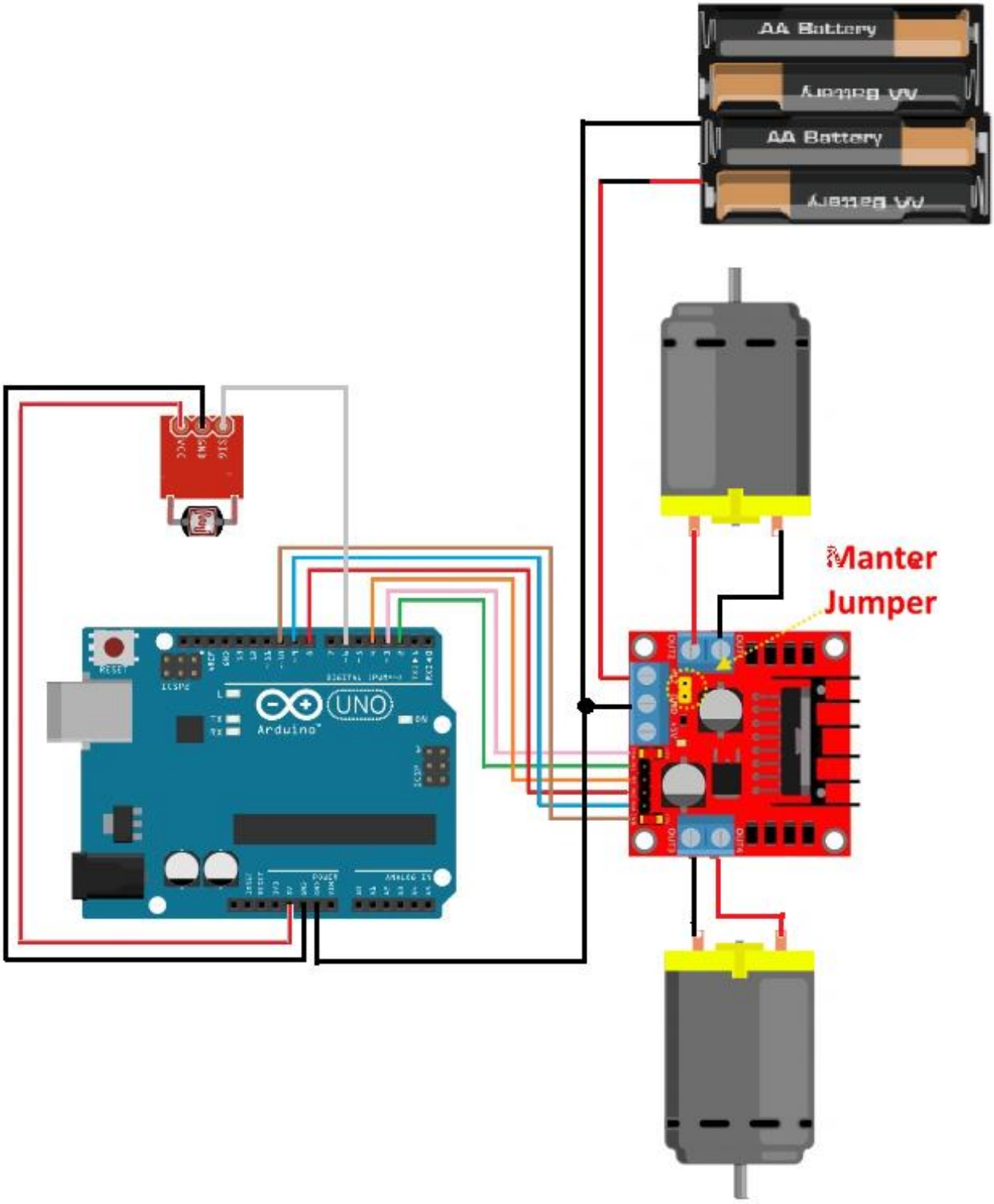


Figura 12 Diagrama Elétrico do robô Arduino

Tabela 1 Ligações entre os componentes do robô

Componente	pino	onde ligar
Driver Ponte H L298	IN1	Pino 2 do Arduino
	IN2	Pino 4 do Arduino
	IN3	Pino 8 do Arduino
	IN4	Pino 9 do Arduino
	Ativa MA	Pino 3 do Arduino
	Ativa MB	Pino 10 do Arduino
	6-35V	Ao polo positivo da bateria ou pilhas
	GND	pino GND do Arduíno polo negativo da bateria ou pilhas
Sensor de Luz	S	pino 6 do Arduíno
	+	pino +5Vcc Arduino
	GND	pino GND do Arduíno

4. PROGRAMANDO O ROBÔ ARDUINO

Para que o robô funcione e realize as tarefas contidas em cada desafio é necessário que ele receba as instruções do que fazer. Essas instruções são passadas para o controlador por meio de uma linguagem de programação. Existem diversas linguagens de programação cada uma com suas particularidades cuja finalidade é transferir para o robô os detalhes de todas as ações a serem realizadas. Na programação são descritas a velocidade de giro das rodas, o tempo que o motor vai funcionar ou ficar parado, o que fazer quando o sensor estiver em contato com um objeto, e todas as ações que o robô deverá executar.

4.1 Instalação da IDE Arduíno

Para escrever a programação e passá-la para o controlador do robô é necessário utilizar um *software* conhecido como IDE, sigla que vem do o inglês *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado. No trabalho apresentado será utilizado a IDE Arduíno. Para que esse software funcione é necessário ter instalado os drives do Arduíno no computador.

Os arquivos necessários para instalação da IDE Arduino podem ser baixados gratuitamente no endereço: https://www.arduino.cc/en/Main/Software. A figura 13 mostra as opções disponíveis para o usuário baixar e utilizar. Será descrito a instalação da IDE utilizando o instalador para Windows, porém, caso seja necessário

o professor poderá explorar e baixar as outras versões conforme a necessidade ou conveniência.

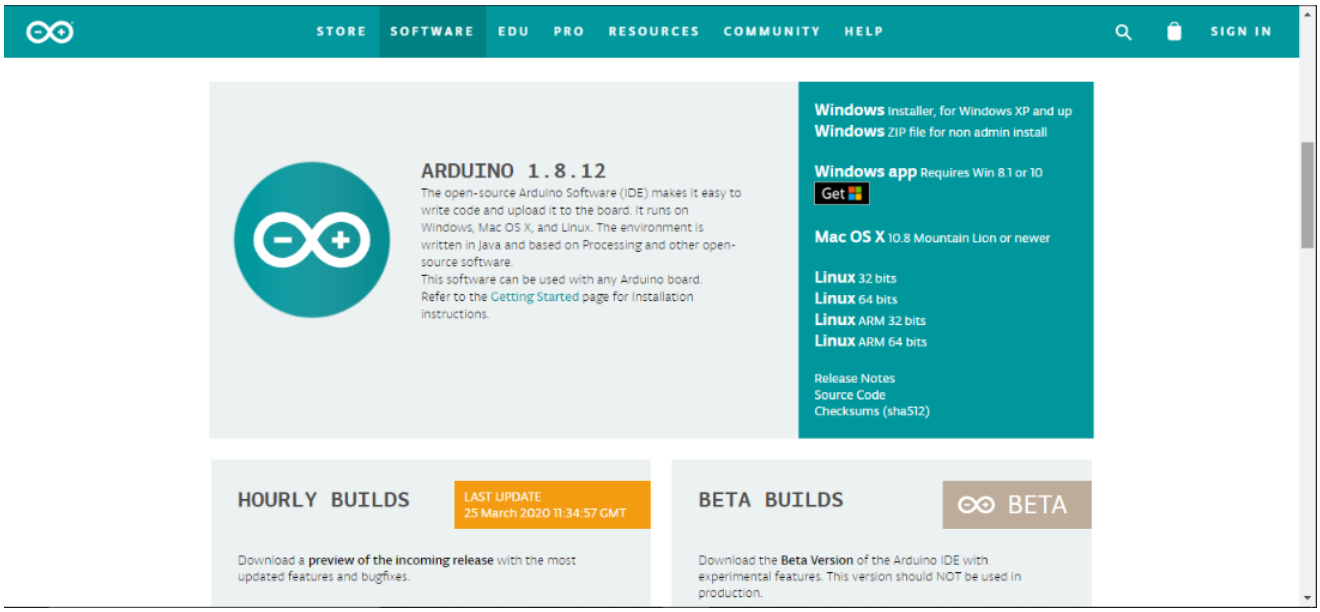


Figura 13 Tela do site para download da IDE Arduíno. Fonte: Arduino.cc

O primeiro passo para instalar a IDE é fazer o download do Instalador. Para isso clicar na opção *"Windows Installer*", conforme mostra a figura 14.

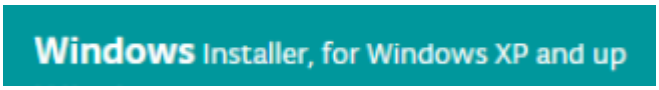


Figura 14 opção para download do instalador para Windows. Fonte: Arduino.cc.

Ao clicar na opção do Instalador irá abrir a tela mostrada na figura 15. Nesta tela clicar em Just Download, conforme figura 15.

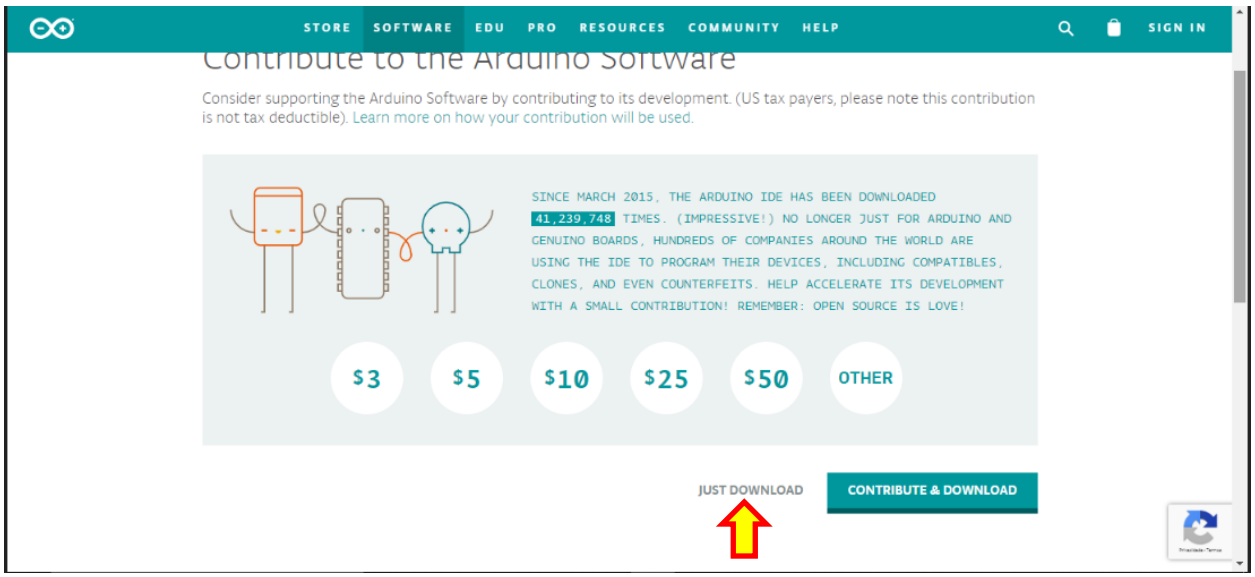


Figura 15 Tela de download do instalador da IDE Arduino. Fonte: Arduino.cc

Ao abrir o instalador será exibida a tela mostrada na figura 16. Nesta tela clicar deverá ser aceito os termos e condições da instalação, clicando em *I Agree*

Figura 16 Tela da Licença de instalação da IDE Arduino. Fonte: Arduino.cc

Após aceitar a licença irá abrir a tela, mostrada na figura 17, para escolha dos componentes a serem instalados. Deixar marcada todas as opções, para que sejam instalados todos os *drivers* necessários para utilização do Arduino, e clicar em *next*.

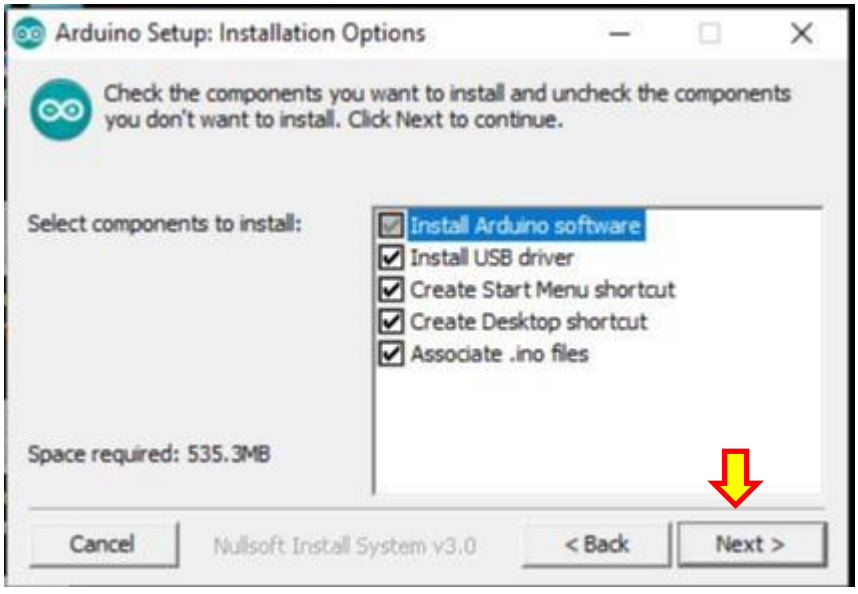


Figura 17 tela de componentes a serem instalados. Fonte: Arduino.cc

Na próxima tela, que mostra o local de instalação da ID, clicar em *Install*. A figura 18 mostra esta tela.

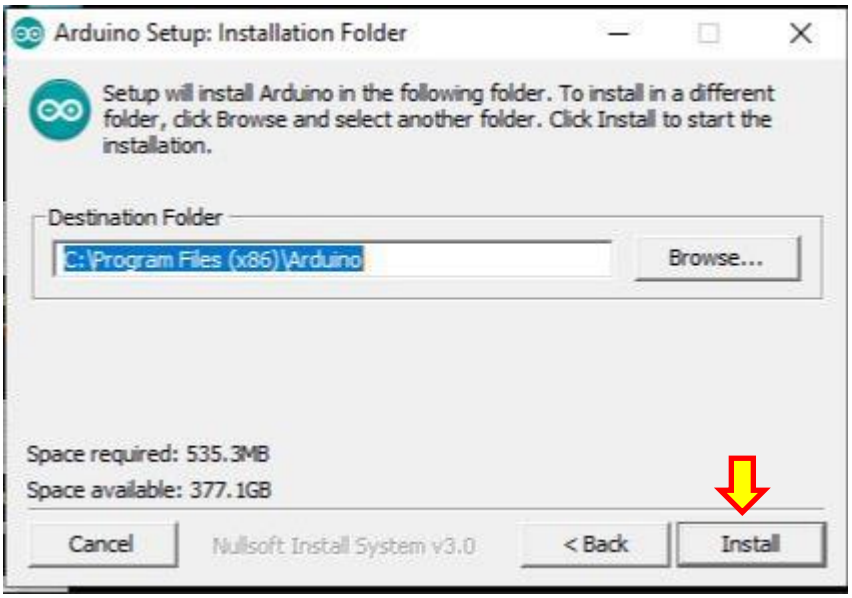


Figura 18 tela de instalação da IDE. Fonte: Arduino.cc

Ao clicar em *install* a instalação irá iniciar, abrindo a tela mostrada na figura 19 para acompanhamento do processo. Ao clicar em *Show details* poderá ser visualizado os detalhes dos componentes instalados, conforme mostra a figura 20.

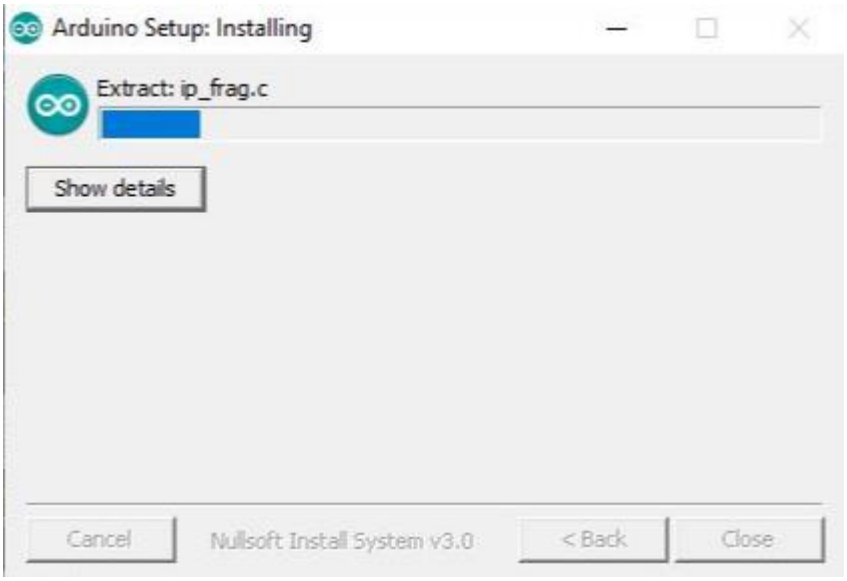


Figura 19 tela de acompanhamento do processo de instalação. Fonte: Arduino.cc

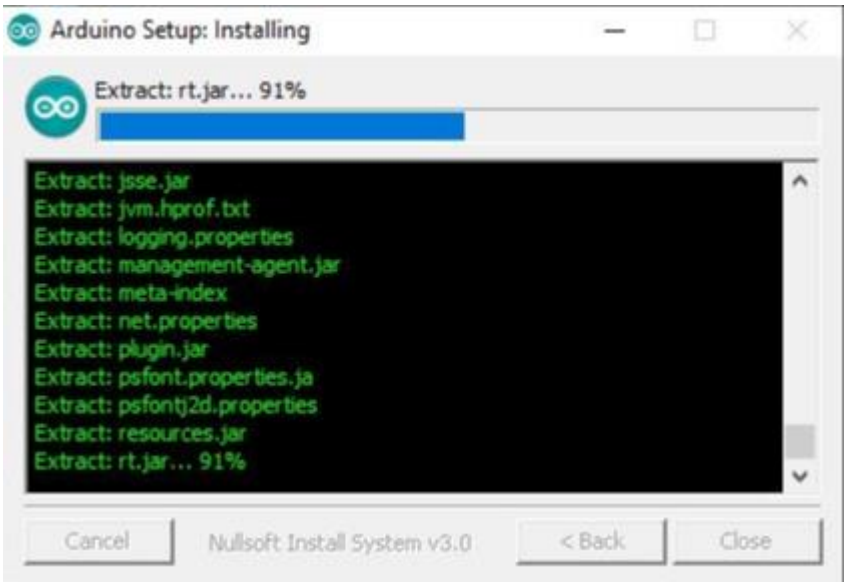


Figura 20 Detalhes da instalação. Fonte: Arduino.cc

No processo de instalação pode ser necessário permitir a instalação dos drivers necessários para o funcionamento da plataforma. A figura 21 mostra um exemplo de solicitação dessa instalação. Basta clicar em Instalar para dar andamento ao processo.

Figura 21 tela de permissão de instalação do driver USB Arduino. Fonte: próprio autor.

Após terminar a instalação clicar em *Close* para finalizar e fechar o instalador, conforme mostra a figura 22.

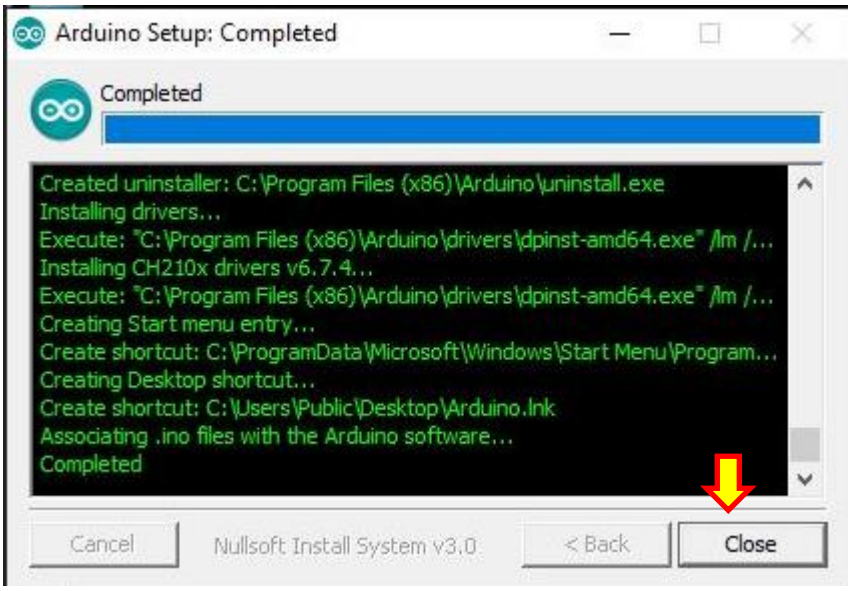


Figura 22 Tela de finalização da instalação. Fonte: Arduino.cc

Ao abrir a IDE Arduino pela primeira vez, poderá ser necessário permitir o acesso e desbloqueio de alguns recursos do aplicativo, conforme mostra a figura 23. Se aparecer essa solicitação, clicar em "*Permitir acesso*".

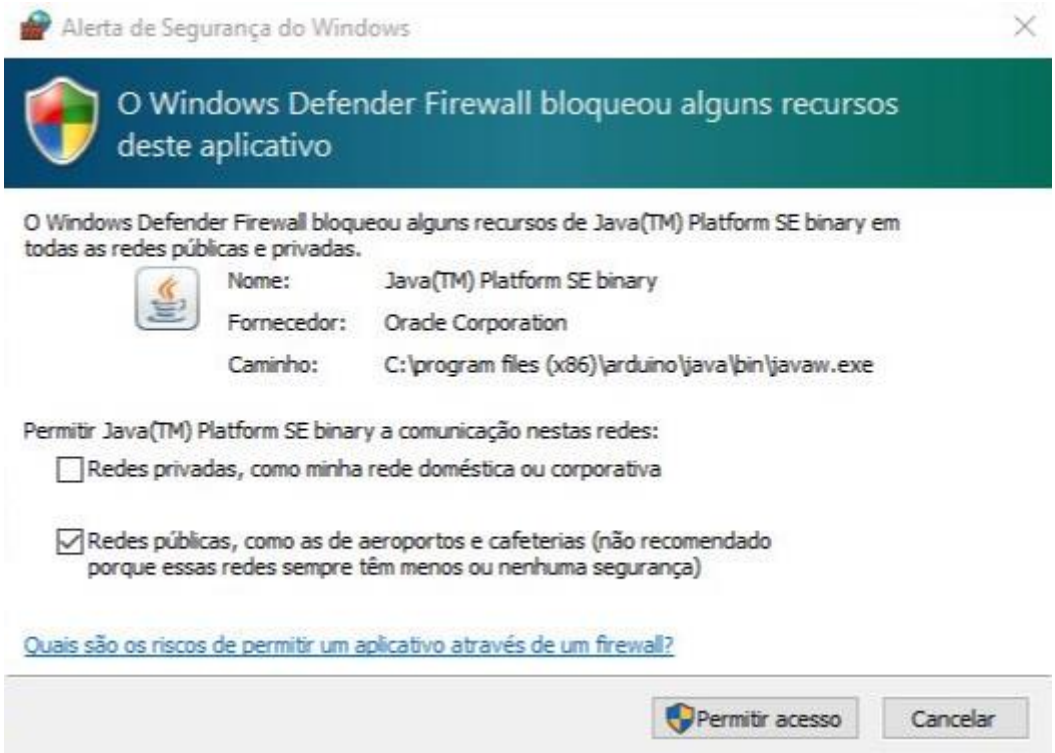


Figura 23 Desbloqueio de recursos do software. Fonte: próprio autor.

Maiores informações sobre a utilização do Arduino podem ser encontradas no endereço eletrônico: www.arduino.cc. Nesse endereço encontram-se dicas e ferramentas para a utilização da plataforma Arduino em diversas áreas.

4.2 Programação utilizando a IDE Arduino

O Arduíno é programado utilizando a linguagem C++. Essa linguagem utiliza comandos escritos que irão executar ações e enviar comandos aos motores do robô. Para entender a programação Arduino, antes de analisar o programa do robô, será analisado um exemplo simples que permite conhecer as principais características da linguagem de programação. O exemplo a ser analisado é o Blink, que faz um led piscar em uma frequência pré-determinada. O led que irá piscar já vem instalado na placa do Arduino, no pino 13, portanto não é necessária nenhuma ligação de componentes externos. Basta conectar o Arduino ao computador e escolher o modelo e a porta correta da conexão na IDE. A figura 24 mostra o detalhe da placa com o led instalado.

Figura 24 Led instalado na placa Arduino. Fonte: Próprio autor

Esse exemplo está disponível na IDE Arduino em Arquivos>Exemplos>1.Basics>Blink. A figura 25 mostra como acessar esse arquivo e a figura 26 mostra como o código irá aparecer na IDE.

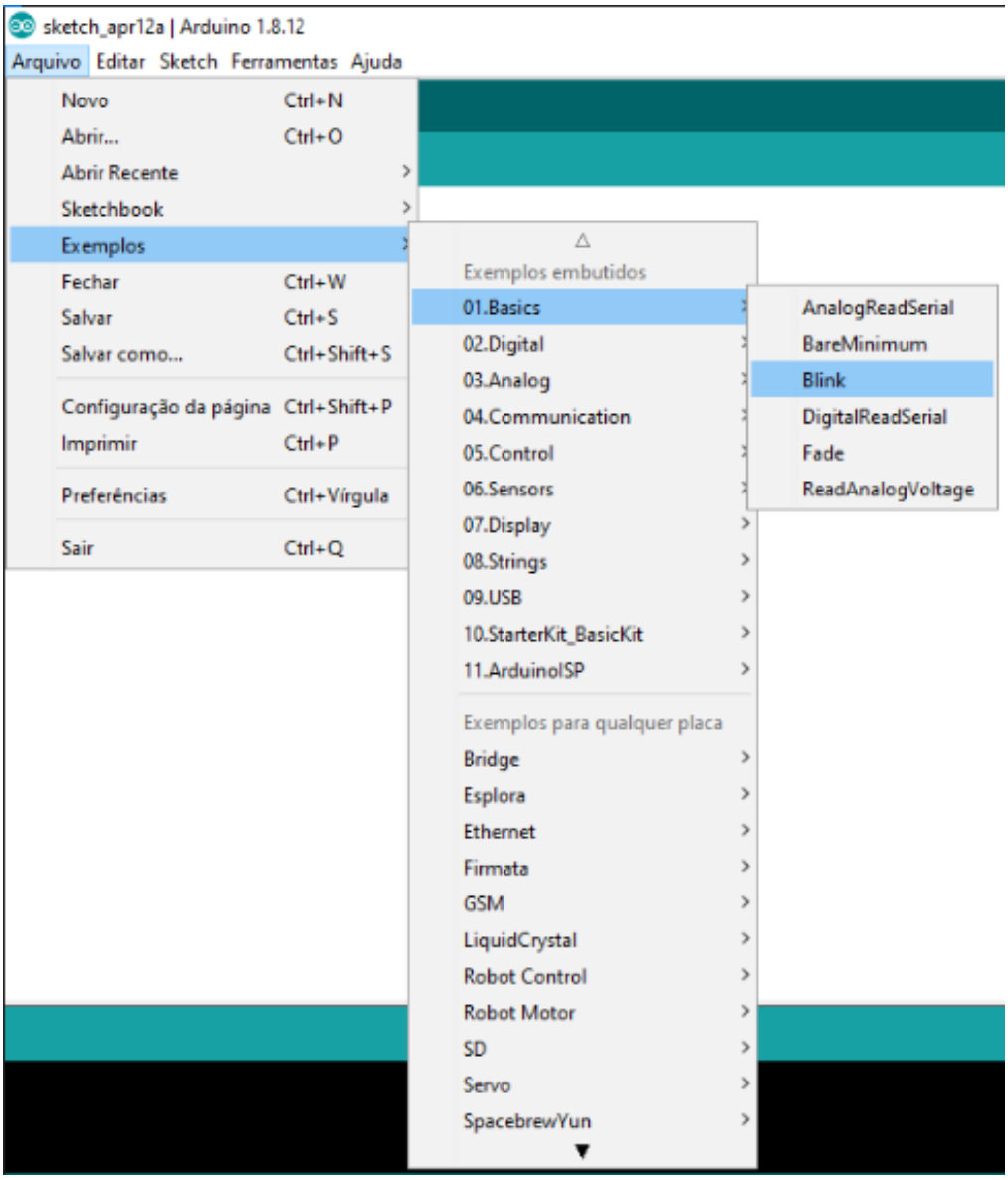


Figura 25 Acesso ao exemplo Blink. Fonte: próprio autor

```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
The circuit:
* LED connected from digital pin 13 to ground.
* Note: On most Arduino boards, there is already an LED on the board
connected to pin 13, so you don't need any extra components for this example.
Created 1 June 2005
By David Cuartielles
http://arduino.cc/en/Tutorial/Blink
based on an orginal by H. Barragan for the Wiring i/o board
*/

int ledPin =  13;    // LED connected to digital pin 13
// The setup() method runs once, when the sketch starts
void setup()   {
 // initialize the digital pin as an output:
 pinMode(ledPin, OUTPUT);
}
// the loop() method runs over and over again,
// as long as the Arduino has power
void loop()
{
digitalWrite(ledPin, HIGH);   // set the LED on
delay(1000);                  // wait for a second
digitalWrite(ledPin, LOW);    // set the LED off
delay(1000);                  // wait for a second
}
```

Figura 26 Código Blink, Fonte: Robocore

No início da programação, antes das linhas de código, verifica-se o cabeçalho com as informações do programa. Essas informações, na linguagem C, são escritas na forma de comentários. Os comentários são interessantes para descrever, por exemplo, o que cada linha de programa irá executar. Para fazer um comentário quer irá se desenvolver por mais de 1 linha, deve ser usado os caracteres:

/* para começar um comentário de mais de 1 linha:

*/ para finalizar os comentários que foram feitos anteriormente

Para fazer um comentário em 1 linha apenas, utiliza-se:

// para fazer um comentário de apenas 1 linha

Os comentários não são obrigatórios na programação, mas facilitam o entendimento das linhas de código.

A estrutura do código possui uma ordem que deve ser respeitada, abaixo estão descritos a ordem e as estruturas que o código deve conter:

1° Estrutura de Inclusão de Bibliotecas: São conjuntos de funções desenvolvidas para uma aplicação particular. O ambiente de desenvolvimento Arduino já vem com algumas bibliotecas instaladas e outras podem ser encontradas para download na internet, de acordo com a necessidade do programador. No exemplo do Blink não há a necessidade de bibliotecas adicionais.

2° Estrutura de Declaração de Variáveis; no exemplo do Blink a declaração de variáveis está na seguinte linha:

```
int ledPin = 13; // LED connected to digital pin 13
```

Os elementos dessa linha de comando são os seguintes:

- int: declara a variável do tipo inteira;
- ledPin = 13: nomeia a variável com o nome ledPin e que o valor desta variável está relacionado ao pino 13;
- // LED connected to digital pin 13: comentário que explica que o led está conectado ao pino digital 13 do Arduino.

Este trabalho não irá se aprofundar nos tipos de variáveis pois não é esse o objetivo do trabalho, porém existem diversos tutoriais na internet que explicam sobre esse assunto de forma didática e de fácil compreensão.

3° Estrutura Setup: O setup roda no início do programa e é responsável pelo ajuste do valor das variáveis e qual será o modo de operação dos pinos utilizados ao iniciar o programa. No exemplo estudado o setup está escrito da seguinte forma:

```
void setup() {

// initialize the digital pin as an output:

pinMode(ledPin, OUTPUT);

}
```

Os elementos dessa estrutura são:

- void setup() { : Declaração que irá começar o Setup do programa. Sempre aberto com uma “{“ e fechada, no fim da declaração, por uma “}”.
- // initialize the digital pin as an output: : Comentário dizendo que o pino digital será inicializado como uma saída
- pinMode(ledPin, OUTPUT); : Escolha do modo do pino, se é entrada (INPUT) ou saída (OUTPUT)

4° Estrutura Loop: O loop é programa principal que ficará rodando por tempo indeterminado e, nesse exemplo, responsável pelo funcionamento do pisca led. Ele está escrito da seguinte maneira:

```
void loop()

{

digitalWrite(ledPin, HIGH); // set the LED on

delay(1000); // wait for a second

digitalWrite(ledPin, LOW); // set the LED off

delay(1000); // wait for a second

}
```

A seguir a explicação do loop:

- void loop() : comando que irá começar o loop do programa e é aberto com uma “{“ e fechado com uma “}”.
- digitalWrite(ledPin, HIGH); // set the LED on : Escrita digital. Por tratar-se de um pino digital, ou você terá nível lógico 1 ou terá nível lógico 0, no caso de um led, ou terá o led acesso (1) ou terá o led apagado (0). O comando então liga o led, ou seja, envia 1 para o pino 13
- delay(1000); // wait for a second : Delay é mais uma função pronta do Arduino. O número que for inserido entre os parêntesis será o valor, em milissegundos, que o Arduino irá esperar para seguir para a próxima

instrução. Neste exemplo há um delay de 1000 milissegundos, ou seja, uma espera de 1 segundo para executar a próxima instrução.

- digitalWrite(ledPin, LOW); // set the LED off e delay(1000); // wait for a second : Estes dois comandos são análogos aos dois vistos anteriormente, com a única diferença que a escrita digital escreverá um 0 no pino do led, ou seja, um nível lógico baixo: o led apagará e o Arduino espera 1 segundo para fazer a próxima instrução que, no caso, volta a ser o digitalWrite(ledPin, HIGH); .

5° Demais estruturas de funções. Em algumas programações podem ser necessárias funções adicionais para que o programa funcione de maneira correta. Nestes casos essas funções serão escritas após as estruturas explicadas anteriormente.

Após terminar a programação é necessário fazer a compilação para verificar se não existem erros e enviá-la para o Arduino. Para compilar basta clicar no botão *verificar,* mostrado na figura 27. Após a compilação se não houver erros aparecerá a mensagem *"compilação terminada"* na barra de status mostrada na figura 28. Após essa etapa o programa pode ser gravado no Arduino através do botão *carregar* mostrado na figura 27. Após carregado, o led mostrado na figura 24 irá começar a piscar no intervalo definido na programação.

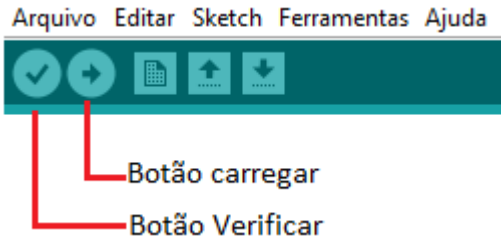


Figura 27 Botões compilar e carregar. Fonte: próprio autor

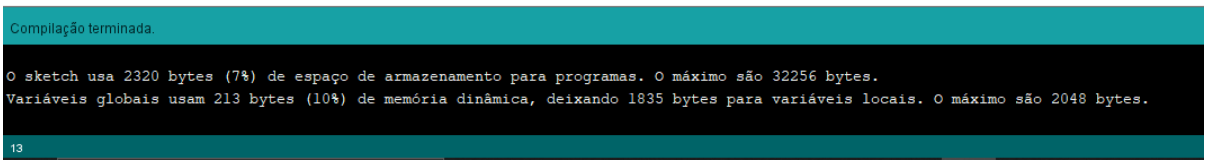


Figura 28 Barra de status. Fonte: próprio autor

Após entendido a estrutura da programação do Arduino, será analisado a programação do robô utilizado nos desafios propostos neste trabalho.

4.3 Programação do robô utilizado nos desafios

Na sequência está o código utilizado para a programação do robô que será utilizado nos desafios de cinemática utilizando a interação entre robôs reais e cenários virtuais.

```
1.  /*DECLARAÇÃO DE VARIAVEIS*/
2.  #define MotorA_sentido1 2 //define a variavel com nome MotorA_sentido1 com valor no pino 2
3.  #define MotorA_sentido2 4 //define a variavel com nome MotorA_sentido2 com valor no pino 4
4.  #define MotorB_sentido1 8 //define a variavel com nome MotorB_sentido1 com valor no pino 8
5.  #define MotorB_sentido2 9 //define a variavel com nome MotorB_sentido2 com valor no pino 9
6.  #define MotorA_PWM 3      //define a variavel com nome MotorA_PWM com valor no pino 3
7.  #define MotorB_PWM 10     //define a variavel com nome MotorA_PWM com valor no pino 10
8.
9.  #define veloc0 0   //define os valores de PWM na variável veloc0
10. #define veloc1 100 //define os valores de PWM na variável veloc1.
11. #define Sensor_luz 6 //define o pino onde será ligado o sensor.
12.
13. bool luz;
14.
15. /*ESTRUTURA SETUP*/
16. void setup() {
17.   Serial.begin(9600);
18.   pinMode(MotorA_sentido1, OUTPUT); //ajusta o pino relacionado a variável como saída
19.   pinMode(MotorA_sentido2, OUTPUT); //ajusta o pino relacionado a variável como saída
20.   pinMode(MotorB_sentido1, OUTPUT); //ajusta o pino relacionado a variável como saída
21.   pinMode(MotorB_sentido2, OUTPUT); //ajusta o pino relacionado a variável como saída
22.   pinMode(MotorA_PWM, OUTPUT)  //ajusta o pino relacionado a variável MotorA_PWM como saída
23.   pinMode(MotorB_PWM, OUTPUT); //ajusta o pino relacionado a variável MotorB_PWM como saída
24.   pinMode(Sensor_luz, INPUT);  //ajusta o pino relacionado ao sensor de luz como entrada
25.
26. }
27.
28.  /*ESTRUTURA LOOP*/
29. void loop() {
30.
31.   digitalWrite(MotorA_sentido1, LOW); //escreve o nível lógico baixo na respectiva saída
32.   digitalWrite(MotorA_sentido2, HIGH); //escreve o nível lógico alto na respectiva saída
33.   digitalWrite(MotorB_sentido1, HIGH); //escreve o nível lógico alto na respectiva saída
34.   digitalWrite(MotorB_sentido2, LOW); //escreve o nível lógico baixo na respectiva saída
35.
36.    /*Leituras do Sensor de luz */
37.   luz = digitalRead(Sensor_luz); //faz a leitura do nível lógico do sensor
38.   Serial.println(luz); //escreve no monitor serial
39.   Serial.print(" nivel logico sensor = "); //escreve no monitor serial
40.
41.    /*condição para o funcionamento dos motores de acordo com as leituras do sensor */
42.  if(luz == false){
43.     analogWrite(MotorA_PWM, veloc1); //escreve o valor analógico na saída MotorA_PWM
44.     analogWrite(MotorB_PWM, veloc1); //A velocidade deve calculada para cada desafio
45.
46.
47.  }else if(luz == true){
48.    analogWrite(MotorA_PWM, veloc0); //O motor para quando encontra os marcadores da pista
49.    analogWrite(MotorB_PWM, veloc0);
50.    digitalWrite(MotorA_sentido1, HIGH); //escreve o nível lógico alto na respectiva saída
51.    digitalWrite(MotorA_sentido2, LOW); //escreve o nível lógico baixo na respectiva saída
52.    digitalWrite(MotorB_sentido1, LOW); //escreve o nível lógico baixo na respectiva saída
53.    digitalWrite(MotorB_sentido2, HIGH); //escreve o nível lógico alto na respectiva saída
54.
55.  }
56. }
```

O código acima faz o ajuste da velocidade do motor para um valor constante, que deverá ser calculado de acordo com os parâmetros da pista e do desafio. Essa velocidade será ajustada no motor de forma “quase instantânea”. Caso precise programar o robô para acelerar até atingir uma determinada velocidade, pode ser utilizado um código desenvolvido pela Robocore (2020). Para isso as seguintes variáveis serão acrescentadas:

```
1.  bool luz;
2.  int i = 0; //declaracao da variavel de incremento para a rampa
3.  const int TEMPO_RAMPA = 500;
```

O valor da variável TEMPO_RAMPA é calculado de acordo com a aceleração desejada. Fica a critério do professor verificar o método de calcular esse valor. Além das variáveis, as linhas 43 e 44, do código anterior, devem ser substituídas pelas linhas de comando mostradas a seguir:

```
1.    //rampa de aceleracao
2.    for (i = 0; i < 256; i=i+10){  //contador crescente responsável pela aceleração
3.      analogWrite(MotorA_PWM, i);
4.      analogWrite(MotorB_PWM, i);
5.      delay(TEMPO_RAMPA); //intervalo para incrementar a variavel i
```

OBS: Esse código foi testado individualmente e não foi aplicado na prática em nenhum desafio. Fica como sugestão ao professor aplicá-lo. O desafio 3 descrito no capítulo 5.3 deste trabalho é o desafio ideal para este teste.

5. ROBÔ DE LEGO

Nos capítulos anteriores foi descrito a montagem do robô utilizando a plataforma Arduíno e a programação em linguagem C. Outra opção para montagem do robô é utilizar o *kit Lego Mindstorms NXT* e a programação em linguagem *JAVA*. Apesar desta linguagem ser um pouco mais complexa que a linguagem C, a programação idealizada para este trabalho não traz grandes dificuldades, mesmo para aqueles que não têm conhecimento em programação. A figura 29 mostra o robô utilizando esta plataforma. Algumas partes são fundamentais para o seu funcionamento: o controlador *NXT* visto na figura 30 é responsável por receber os sinais dos sensores, a programação e controle dos motores do robô; O sensor é o "olho" do robô, é através dele que o robô "enxerga" os objetos da pista; os motores recebem os sinais do controlador e fazem o robô se mover; a bateria fornece energia para o funcionamento de todo o sistema.

No controlador podem ser programados vários parâmetros, entre eles os que serão utilizados nos desafios:

- Velocidade do giro das rodas em graus por segundos;
- Aceleração em graus por segundos ao quadrado;
- Direção da rotação das rodas;
- Tempo de funcionamento e parada dos motores.

Figura 29 Robô utilizado nos desafios

Fonte: Autoria Própria

Figura 30 Controlador NXT
Fonte: Lego

Os botões do controlador têm as seguintes funções: O botão central, na cor laranja, é a tecla *"Enter"* e serve também para ligar o robô; as setas laterais servem para navegar nos menus; a tecla inferior, cinza escuro, é o botão *scape* e também tem a função de desligar o controlador.

Na figura 29 é possível ver o sensor de luminosidade utilizado para fazer a leitura da pista durante os desafios. Esse sensor recebe a luz e a transforma em sinais elétricos que são lidos pelo controlador. O valor lido pelo controlador depende da intensidade da luz que incide no sensor. Dessa forma pode-se fazer uma leitura, calibrada, de 0 a 100 onde zero corresponde ao preto e cem corresponde ao branco.

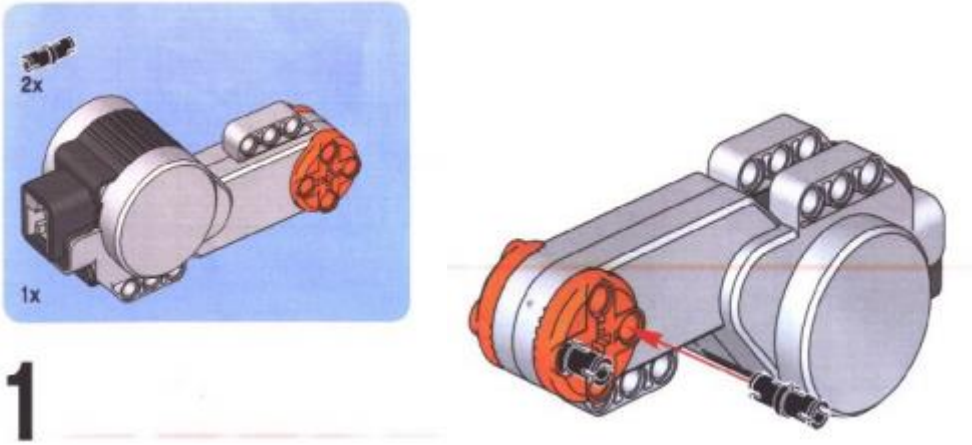
6. MONTAGEM DO ROBÔ LEGO

Os passos para montagem do robô foram copiados do manual que acompanha o *kit Lego Mindstorm*. Trata-se de uma montagem fácil e nada impede que cada um modifique a montagem e crie um robô personalizado.

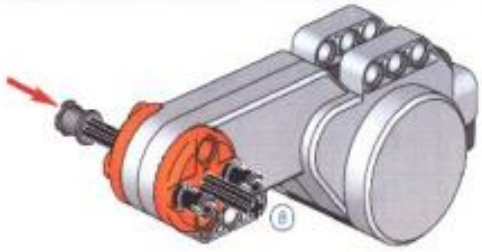
O robô utilizado como exemplo ficará com o aspecto mostrado na figura 31

Figura 31 Robô montado como exemplo
Fonte: Lego

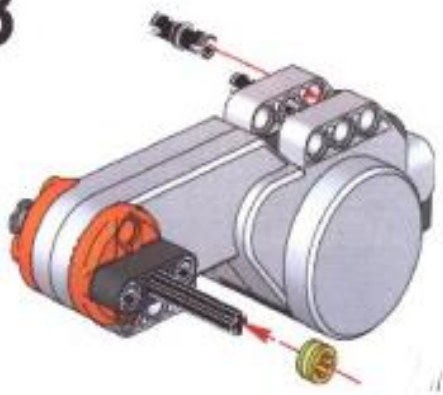
A seguir estão as imagens retiradas do manual do *kit Lego Mindstorms NXT* que mostram o passo a passo da montagem do robô. Os passos estão numerados e em cada etapa existem retângulos com os detalhes de encaixe das peças e quantidade de peças utilizados.



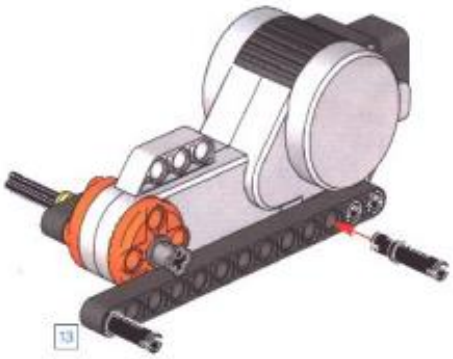
2

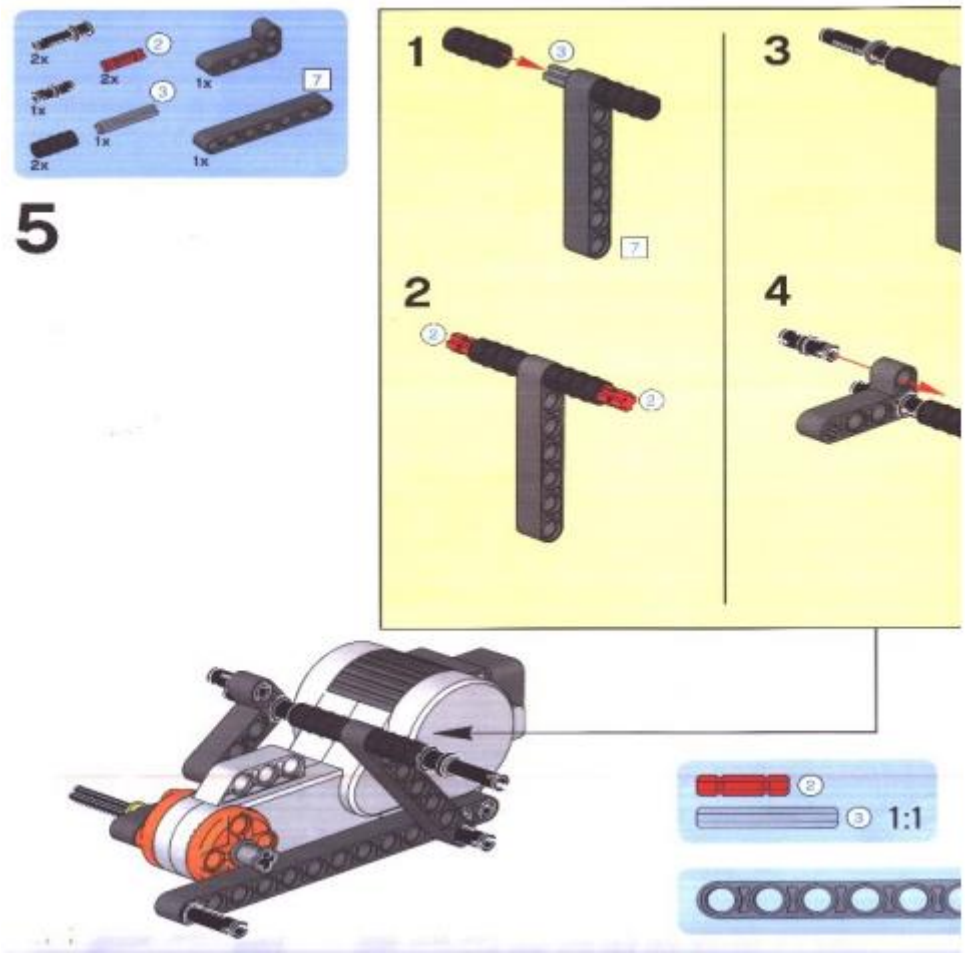


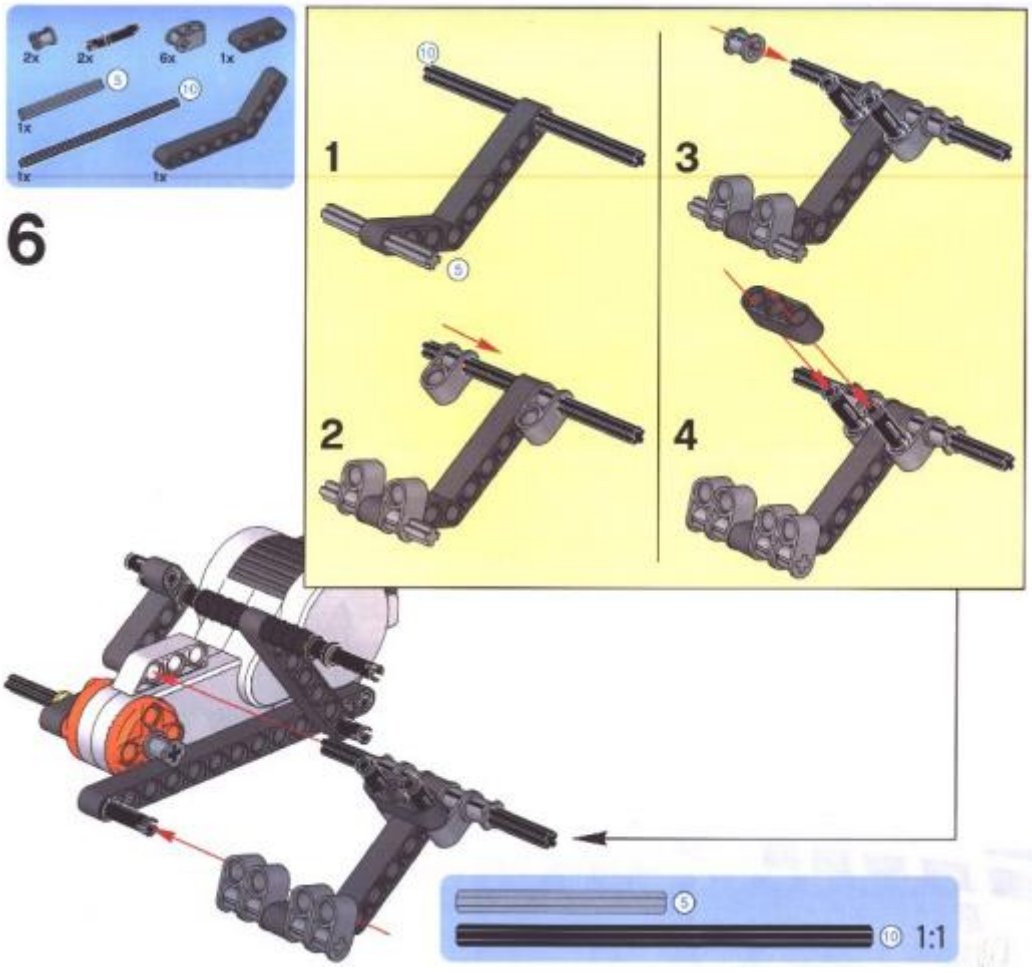
3



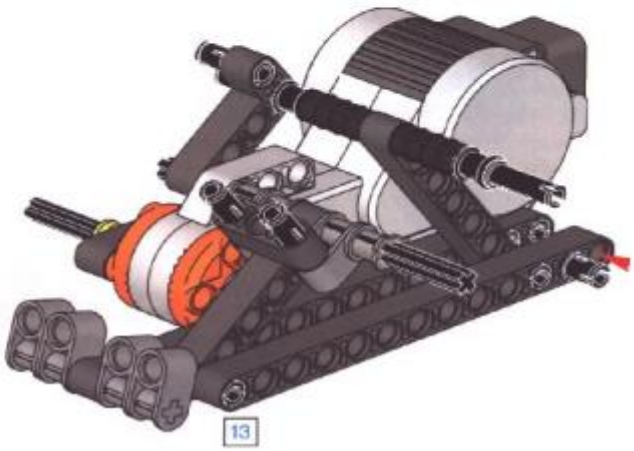
4

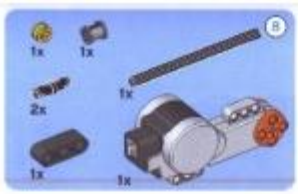



5
1
2
3
4
1:1

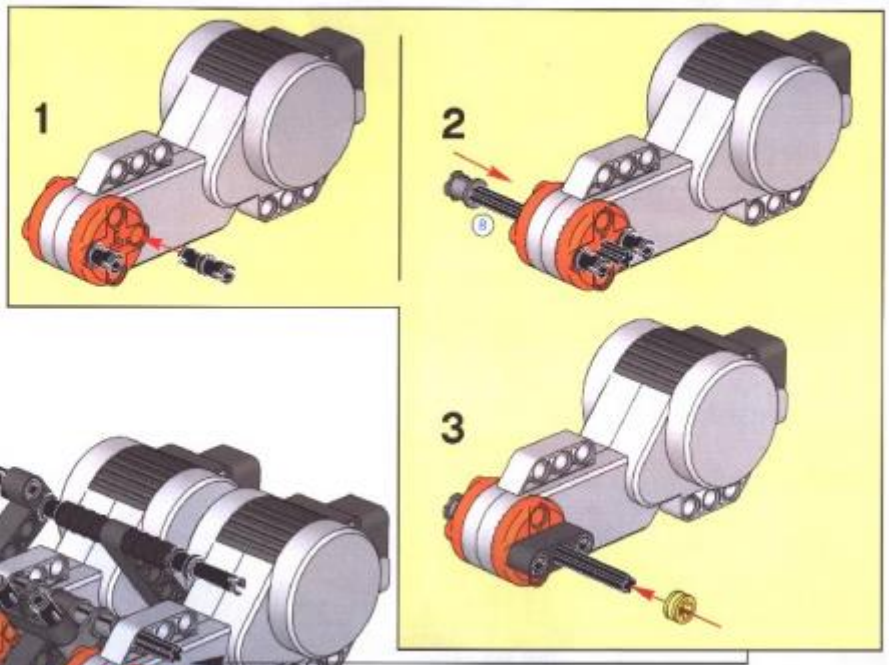

6
1
2
3
4
1:1

7

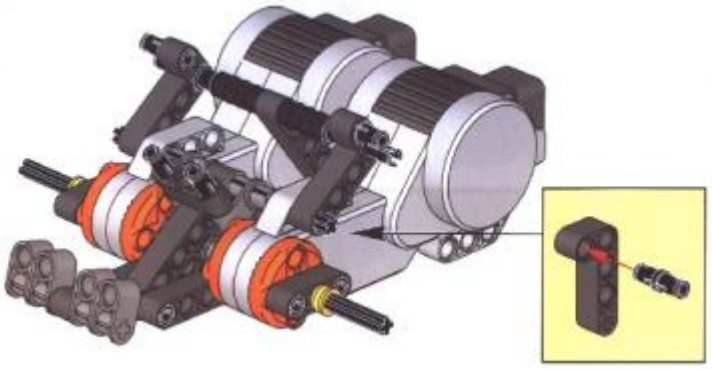




8

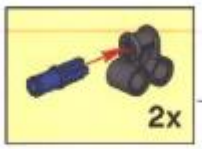


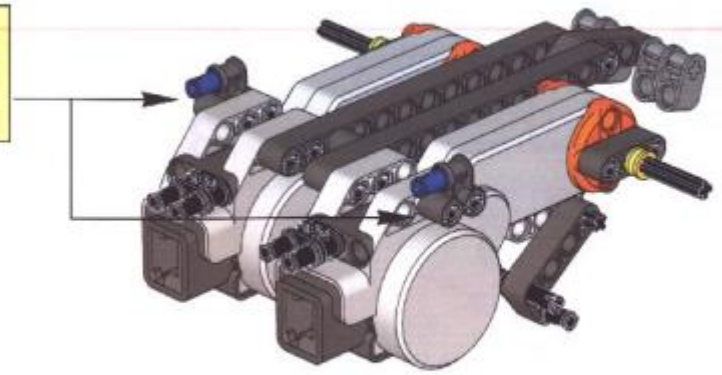
9



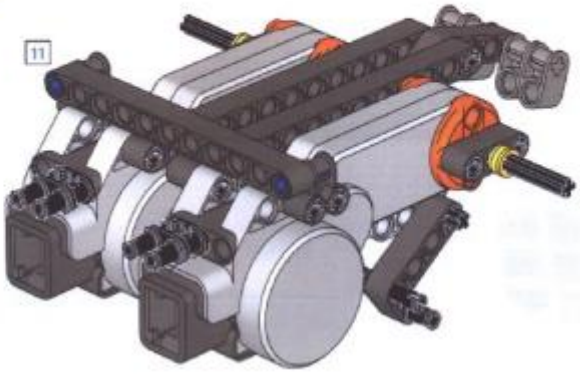
10

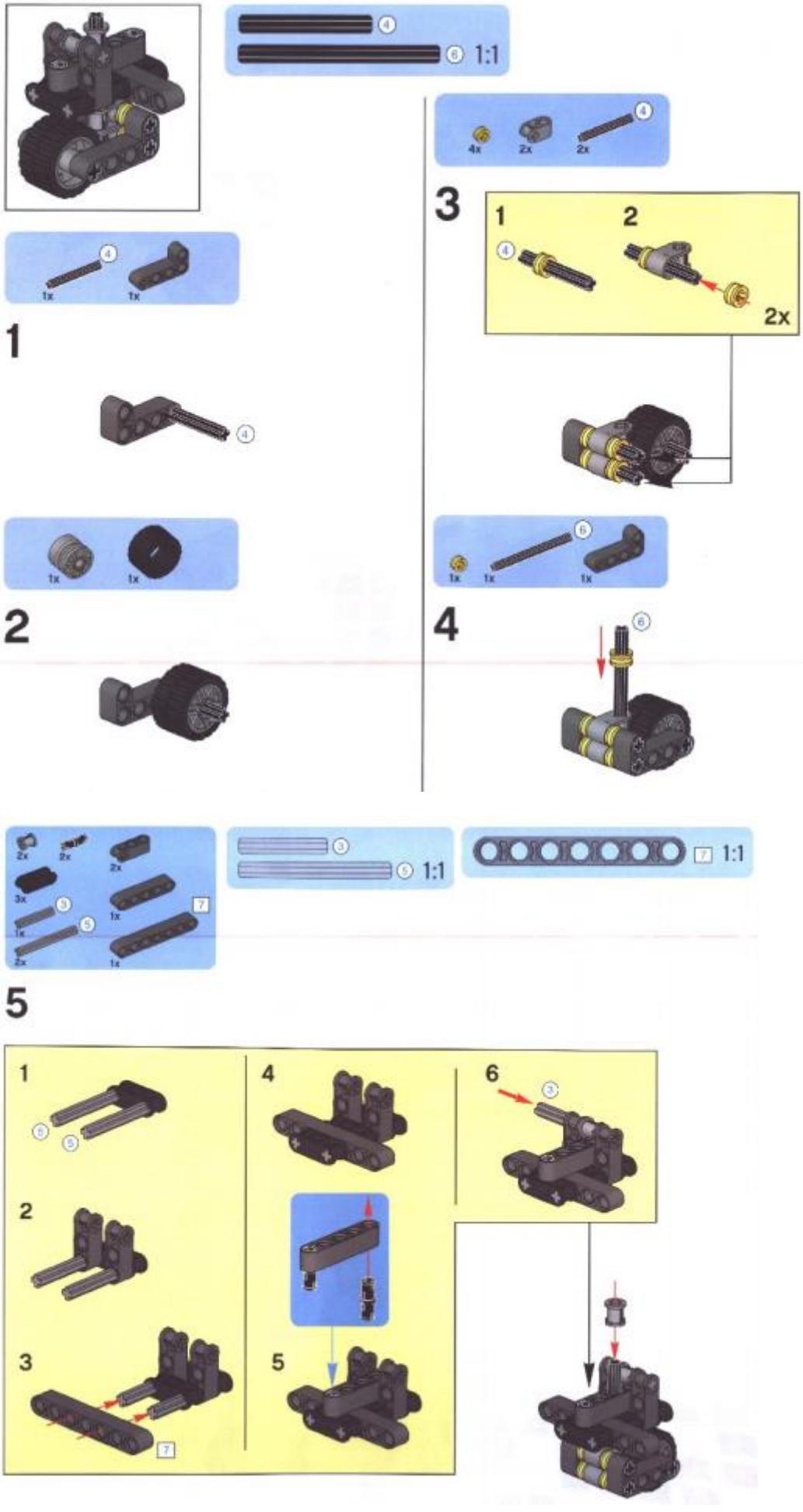
11



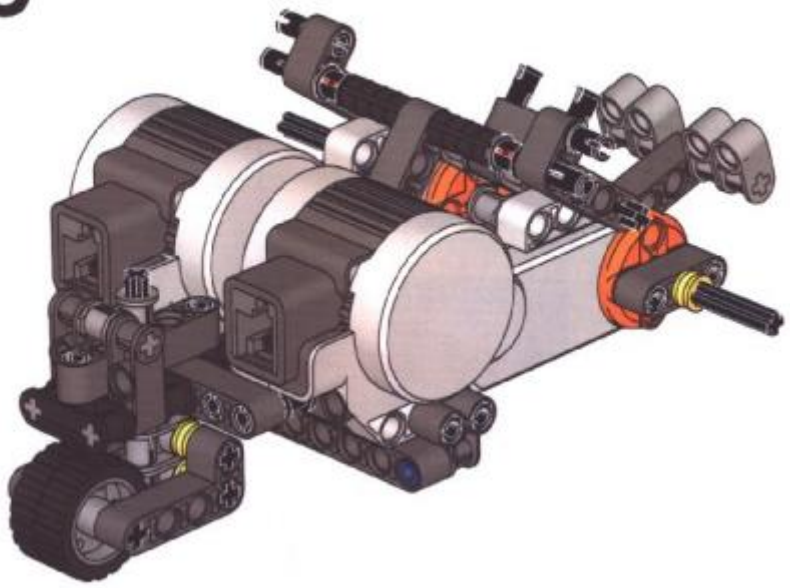


12

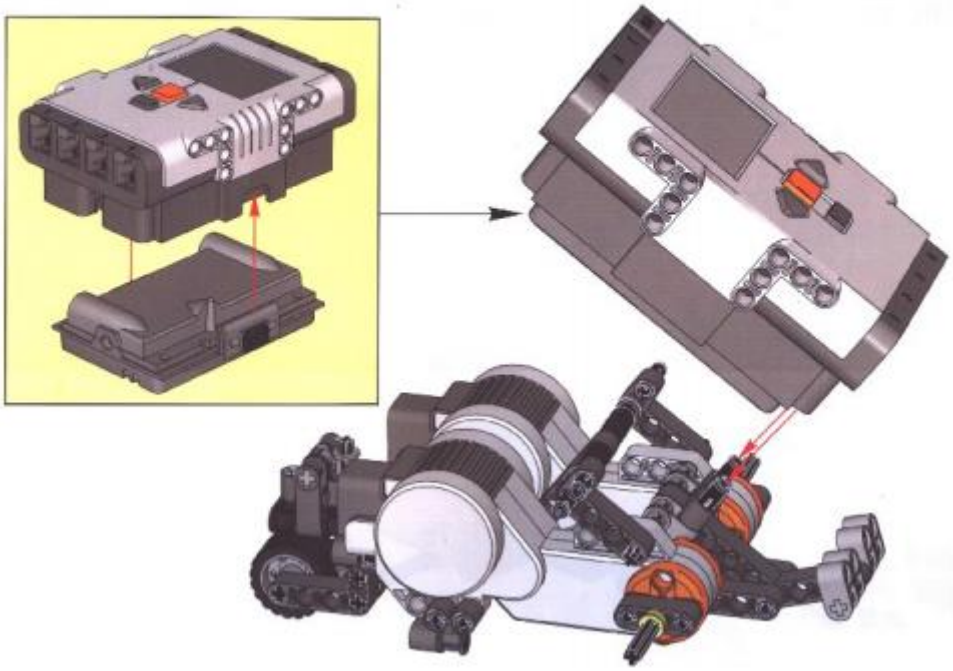



1
2
3
4
5
1:1
2x

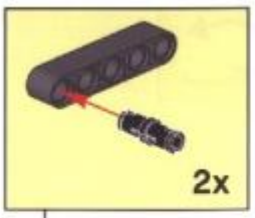
13

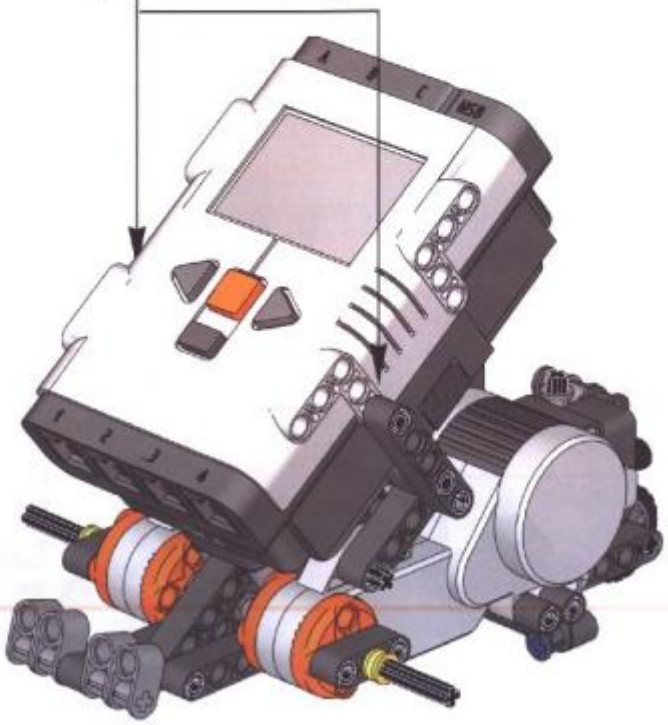


14



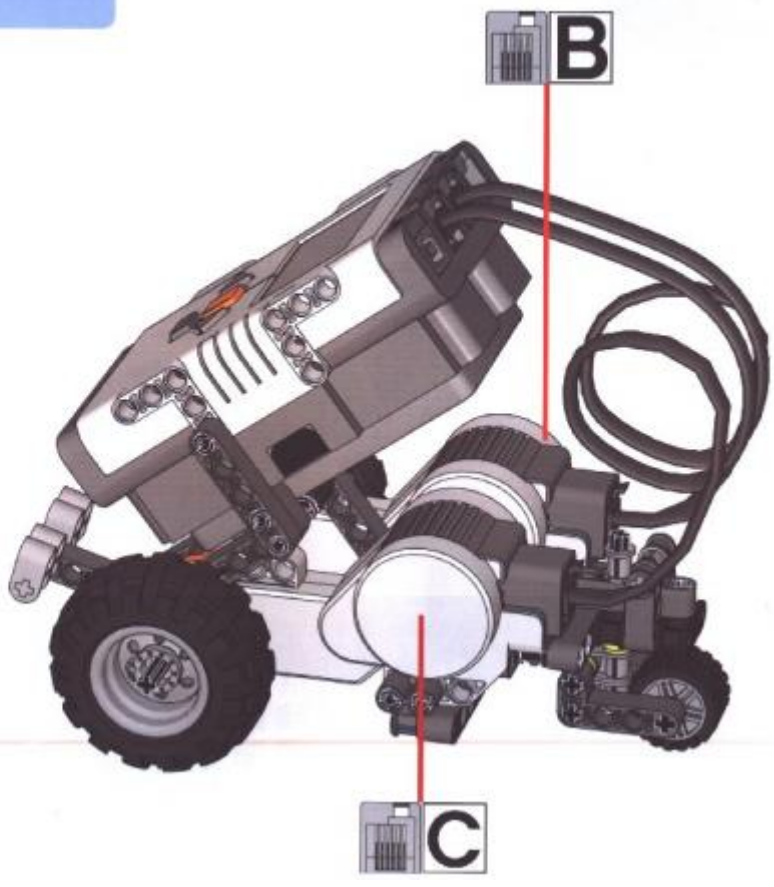
15

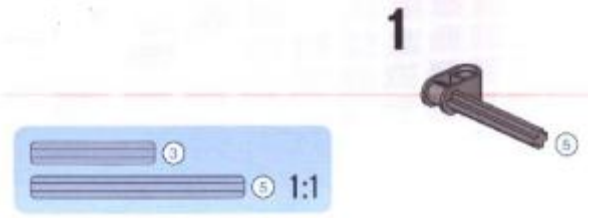


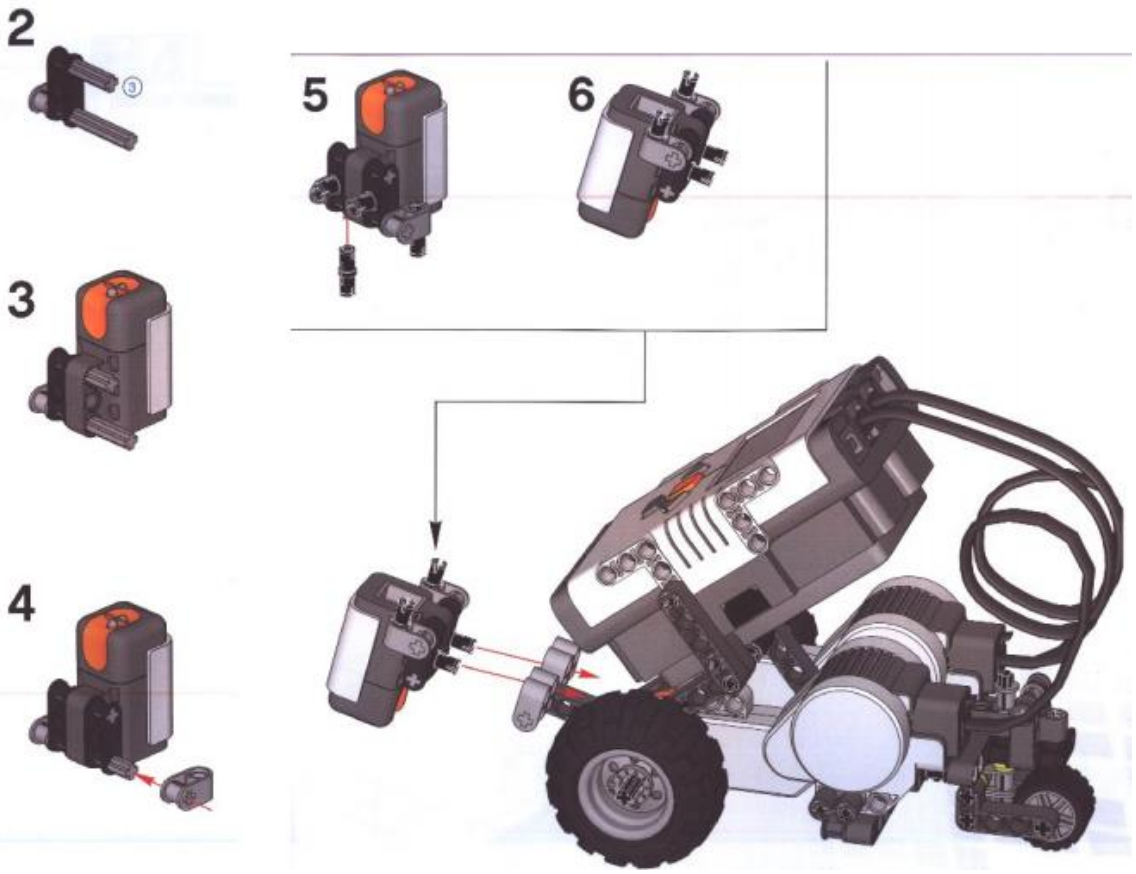


16

17







Montagem do sensor:

Nas figuras dessa página está a sequência de montagem do sensor de luminosidade.

Está mostrado a montagem do sensor apontado para o chão, pois é a maneira utilizada em pistas convencionais. No caso dos desafios apresentados nesse trabalho o sensor será montado apontado para cima, pois é de onde virá a luz das pistas virtuais.

7. PROGRAMANDO O ROBÔ LEGO

Para que o robô funcione e realize as tarefas contidas em cada desafio é necessário que ele receba as instruções do que fazer. Essas instruções são passadas para o controlador por meio de uma linguagem de programação. Existem diversas linguagens de programação cada uma com suas particularidades cuja finalidade é transferir para o robô os detalhes de todas as ações a serem realizadas. Na programação são descritas a velocidade de giro das rodas, o tempo que o motor vai funcionar ou ficar parado, o que fazer quando o sensor estiver em contato com um objeto, e todas as ações que o robô deverá executar.

Além das funções em algumas linguagens, como Java por exemplo, outros parâmetros, próprios da linguagem, devem ser escritos durante a programação. Outras programações utilizam blocos ao invés de escrita, o que facilita o trabalho em alguns casos. Como o objetivo desse trabalho não é formar programadores, não serão detalhados todos os parâmetros de programação. Será ensinado o básico para que o robô possa funcionar e realizar os desafios.

7.1 Instalação da Ferramenta Java (JRE e JDK)

Para escrever a programação e passá-la para o controlador do robô é necessário utilizar um *software* conhecido como IDE, sigla que vem do o inglês *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado. No trabalho apresentado será utilizado o programa *Java Eclipse Neon* que não serve apenas para programar robôs de Lego, mas também para criar programas para o computador. Para que esse software funcione é necessário ter o conjunto de ferramentas e compilador Java (JDK) instalado e configurado. Além da IDE é necessário um complemento, dentro da IDE, para fazer a comunicação do computador com o robô, esse complemento é denominado LeJOS.

Primeiramente é necessário baixar o arquivo necessário para a instalação utilizando o seguinte endereço eletrônico:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Recomenda-se que seja utilizado a versão de 32 bits (Windows x86, pois a versão de 64 bits pode apresentar algumas incompatibilidades ao rodar no sistema operacional Windows. A versão do Java utilizada para este trabalho foi a versão 8 (*jdk1.8.0_131*), rodando em um computador com Windows 10 64 btis.

Após baixar os arquivos executar o instalador e seguir as instruções na tela clicando em "*next*" para ir ao próximo passo e *"finish"* para concluir. É interessante informar o diretório de instalação, importante para realizar a configuração do JDK conforme Figura 32.

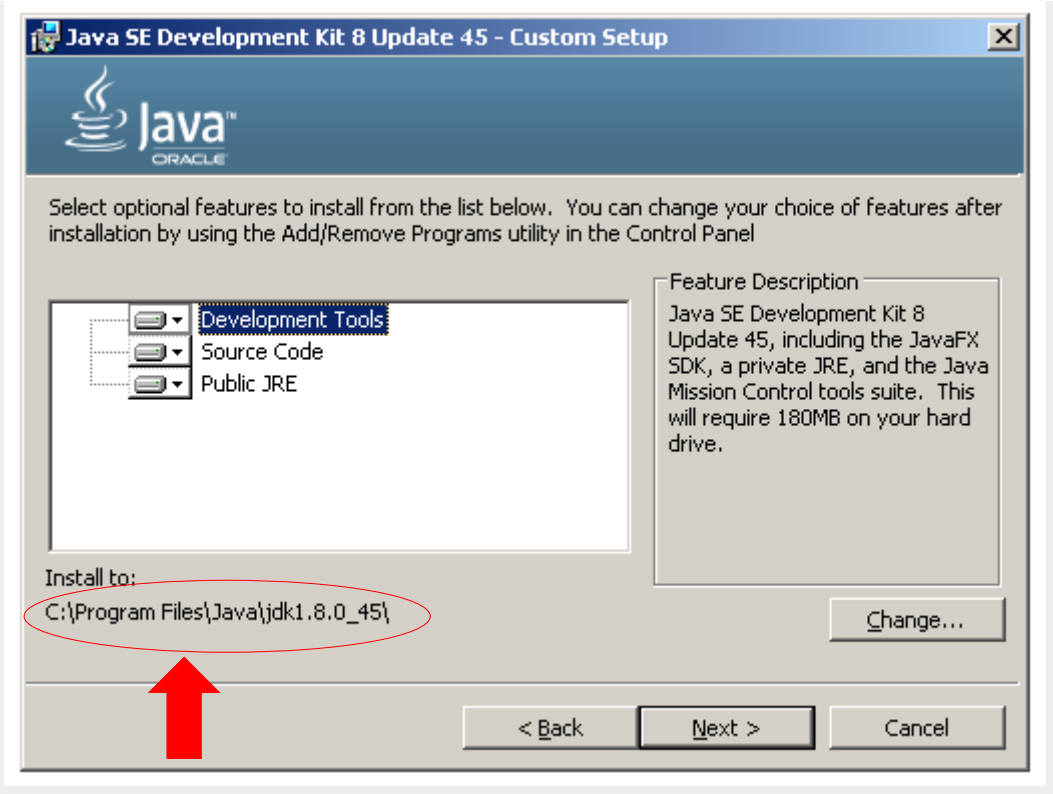


Figura 32 Tela do instalador JDK
Fonte: Matera Systems

Será feita a instalação do JRE (*Java Runtime Environment*), juntamente com o JDK, que é o aplicativo que permite executar os programas em JAVA no seu sistema operacional.

Com a instalação concluída é hora de configurar o Java no Windows, para isso deve-se clicar com o botão direito em Computador e seguir as opções *Propriedades > Configurações avançadas do sistema > (aba Avançado) Variáveis de Ambiente*

Criar e editar as variáveis do sistema listadas nas próximas linhas:

- JAVA_HOME - informe o diretório da instalação do JDK conforme mostra a figura 33.

Na figura o diretório está no endereço *C:\Program Files\Java\jdk1.8.0_045*, esse diretório é informado durante a instalação. A numeração final pode variar de acordo com a versão instalada.

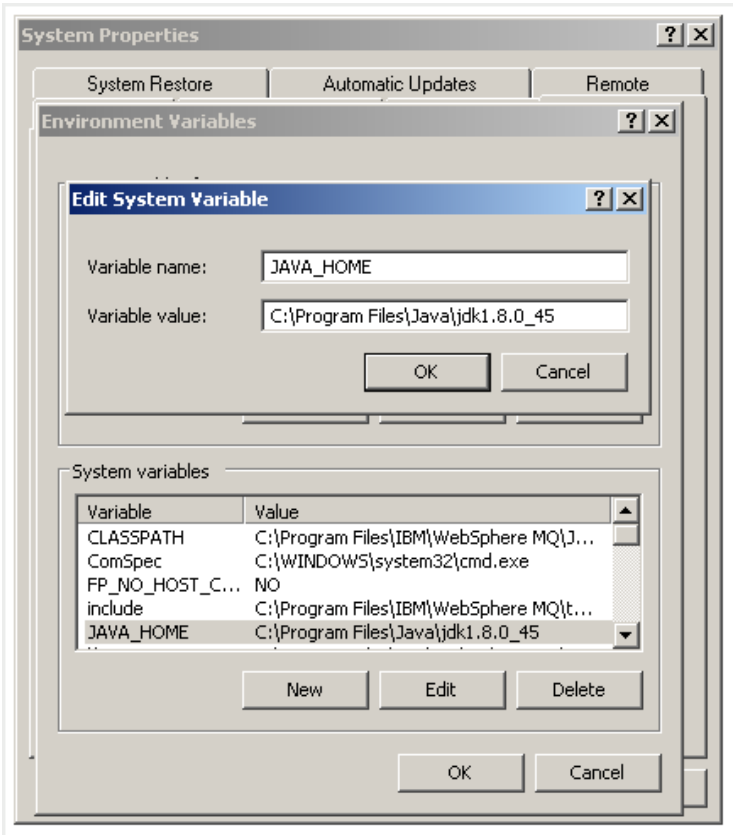


Figura 33 Configuração da variável "JAVA_HOME"
Fonte: Matera Systems

- Path: informe *%JAVA_HOME%\bin* conforme a figura 34

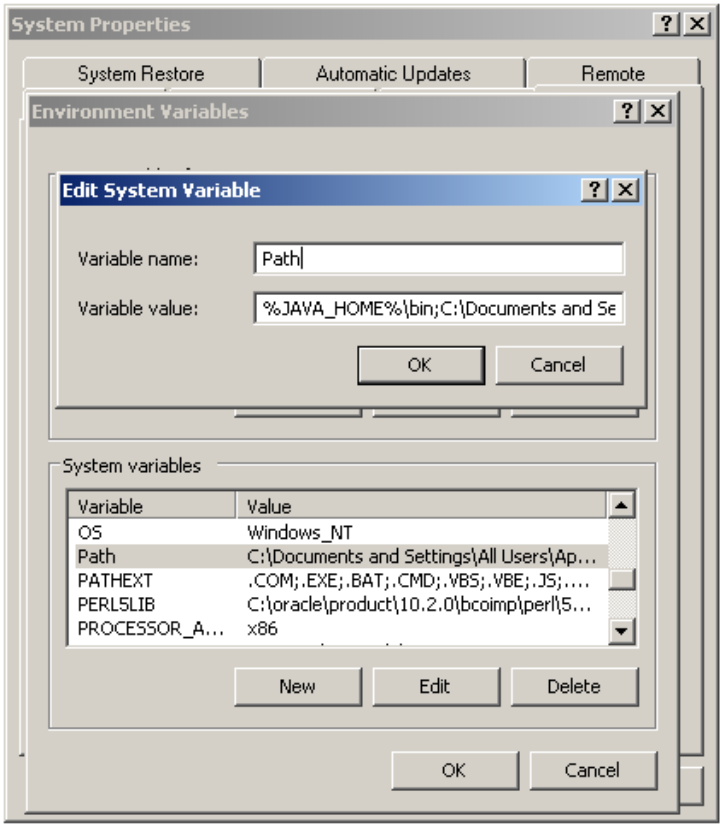


Figura 34 Configuração da variável "Path"
Fonte: Matera Systems

- CLASSPATH: informe *%JAVA_HOME%\lib\tools.jar* conforme mostra a figura 35

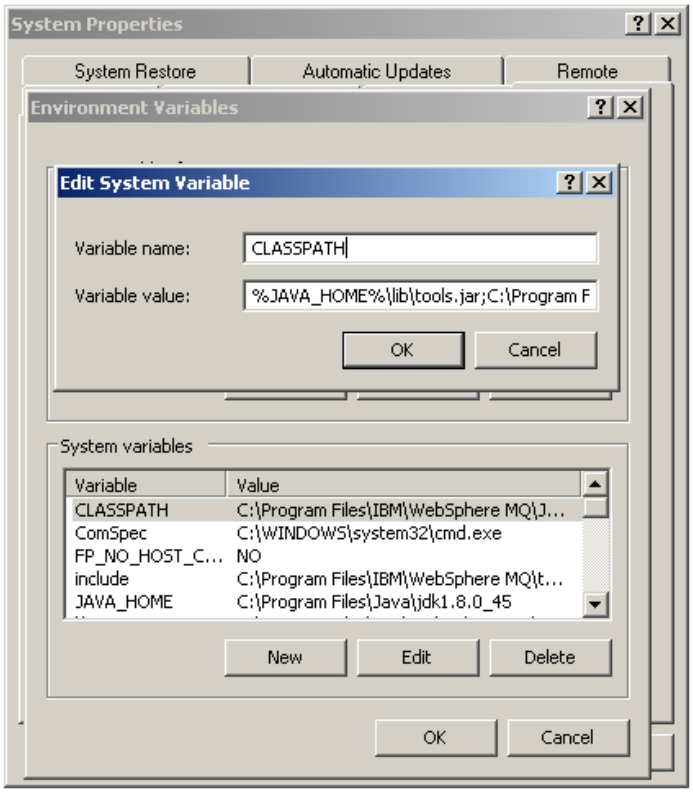


Figura 35 configuração da variável "CLASSPATH"
Fonte: Matera Systems

Para testar a instalação e configuração é necessário executar os comandos "*java -version*" e "*javac -version*" no prompt de comando. Para abrir o prompt de comando digitar CMD na barra de pesquisa, caso esteja utilizando o sistema operacional Windows 10, ou na opção executar em outras versões do Windows. Após executar os comandos espera-se o resultado mostrado na figura 36:

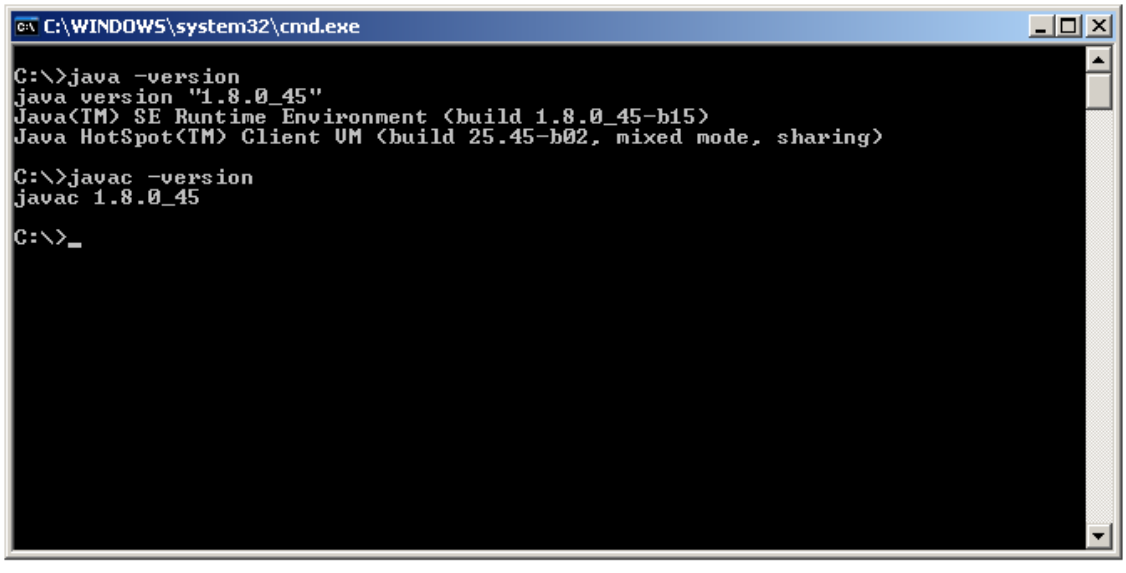


Figura 36 resultado do teste de instalação e configuração JAVA
Fonte: Matera Systems

7.2 Instalação do IDE Eclipse

O processo se inicia baixando o instalador no endereço: http://eclipse.org/downloads/

Assim como o Java, recomenda-se utilizar a versão 32 bits. Com o instalador no computador executar o arquivo *eclipse.exe* para iniciar a instalação. Ao iniciar o eclipse definir o *workspace* de trabalho que é o nome dado a pasta onde será mantido seu espaço de trabalho, conforme mostra a Figura 37.

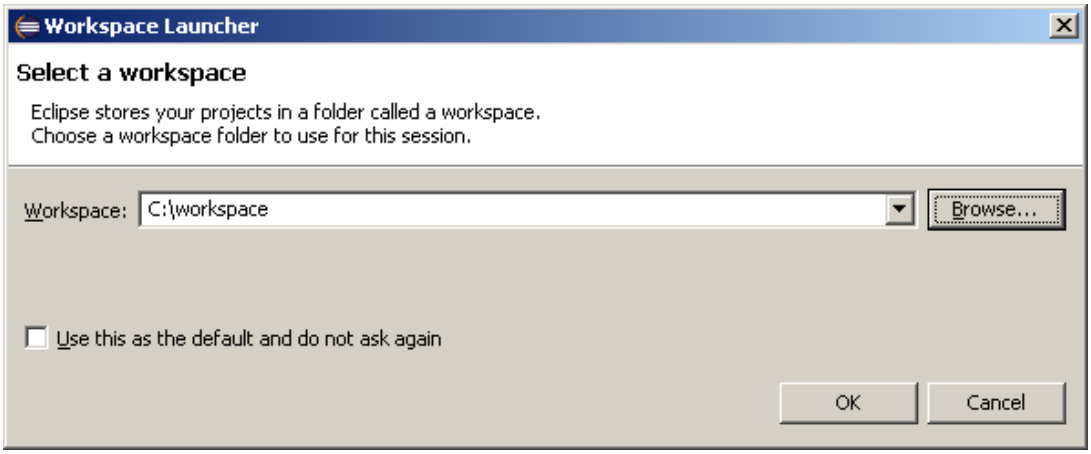


Figura 37 Definindo o Workspace do IDE
Fonte: Matera Systems

Após a definição do *workspace* a instalação do programa poderá ser finalizada. Ao abrir o programa pela primeira vez aparecerá a tela mostrada na figura 38.

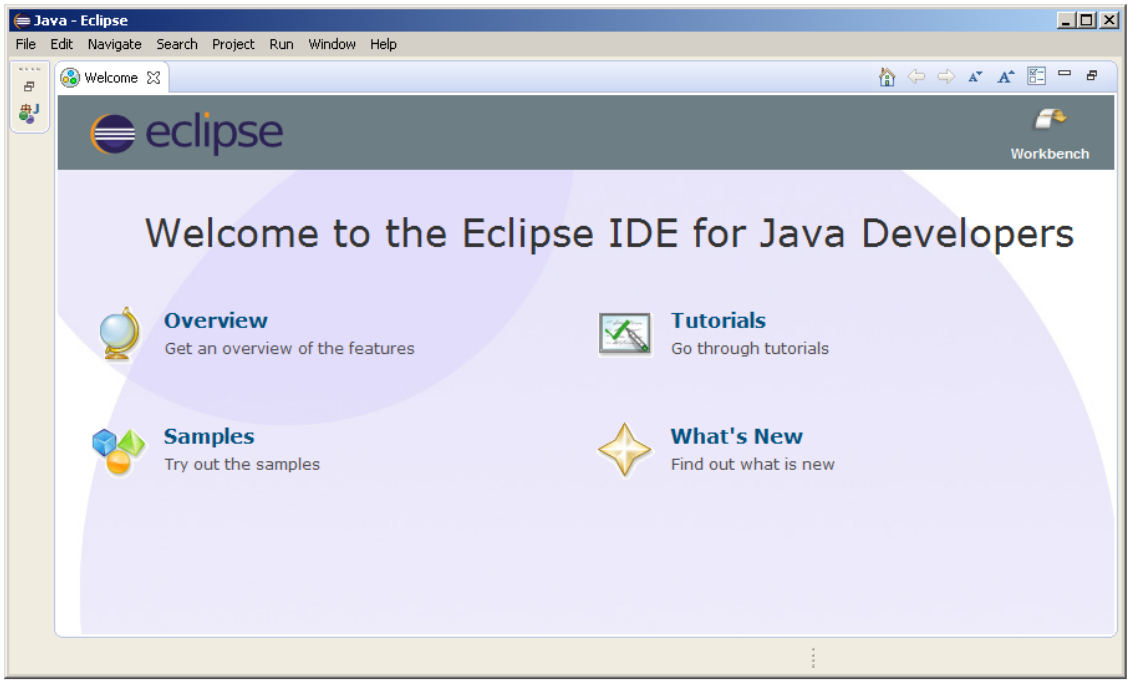


Figura 38 Tela inicial do Eclipse
Fonte: Matera Systems

7.3 Instalação do *LeJOS* e Drives do controlador NXT do kit Lego Mindstorm.

Para instalar o *LeJOS* e poder programar o robô utilizando a linguagem Java é necessário ter os drives adequados instalados no computador. Os drives podem ser encontrados na página da Lego utilizando através do endereço: https://www.lego.com/en-us/mindstorms/downloads. Nesta página encontrar as opções de download dos softwares NXT para Mac e PC e baixar o *"NXT Fantom Driver".* Outros softwares estão disponíveis nesta página, incluindo um para programação em blocos que não será abordado nesse trabalho.

O "*NXT Fantom Driver"* será baixado em um arquivo compactado (*.zip*) após descompactar o arquivo encontrar o aplicativo *"setup"* dentro da pasta conforme visto na figura 39, lembrando que a localização da pasta poderá variar de acordo com o local em que foi descompactada:

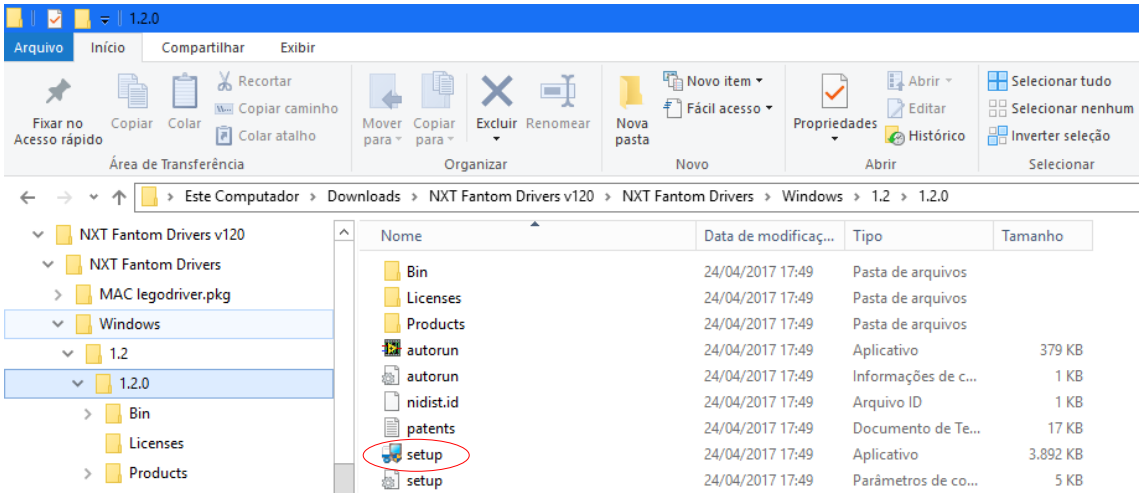


Figura 39 Localização do setup do "NXT Fantom Driver
Fonte: Autoria Própria

A instalação é feita de maneira intuitiva basta seguir as instruções mostradas na tela ao executar o *setup*.

Quando a instalação for concluída é o momento de instalar o *LeJOS NXT* que pode ser baixado na página de *download* com o endereço: http://www.lejos.org/nxj-downloads.php. Ao executar o instalador para Windows aparecerá a seguinte tela da figura 40:

Figura 40 tela inicial do instalador LeJOS NXT
Fonte: Autoria Própria

Clique em "Avançar". Neste momento será verificado se o driver fantom está instalado e atualizado. Caso não esteja aparecerá um erro na tela e poderá ser escolhido abrir a página de download do *driver Fantom* ou continuar sem instalar ou atualizar o *driver Fantom*. Sem o *driver Fantom* a comunicação USB com o NXT pode não funcionar corretamente. Se a instalação continuar, será visto a próxima página do instalador, mostrado na figura 41.

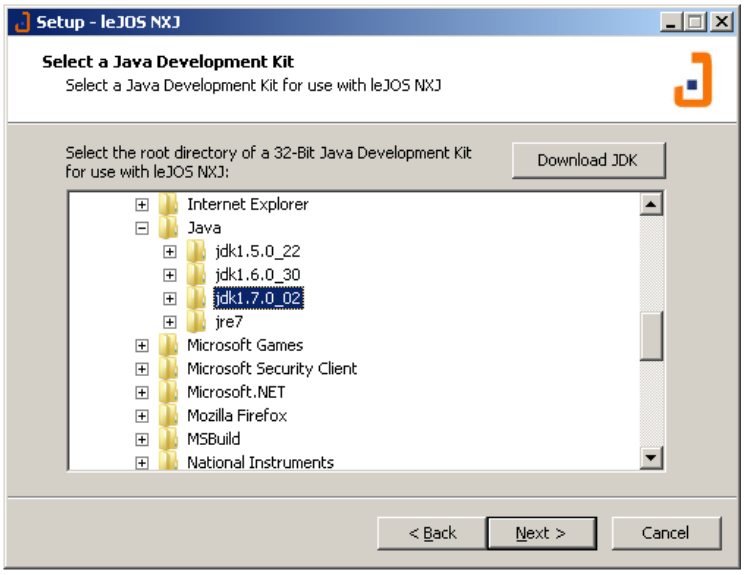


Figura 41 Solicitação para selecionar pasta raiz do JDK
Fonte: LeJOS

Esta página solicita a seleção da pasta raiz de um JDK (*Java Development Kit*) de 32 bits. Não será possível continuar o processo de instalação sem fazê-lo. A pasta selecionada será atribuída automaticamente à variável de ambiente

LEJOS_NXT_JAVA_HOME no final do processo de instalação. O próximo passo é selecionar o diretório de destino, no qual o *leJOS* será instalado, conforme figura 42.

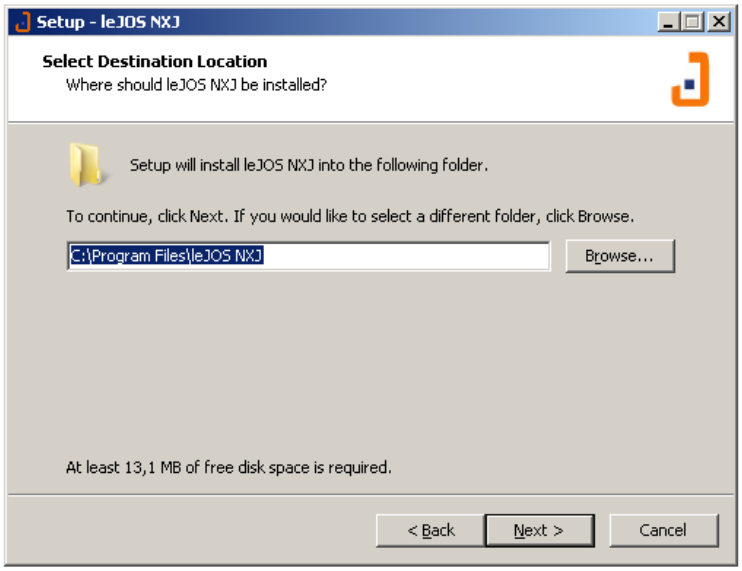


Figura 42 inserindo o diretório destino do LeJOS NXT
Fonte: LeJOS

O destino poderá ser mudado caso seja necessário, clicando no botão "Procurar" para alterar o destino. Na próxima página, será feito a escolha de quais componentes instalar (figura 43):

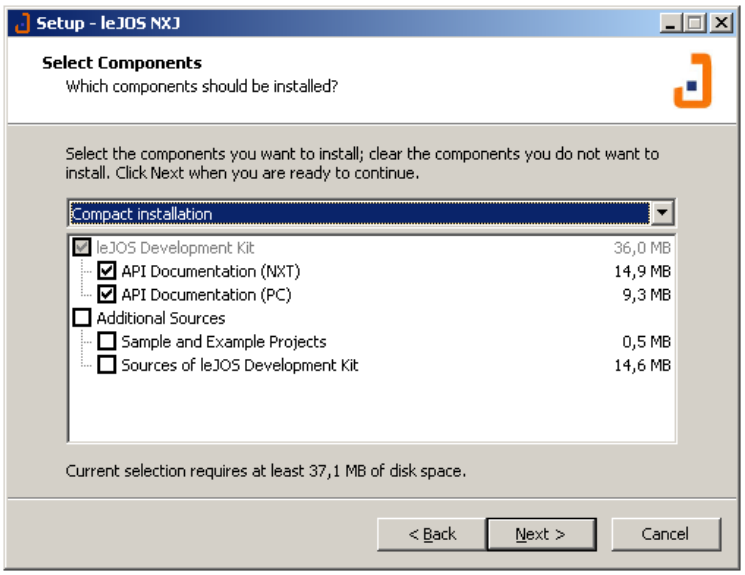


Figura 43 seleção de componentes a instalar
Fonte: LeJOS

Para as aplicações deste trabalho não é necessário fazer nenhuma modificação nessa página. Com isso a página mostrada na figura 44 aparecerá, permitindo ajustar a pasta no menu Iniciar, que os atalhos *LeJOS* serão instalados:

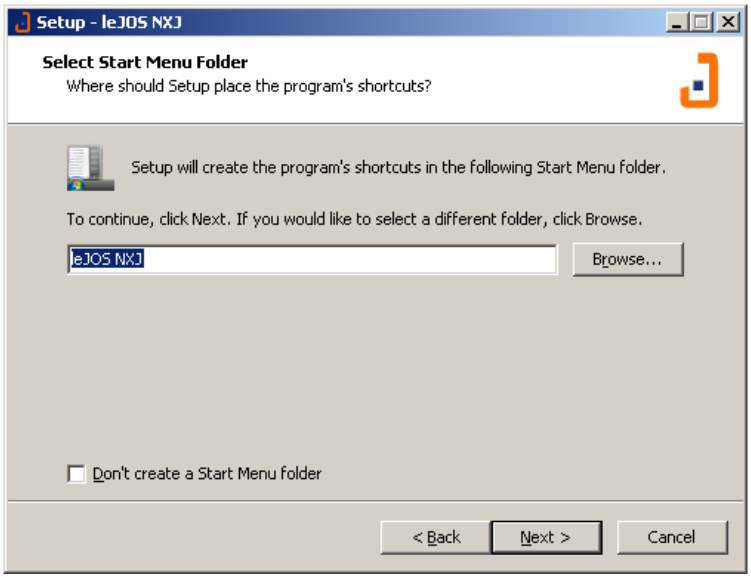


Figura 44 ajuste dos atalhos LeJOS
Fonte: LeJOS

Clicando em *next* é mostrado um resumo das escolhas feitas para a instalação conforme visto na figura 45.

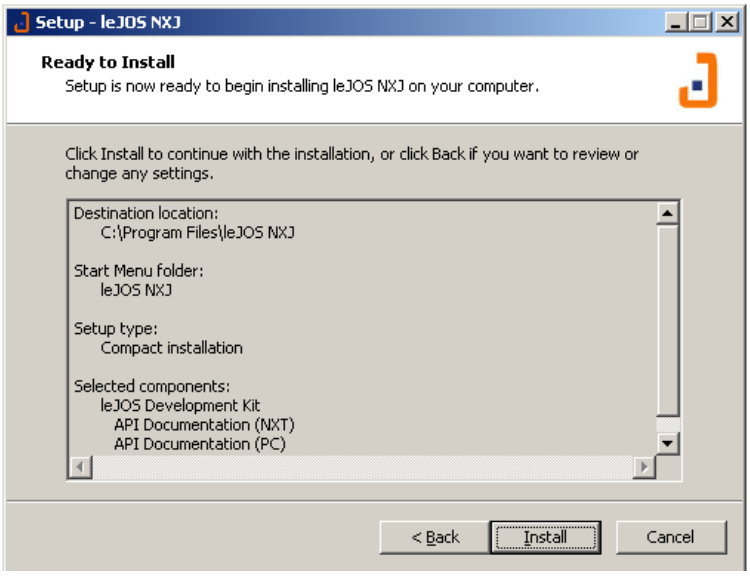


Figura 45 resumo das escolhas para instalação
Fonte: LeJOS

Ao clicar em "Instalar" começará a instalação como mostra a figura 46:

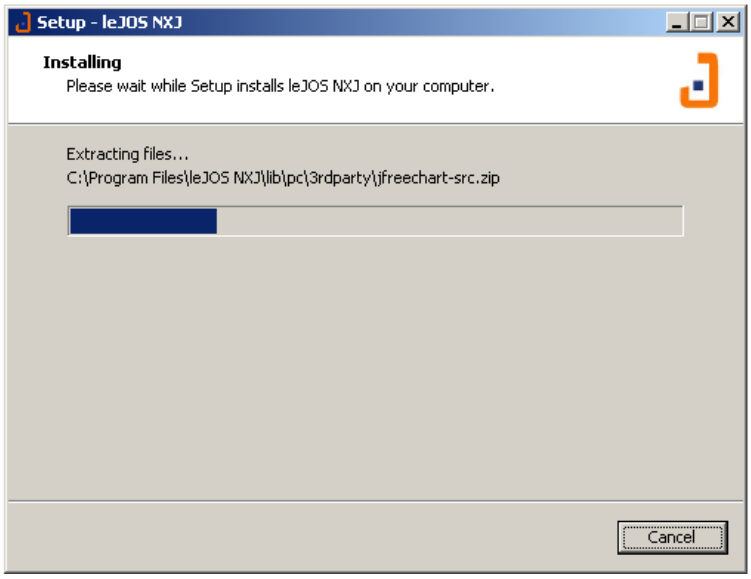


Figura 46 Processo de instalação iniciado
Fonte: LeJOS

A página mostrada na figura 47 aparecerá após a conclusão da instalação:

Figura 47 Tela final de instalação do LeJOS NXT
Fonte: LeJOS

Basta desmarcar a opção *"Launch Flash utility"* e finalizar a instalação clicando em *"finish".*

O último requisito para programação do Lego Mindstorm em Java é instalar o plugin do LeJOS NXT para Eclipse. Para isso ir até o menu *Help, Install New Software* (figura 48):

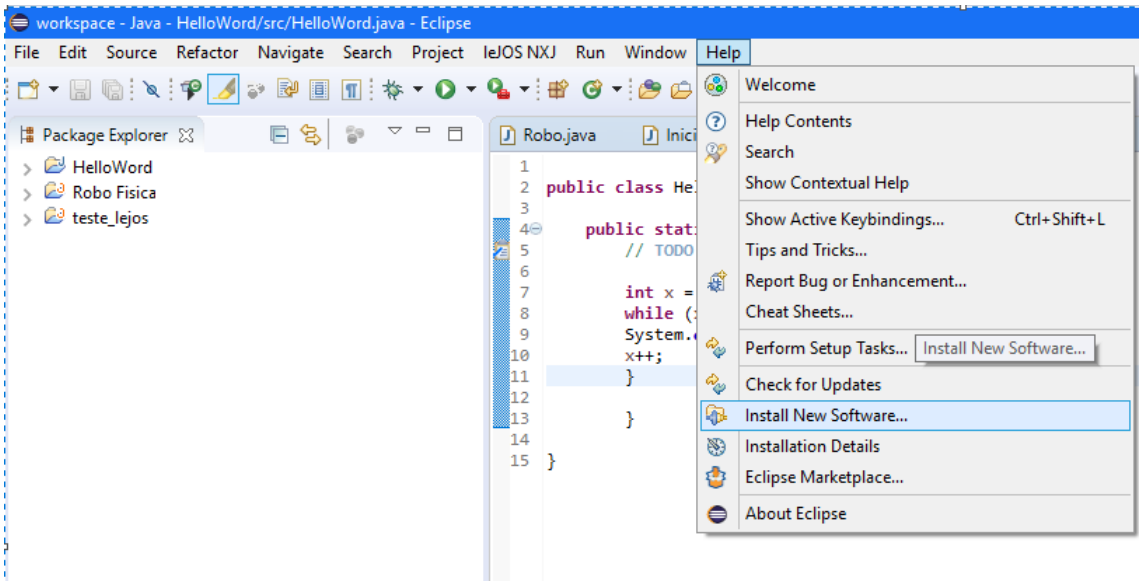


Figura 48 Menu Help, Install New Software
Fonte: Autor

A tela da figura 49 irá aparecer. Preencher o campo *"Work whith:"* com o endereço http://www.lejos.org/tools/eclipse/plugin/nxj/ e selecionar o item *"LeJOS NXT Support".* Após isso basta seguir as instruções (*next e finish*) e o plugin será instalado.

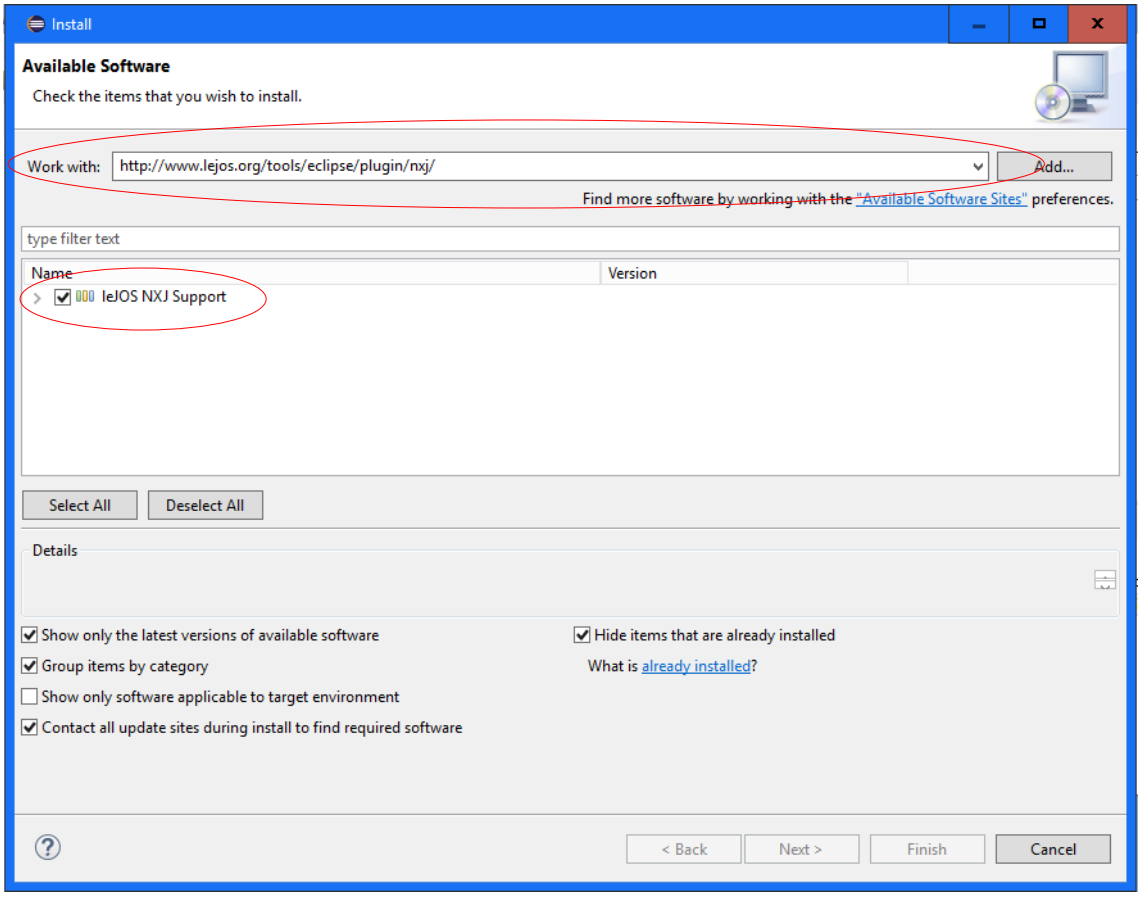


Figura 49 Tela de instalação de plugins do Eclipse
Fonte: Autor

7.4 Programação do Lego Mindstorm Utilizando o Eclipse

O primeiro passo para programar o robô é criar um projeto Java na IDE, para isso após abrir o programa deve se seguir os seguintes passos:

Ir até o menu *File, New, Java Project* (figura 50)

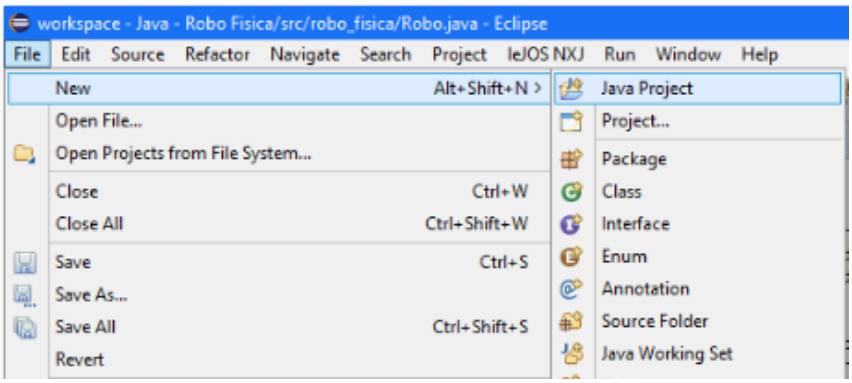


Figura 50 menu File
Fonte: Autoria Própria

A tela mostrada na figura 51 irá se abrir. Colocar o nome do projeto e clicar em *next*;

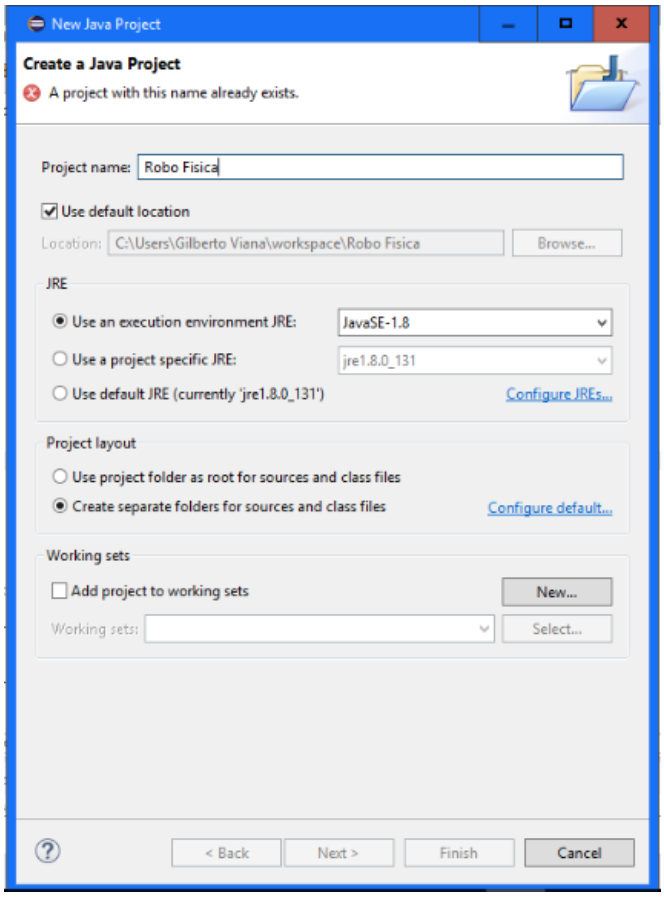


Figura 51 primeira janela a abrir durante criação do projeto Java
Fonte: Autoria Própria

Irá abrir a tela de configuração do projeto mostrado na figura 52. Clicar em *Finish*;

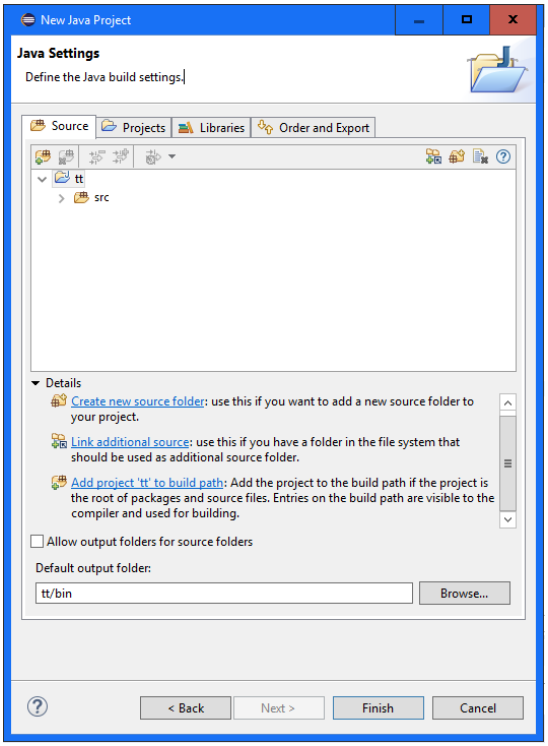


Figura 52 configurações de construção do projeto
Fonte: Autoria Própria

No *software* irá aparecer na guia da IDE o projeto conforme visto na figura 53:

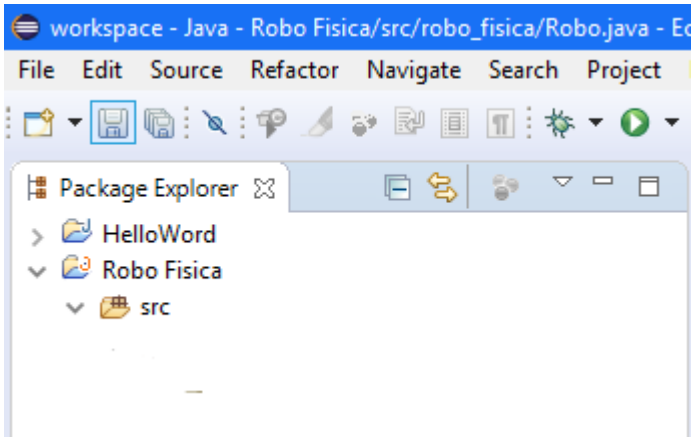


Figura 53 Guia da IDE
Fonte: Autoria Própria

Uma nova classe deverá ser criada clicando com o botão direito no nome do projeto e selecionando, no menu de contexto, as opções *New, Class* (figura 54):

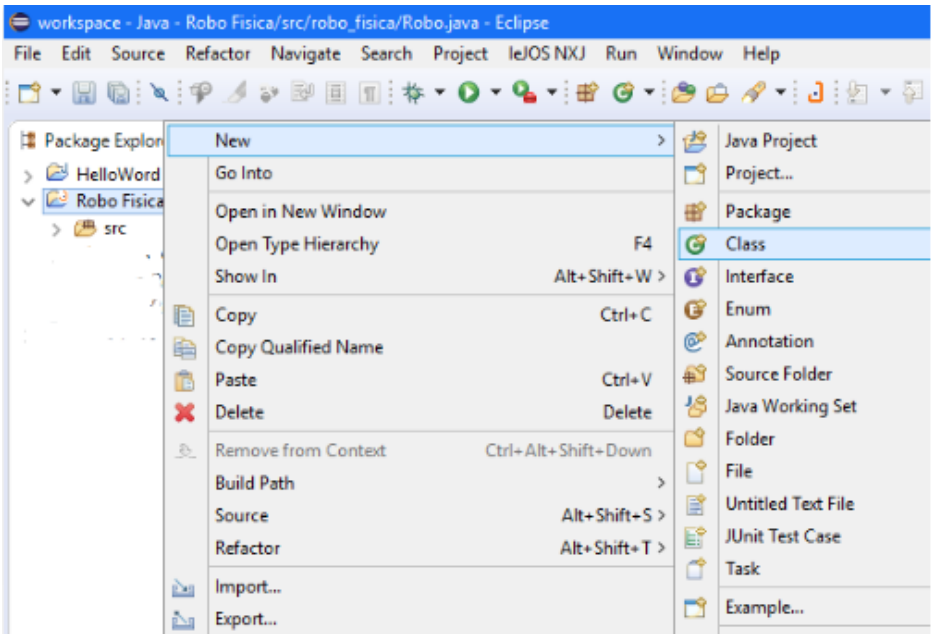


Figura 54 Abas para criação de uma nova classe
Fonte: Autoria Própria

A janela mostrada na figura 55 irá abrir. Preencher os itens *Package* e *Name* e não esquecer de selecionar a opção *public static void main (String[] args)*. Após clicar sobre o botão *Finish* para finalizar e criar a classe.

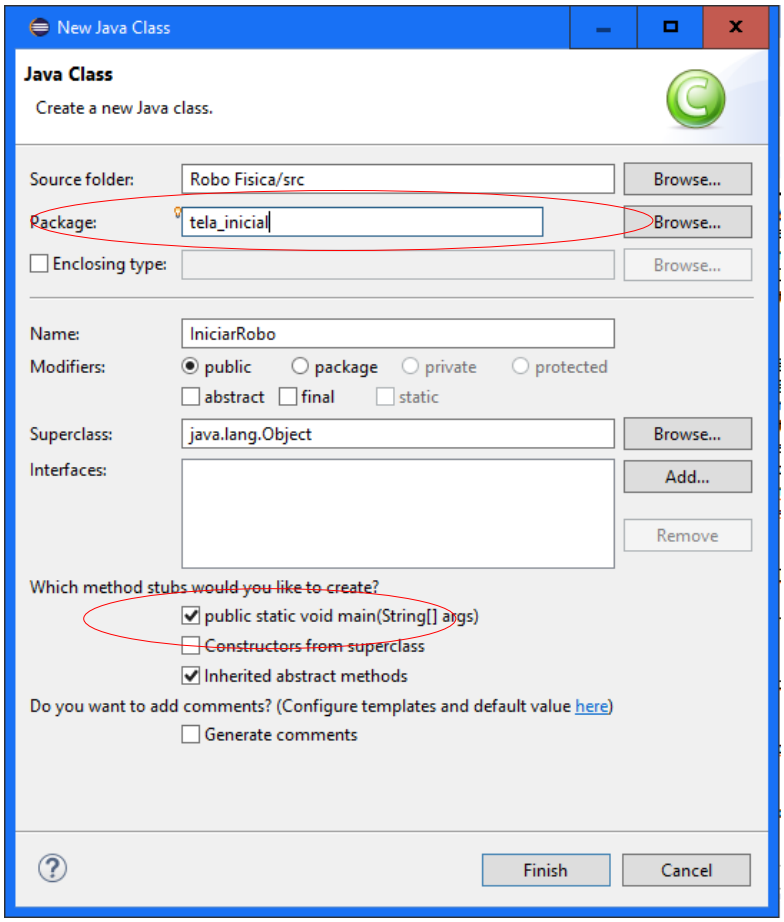


Figura 55 configurações da classe a ser criada
Fonte: Autoria Própria

Os programas em Java, basicamente, seguem a seguinte estrutura: bloco de programas; declaração de variáveis; sintaxes dos comandos. Essa estrutura forma um bloco principal. Esse bloco é denominado classe e dentro dele o programa se desenrola. A nomenclatura da classe segue algumas regras, que não será detalhada no momento, fica a critério do aluno buscar as explicações caso tenha interesse. O importante é saber que na nomenclatura definimos o nome e acesso da classe, como pode ser visto no exemplo abaixo:

```
public class Robo {

    }
```

Nesse exemplo "*public class*" define que é uma classe pública que pode ser acessada por qualquer um. Robo é o nome dado a classe ou a identificação utilizada para chamá-la. Tudo que estiver escrito entre as chaves, que indicam o início e fim do bloco, será executado ao chamar a classe Robo.

Para que as ações possam ser executadas é necessário acrescentar as funções, que em Java são chamados de métodos. Todas as funções ocorrem dentro dos métodos. O método principal *(main()*) é definido da seguinte forma:

```
public static void main(String[] args) {


}
```

Os parâmetros do método são os seguintes:

- *public: o método main()* pode ser acessado por qualquer Classe;
- *static*: informa ao compilador que main() não requer a chamada de uma instância desta Classe;
- *void*: indica que nada é retornado por main();
- *String [ ] args*: declaração da matriz tipo *String*, que recebe os parâmetros.

Como o objetivo não é se aprofundar em programação, não é necessário se preocupar com a explicação dos parâmetros contidos nessa nomenclatura, basta saber que o programa começa por aí.

A classe e o método são as estruturas que formam o bloco de programa, sabendo isso já é possível programar o robô utilizado nos desafios. Dentro do programa aparecerão outros blocos de comando, necessários para que o robô tome a decisão correta para concluir as ações. A seguir será analisado a estrutura de um exemplo de programação em Java que pode ser utilizado no robô. A estrutura mostrada na figura 56 foi desenvolvida após criar o projeto na IDE Java Eclipse Neon. Para esse projeto foram acrescentadas duas classes conforme veremos a seguir:

Robo.java IniciarRobo.java

```
1  package robo_fisica;
2
3  import lejos.nxt.Button;
4
5  public class IniciarRobo {
6
7      public static void main(String[] args) {
8          // TODO Auto-generated method stub
9          Robo r = new Robo();
10
11         r.calibrar();
12
13         System.out.println("Pressione um botao para iniciar");
14         Button.waitForAnyPress();
15
16         while(!Button.ESCAPE.isDown()) {
17             r.mostrarLeitura();
18             r.testar();
19
20         }
21     }
22
23 }
```

Figura 56 Classe "IniciarRobo"
Fonte: Autoria Própria

A primeira classe, mostrada na figura 56, foi criada com o nome "*IniciarRobo*" e contém o método *main().* Na linha 1 encontra-se o “pacote” onde se encontra o bloco de programação. Na linha 3 o *import* é utilizado para importar classes de bibliotecas prontas do Java. Na linha 5 se inicia a classe *IniciarRobo*. A linha 9 contém os atributos dessa classe e após está o método que será executado. Dentro desse método: na linha 11 é chamado o método calibrar, que será visto na descrição de outra classe; na linha 13, está o comando que faz aparecer na tela a mensagem "Pressione um botão para iniciar"; na linha 14 o comando faz com que o controlador aguarde um botão ser pressionado para iniciar o programa. Na linha 16 o comando “*while*” cria uma repetição que faz o robô executar outros dois métodos (*MostrarLeitura* e *testar*) enquanto o botão escape não for pressionado. Nas linhas 20, 21 e 23 estão as chaves que fecham os blocos de comandos, método e classe. Os métodos chamados até aqui serão descritos na próxima classe que compõe a programação deste robô.

Até este momento foi descrito o que o robô irá fazer ao ser iniciado e mostrado na tela, para outras ações foi criado outra classe, mostrada na figura 57, denominada *Robo* que está descrita na sequência deste trabalho:

Robo.java | IniciarRobo.java

```java
package robo_fisica;

import lejos.nxt.Button;

public class Robo {
        // Atributos da classe
        private LightSensor sensorLuz = new LightSensor(SensorPort.S1);

        // Métodos
        public void mostrarLeitura() {
            try {
                System.out.println("Leitura: " + sensorLuz.getLightValue());
                Thread.sleep(75);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

        public void andarFrente() {
            Motor.A.forward();
            Motor.B.forward();
        }

        public void rotacionar() {
            Motor.A.backward();
            Motor.B.forward();
        }

        public void parar() {
            Motor.A.stop();
            Motor.B.stop();
        }

        public void testar() {
            if(sensorLuz.getLightValue() > 50) {
                Motor.A.setSpeed(51);
                Motor.B.setSpeed(51);
                andarFrente();


                }

            else {
                if(sensorLuz.getLightValue() < 50) {
                    parar();
                }
                            }
        }

        public void calibrar() {
            System.out.println("Calibrar PRETO.");
            Button.waitForAnyPress();
            sensorLuz.calibrateLow();
            System.out.println("Calibrar BRANCO.");
            Button.waitForAnyPress();
            sensorLuz.calibrateHigh();
        }
    }
```

Figura 57 Classe "Robo"
Fonte: Autoria Própria

Nessa classe estão todas as instruções das ações que serão realizadas pelo robô. Na linha 3 está o comando que importa as bibliotecas necessárias, este comando é preenchido automaticamente pelo *software*. Na linha 9 se inicia a classe *Robo* e na linha 10 se encontra um comentário descrevendo que abaixo dessa linha estão os atributos da classe. O atributo, neste caso o sensor de luz, é a variável declarada que fornecerá o parâmetro de comparação para as decisões e ações tomadas. A palavra "*private*" antes dos atributos declara que estes não podem ser acessados fora desta classe.

A partir da linha 14 começam os métodos utilizados no programa. O primeiro método é denominado "*mostrarLeitura*" que contém a seguinte linha comando: *System.out.println("Leitura: " + sensorLuz.getLightValue())*. Essa linha (linha 16) é responsável por exibir na tela o valor lido pelo sensor de luz, sendo esse o objetivo desse método. Os outros conteúdos desse primeiro método servem, basicamente, para determinar o intervalo em que será feito a leitura do sensor. O próximo método, escrito a partir da linha 24, descreve como os motores deverão funcionar e fazer o robô andar para frente. Na linha 29 se inicia o método adotado para o robô fazer uma curva e na linha 34 começa o método para o robô parar. A linha 34 é o início do método testar responsável pela decisão do que o robô irá fazer durante o desafio. Para essa decisão são utilizados os comandos *"if"* e *"else"* que apresentam a seguinte sintaxe:

```
if (expressão booleana) {
    // bloco de código 1
}
else {
    // bloco de código 2
}
```

O método "*testar*" faz a leitura do sensor de luz e se este valor for maior que 50 a velocidade dos motores A e B serão ajustadas em 51°/s e o método "*andarFrente*" será executado. Essas ações estão descritas nas linhas 34 a 46. Se o primeiro teste não for verdadeiro, o comando passa para a segunda análise que começa na linha 48 com o comando *"else".* A partir dessa linha ele faz a leitura e se o valor do sensor for menor que 50 o método "*parar*" será executado.

Após o método testar está o método calibrar, a partir da linha 55, responsável por fazer uma calibração do sensor correspondentes ao branco e preto projetado nas pistas durante os desafios. Esse método é chamado ao iniciar o robô na classe *IniciarRobo*, mais especificamente na linha 16 da mesma.

Com as classes e métodos descritos o robô é capaz de identificar as linhas e objetos que compõe as pistas virtuais e tomar a decisão de andar quando estiver em uma superfície projetada branca e parar quando entrar em contato com um objeto com preenchimento preto. Essas são as condições básicas necessárias para cumprir os desafios, devendo o aluno após análise de cada desafio modificar os parâmetros de velocidade (linhas 41 e 42 do método "*Robo*"), e/ou acrescentar comandos de aceleração e tempo de pausa no movimento do robô.

Após criar o programa é necessário transferi-lo para o controlador NXT. Para isso o controlador deve ser ligado apertando o botão central na cor laranja e conectado ao computador através da porta USB. Feito isso na tela do IDE Eclipse clicar na seta ao lado do botão *Run* ir na aba "*Run as"* e escolher a opção "*LeJOS NXT Program"* conforme mostra a figura 58. O programa será gravado na memória do controlador e quando terminado as primeiras instruções do robô irá abrir na tela do controlador pedindo para calibrar os sensores. Para isso basta posicionar o robô com o sensor recebendo a projeção da pista com a cor a ser calibrada e apertar "Enter". Após a calibração o robô estará pronto para iniciar os desafios de Física.

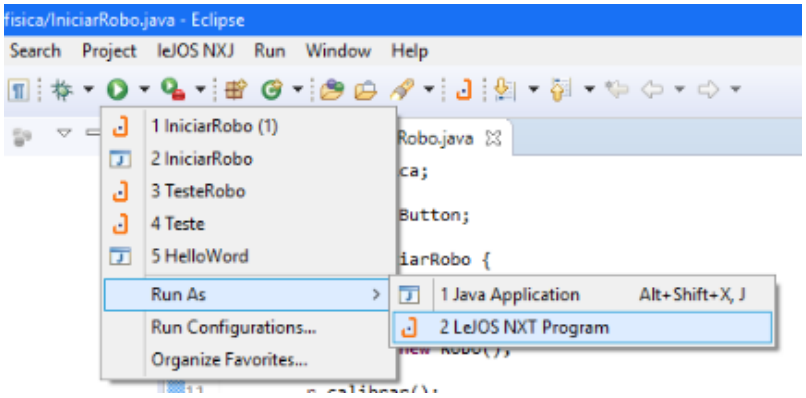


Figura 58 opção "run as" contida no botão "Run"
Fonte: Autoria Própria

Outros comandos podem ser necessários para realizar os desafios. O comando para definir a aceleração do motor A, por exemplo, tem a seguinte sintaxe:

```
Motor.A.setAcceleration(acceleration);
```

No parâmetro "*acceleration*" entre parênteses deve ser digitado o valor da aceleração desejada em graus por segundo ao quadrado. Para programar a aceleração do motor B, basta mudar a escrita para:

```
Motor.B.setAcceleration(acceleration);
```

Para acrescentar que o motor fique parado por exemplo, utiliza-se a seguinte sintaxe:

```
Thread.sleep(milisegundos);
```

Após digitar esse comando o próprio programa pedirá para completar o bloco de comando, ficando da seguinte forma:

```
try {
    Thread.sleep(milisegundos);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Não será aprofundado a explicação dessas linhas adicionais, pois esse não é o objetivo do trabalho. Lembrando que nos parênteses é colocado o valor do tempo de pausa em milissegundos.

Outras opções de controle dos motores e sensores para o *Lego Mindstorm* programado em Java podem ser encontradas no endereço eletrônico: http://www.lejos.org/rcx/api/index.html. Com o que foi visto até agora já é possível programar o robô para executar as tarefas e cumprir os desafios de Física propostos no próximo capítulo.

8. DESAFIOS

Até agora foram descritos a parte de montagem e programação do robô. Neste capítulo estão as descrições das pistas que formam o ambiente virtual em que o robô deverá interagir com os objetos para cumprir os desafios.

Para cumprir cada desafio será necessário conhecer um conceito Físico. Esses conceitos podem ser encontrados na literatura utilizada na disciplina de Física, mais especificamente no conteúdo de Cinemática. Cabe ao professor escolher os livros de Física que irá utilizar para o estudo dessa disciplina. Uma análise dos dados fornecidos e variáveis deve ser feita antes de programar o robô. Provavelmente todos os desafios poderão ser concluídos com a mesma programação básica, somente modificando alguns parâmetros na hora de programar as tomadas de decisão do robô.

É necessário prestar atenção nas unidades de medida utilizadas pelo controlador NXT e a necessidade de fazer algumas conversões de unidade para que as tarefas possam ser executadas da maneira correta. Da mesma forma verificar as medidas da pista que podem mudar de acordo com a distância em que está sendo projetada. Para facilitar as tomadas de medida, cada pista tem uma escala de conversão de medidas ou algum referencial que pode ser utilizado como medida padrão.

8.1 Utilizando o programa *Construct 2* na criação de pistas interativas virtuais

Para criar o ambiente virtual ou pistas interativas foi utilizado o programa construct 2. Os passos mostrados abaixo ensinam como fazer a construção e configuração básica das pistas, com isso o professor poderá criar as pistas descritas neste trabalho e criar ambientes de acordo com sua necessidade. As explicações sobre todos os passos para criação de novos ambientes podem ser encontradas no site do desenvolvedor no endereço https://www.scirra.com/tutorials/262/comeando-com-o-construct-2-tutorial-in-portuguese.

A primeira etapa é criar o projeto, para isso clique em *New,* localizado no menu *file* (figura 59), e na janela que se abrirá clique em *New empty Project,* conforme visto na figura 60*.* Na aba vazia, chamada Layout 1 serão colocadas as imagens e objetos.

Figura 59 janela New project
Fonte: www.scirra.com

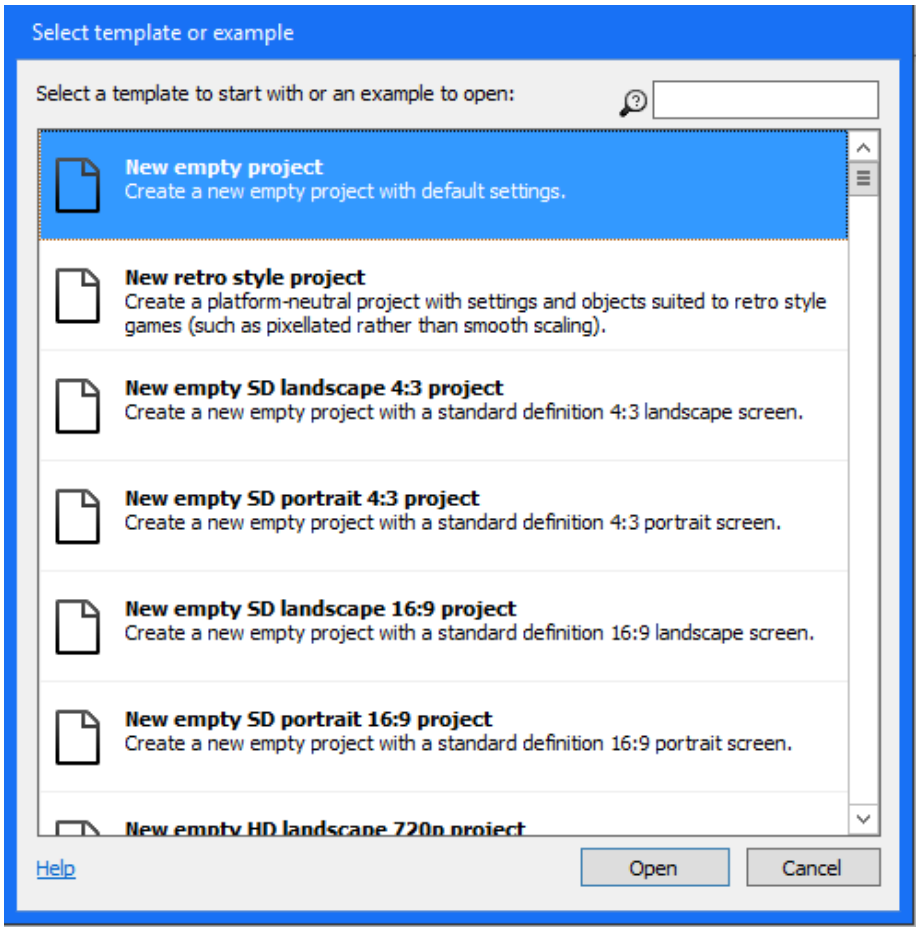


Figura 60 New empty project
Fonte: Autoria Própria

Com o *layout* 1 aberto, podemos inserir os objetos que serão os obstáculos e referências do ambiente virtual. Para inserir um objeto clicar com o botão direito do mouse dentro da aba layout 1 e escolher a opção *Insert new object,* conforme figura 61. Os passos mostram a criação da pista para o primeiro desafio apresentado nesse trabalho, que trata da velocidade média. Após as instruções de programação serão apresentadas as instruções para realização do desafio e outros exemplos de desafios projetados para esse trabalho.

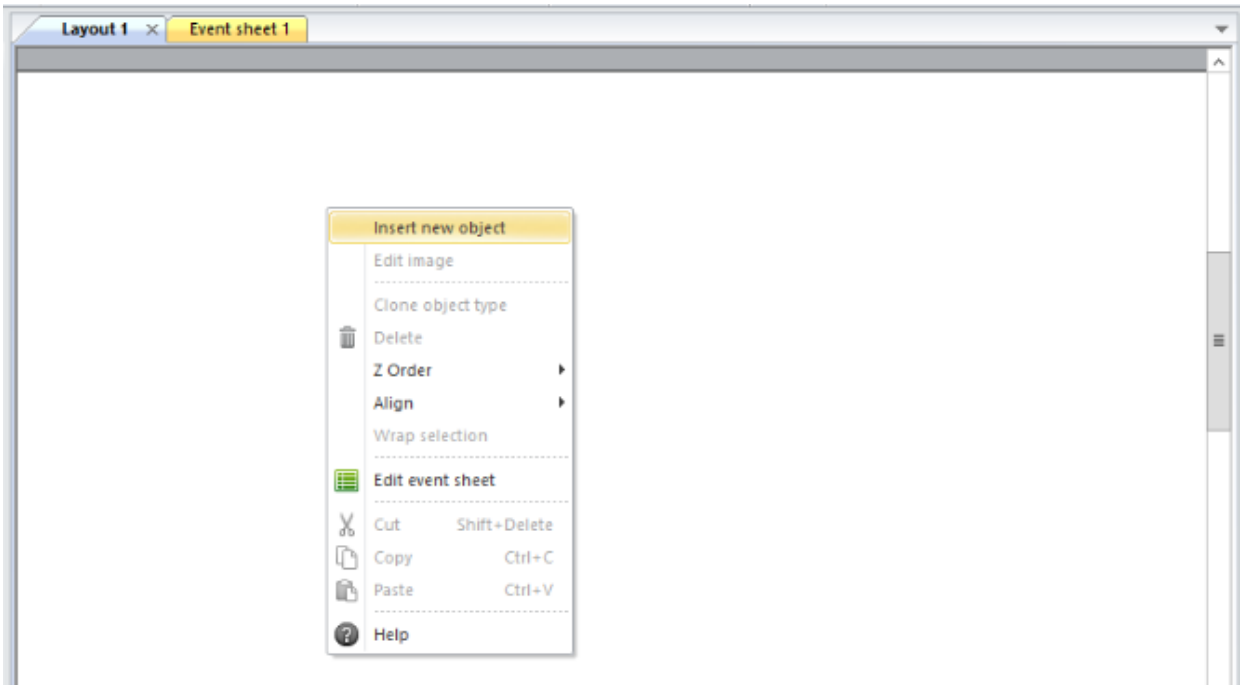


Figura 61 Insert new object
Fonte: Autoria Própria

A janela mostrada na figura 62 aparecerá e nela dar um clique duplo em '*Sprite*'.

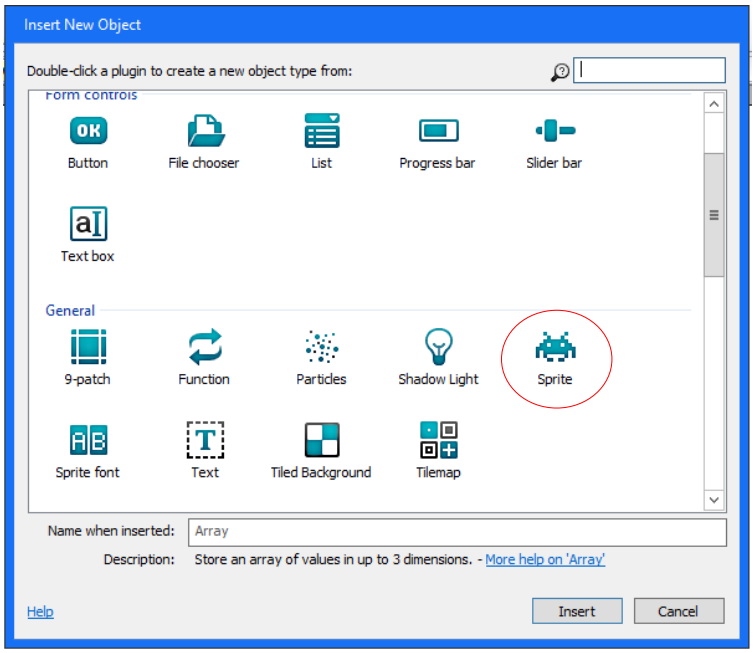


Figura 62 janela de escolha de objetos
Fonte: Autoria Própria

Quando o mouse virar uma cruz, clique em algum lugar do layout. Junto com a cruz aparece uma mensagem mostrando o nome da camada ativa. No nosso caso, estará a *'layer 0'*. O editor de texturas aparece. Clique no icone *fill* (com desenho de balde) e na janela Collor *palet* escolha o preto (figuras 63 e 64). Ao clicar dentro do quadrado desenhado no editor, ele será preenchido com a cor escolhida. O preto é utilizado para facilitar a leitura do sensor do robô.

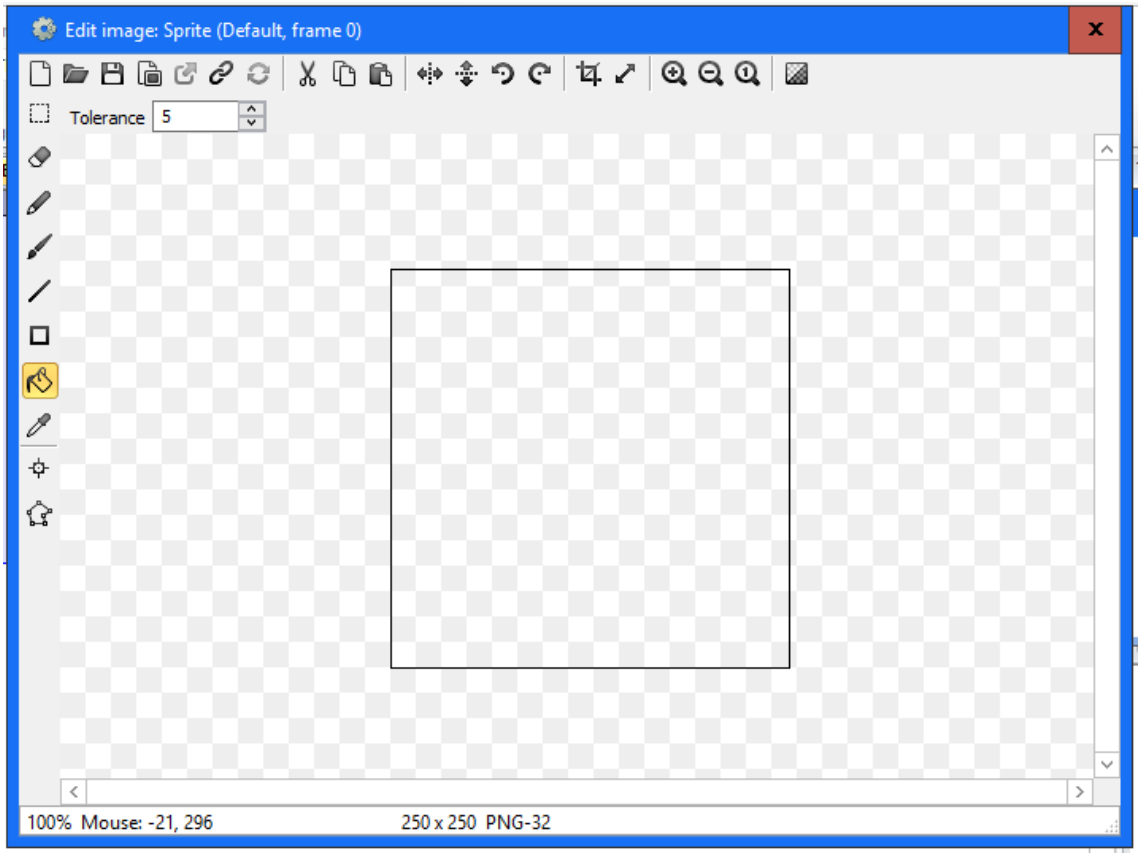


Figura 63 Editor de imagem
Fonte: Autor

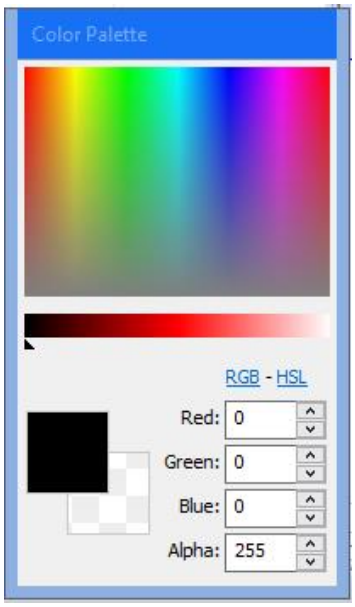


Figura 64 Paleta de cores do editor de imagem
Fonte: Autoria Própria

Ao fechar o editor e salvar suas mudanças, o objeto será visto no layout. Utilizando o mouse poderá ser ajustado a forma e posição do objeto. Para esse exemplo deixar com o formato e posição mostrado na figura 65.

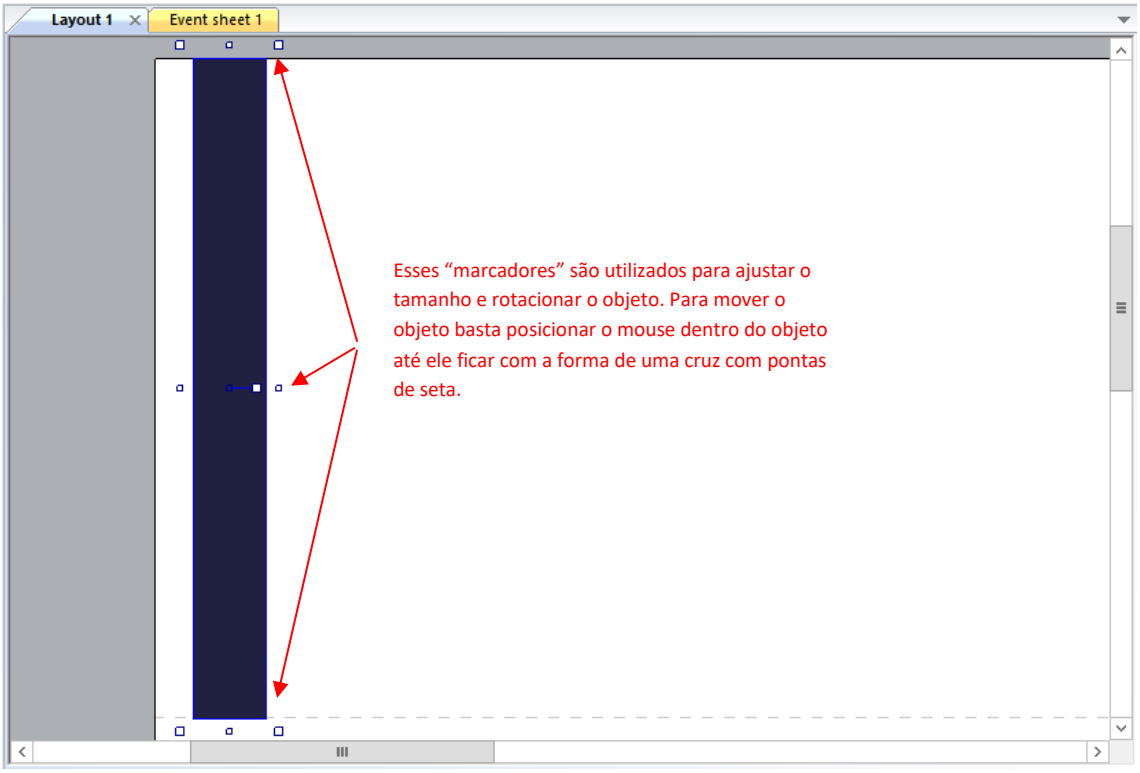


Figura 65 Objeto modificado e posicionado
Fonte: Autor

Para a pista do primeiro desafio, esse processo é repetido por três vezes, criando uma estrutura parecida com a figura 66

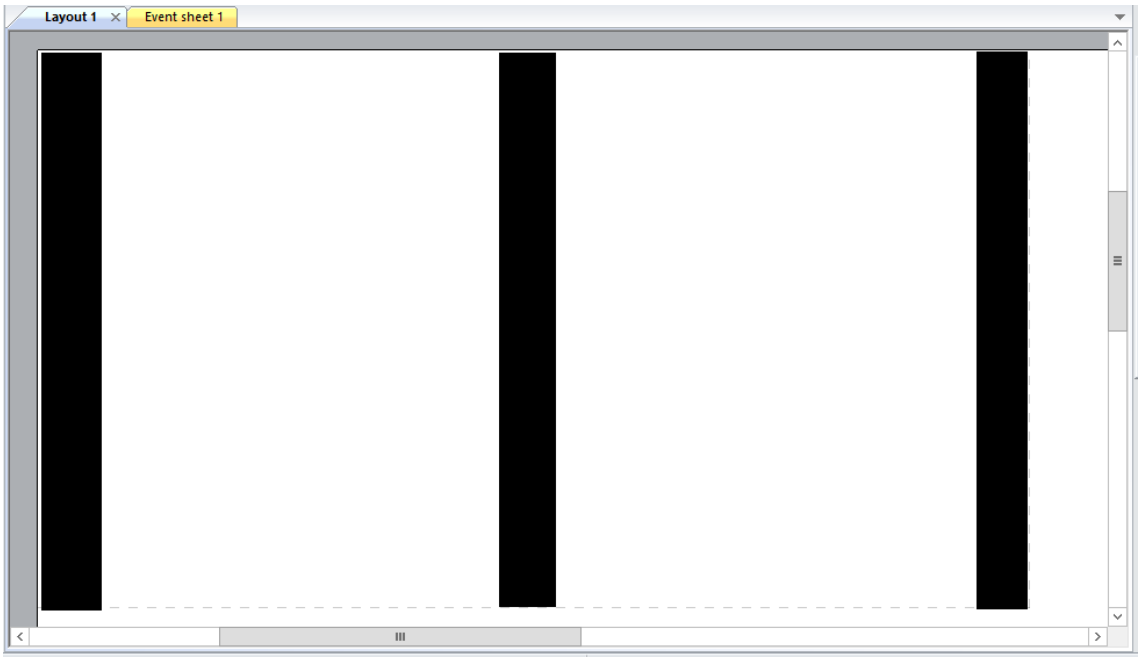


Figura 66 Posição dos objetos no layout
Fonte: Autor

Os objetos estarão com os nomes de *Sprite, Sprite2, Sprite3 e Sprite4*, algo que não é muito funcional. Deverá ser mudado para Largada, intermediaria e *chegada*. Para fazer isso, selecionar o objeto e mudar a propriedade *Name* na barra de propriedades (*Properties bar*), conforme mostra a figura 67.

Figura 67 Barra de propriedades
Fonte: www.scirra.com

A próxima etapa é inserir os botões de controle da pista, clicando com o botão direito do mouse dentro da aba layout 1 e escolher a opção *Insert new object.* Escolher a opção *button.* São necessários dois botões: um para iniciar o desafio e outro para reiniciá-lo. A figura 68 mostra a tela para esta tarefa.

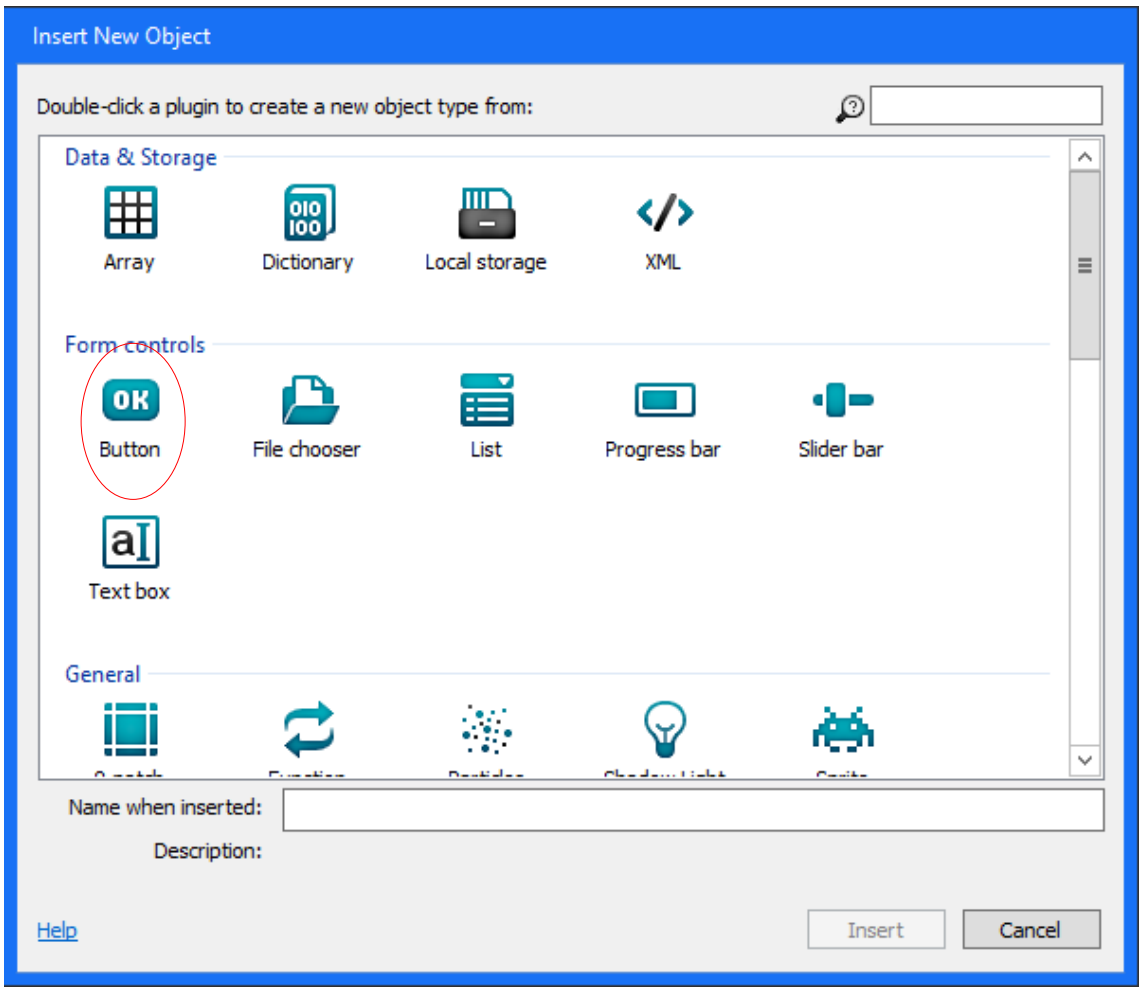


Figura 68 Escolha do botão para controle
Fonte: Autoria Própria

Os botões, por padrão, vêm com o texto *"OK"* e o nome "*Button"*, que pode ser mudado na barra de propriedades (*Properties bar*), conforme visto na figura 69. Mudar para os nomes e textos para "Iniciar" e "Reiniciar" e posicioná-los conforme a figura 70.

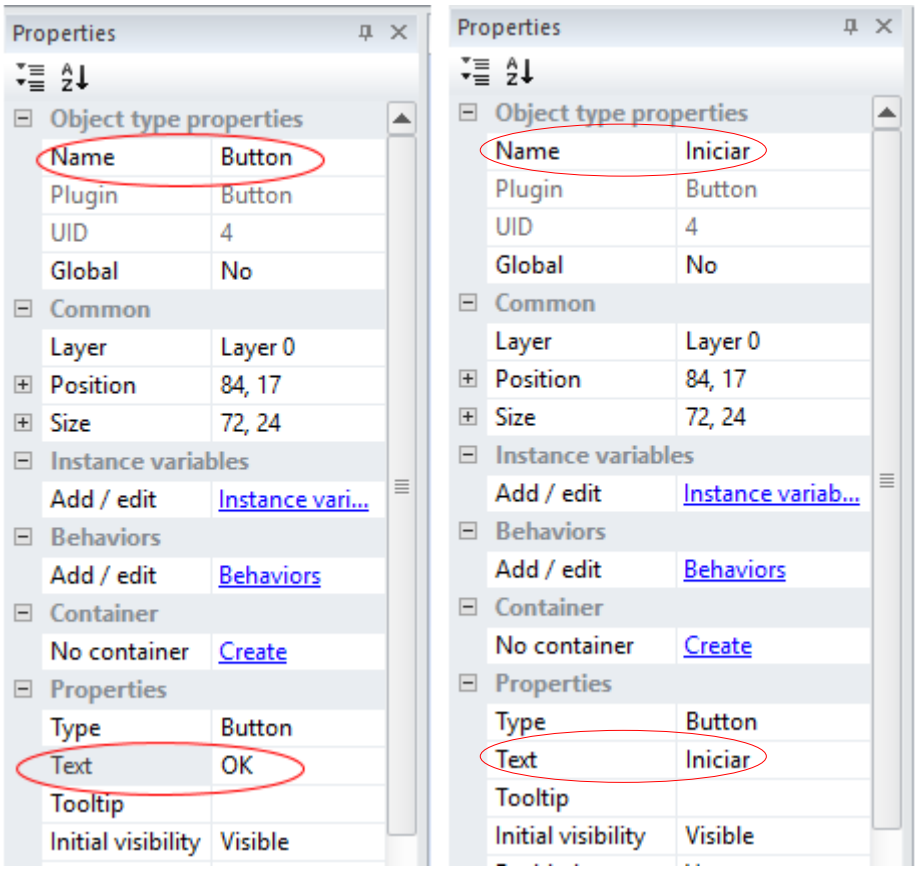


Figura 69 Propriedade do objeto "Button" antes e depois da modificação
Fonte: Autoria Própria

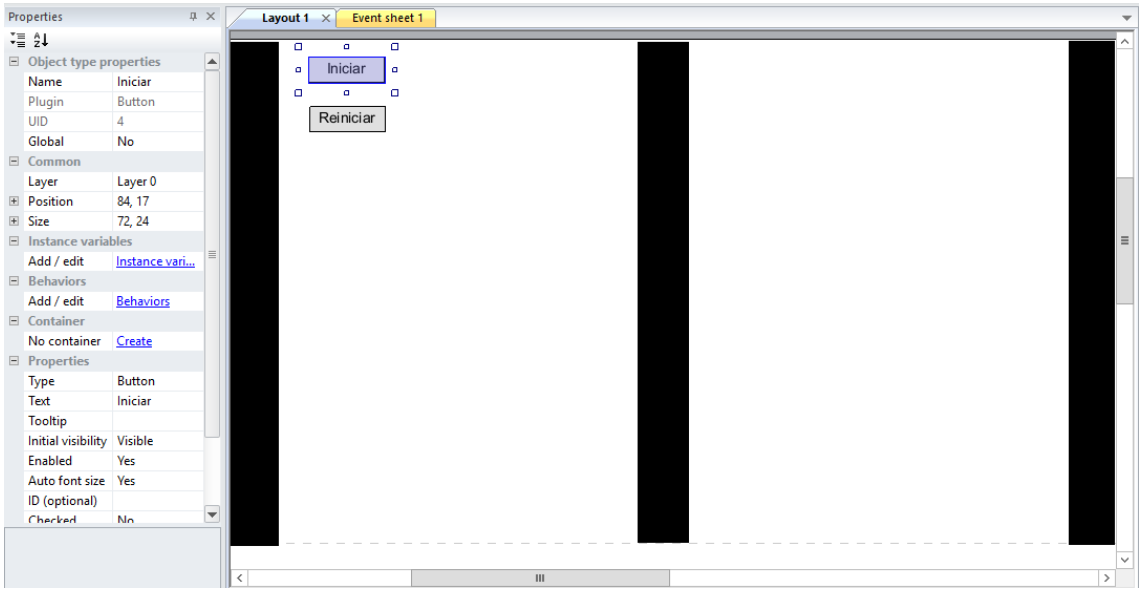


Figura 70 Posição dos objetos e botões
Fonte: Autor

Agora é o momento de inserir as legendas e textos que apareceram durante o desafio. Para acrescentá-los deve seguir o mesmo procedimento utilizado para os outros objetos, dessa vez selecionando a opção *text,* conforme visto na figura 71.

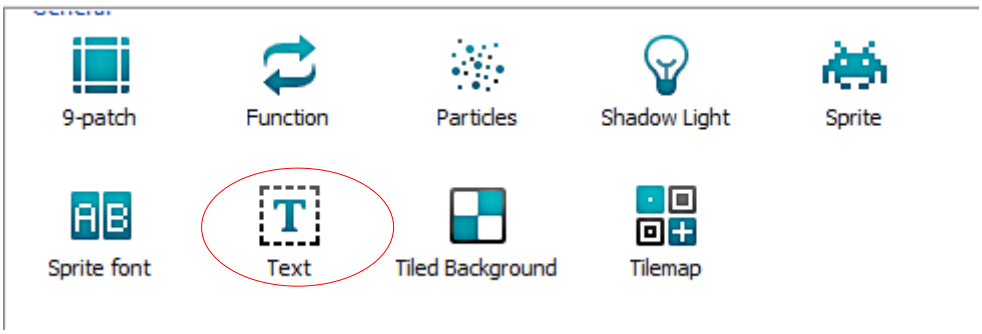


Figura 71 seleção do objeto text
Fonte: Autoria Própria

Inserir sete objetos de texto nas posições mostradas na figura 72. Os nomes e textos modificados podem ser vistos na figura 73, seguindo a ordem marcada em vermelho na figura 72. O procedimento é o mesmo para mudança dos nomes dos objetos vistos anteriormente. A figura 73 também mostra algumas outras modificações feitas nas propriedades dos objetos de texto 2, 3, 5 e 7, seguindo a ordem da legenda da figura 72. Nesses objetos além dos textos foram modificados a condição de visibilidade inicial nos objetos de texto 2,3,5 e 7, o tamanho da fonte do texto 3, e a cor dos objetos 5 e 7. Para fazer essas mudanças basta clicar na propriedade a ser modificada e selecionar a opção desejada.

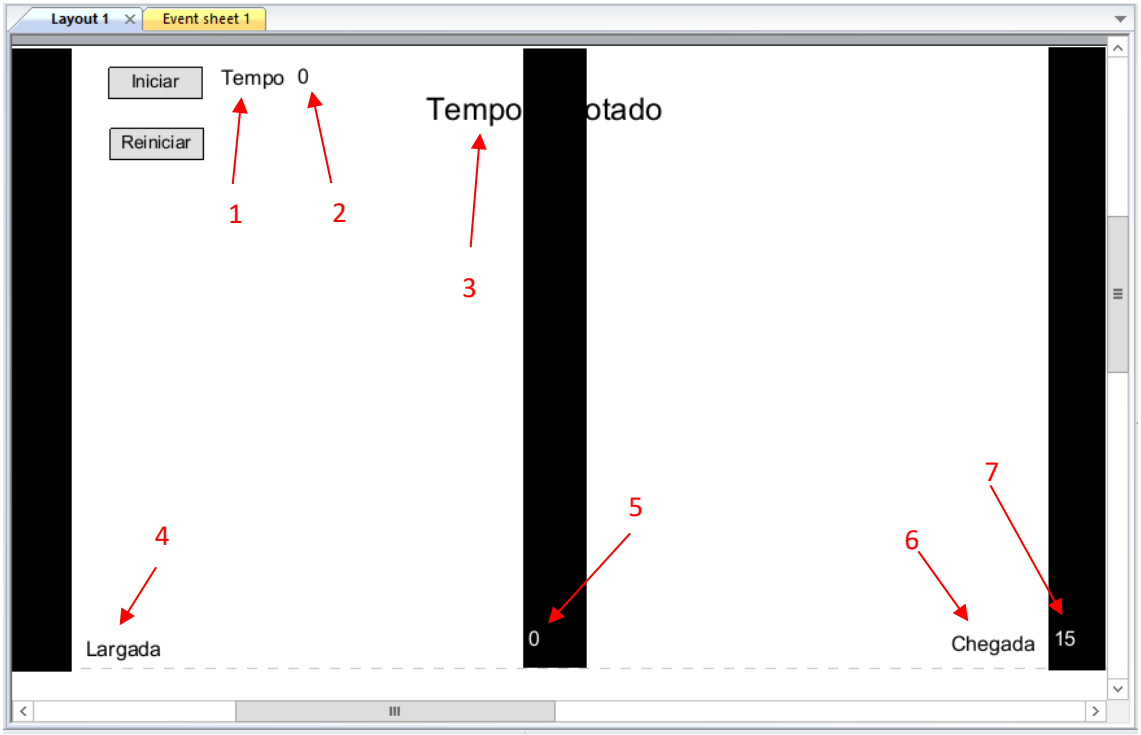


Figura 72 Disposição dos textos no layout da pista
Fonte: Autoria Própria

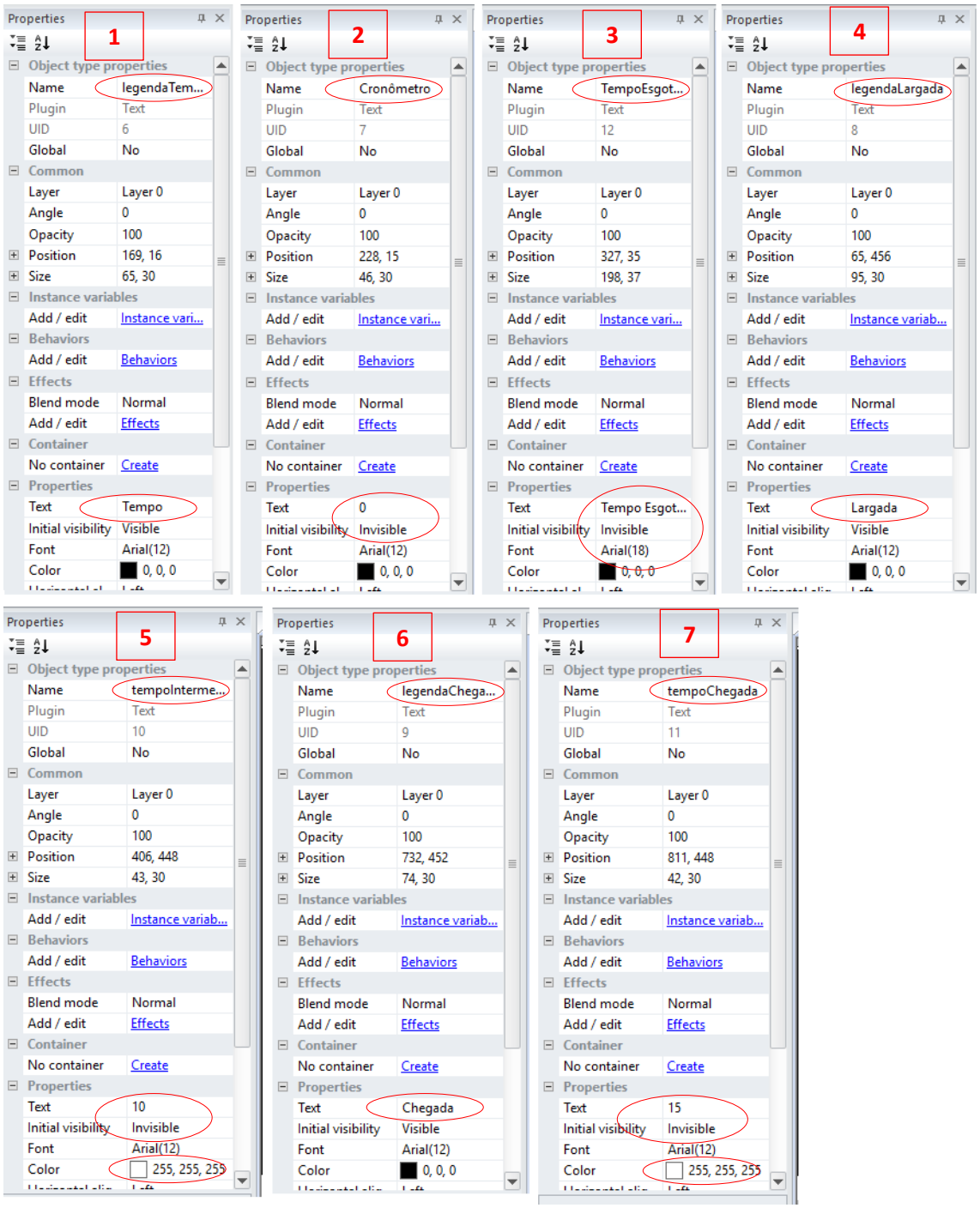


Figura 73 Modificações feitas nas propriedades dos objetos de texto
Fonte: Autoria Própria

Durante os desafios é necessário saber as dimensões da pista e dos objetos. O *software* trabalha com as medidas em pixels, portanto utilizar uma escala para conversão pode facilitar o trabalho. A escala pode ser criada a partir de um objeto "*Sprite*". Ao adicionar esse objeto e abrir o editor de imagem, clicar na opção *resize* e na janela que se abrir digitar os valores para as dimensões de altura de 40 pixels e comprimento de 200 pixels (figura 74).

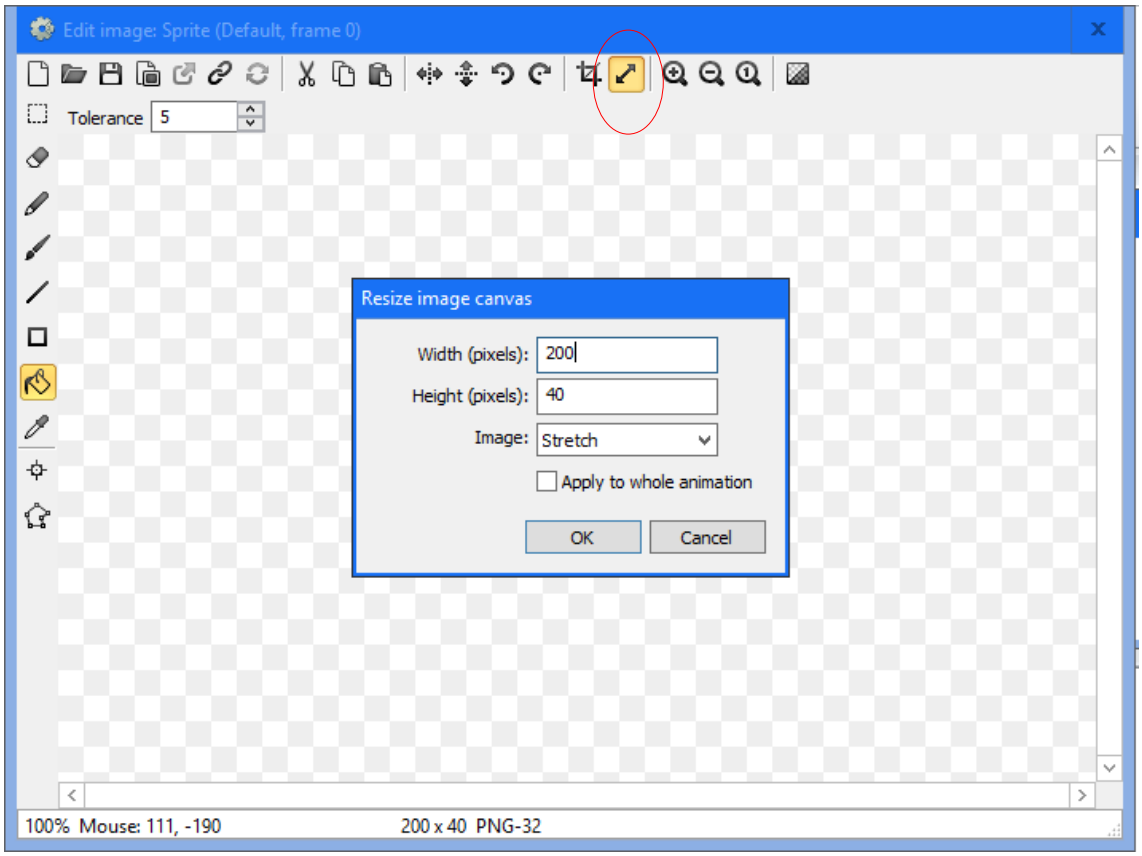


Figura 74 redimensionando um objeto
Fonte: Autoria Própria

Criar um padrão de retângulos nas cores preta e vermelha, com comprimento de 20 pixels para cada retângulo, conforme aparece na figura 75. Para facilitar o desenho com tamanhos iguais, pode ser utilizado as informações de posição do mouse (figura 47).

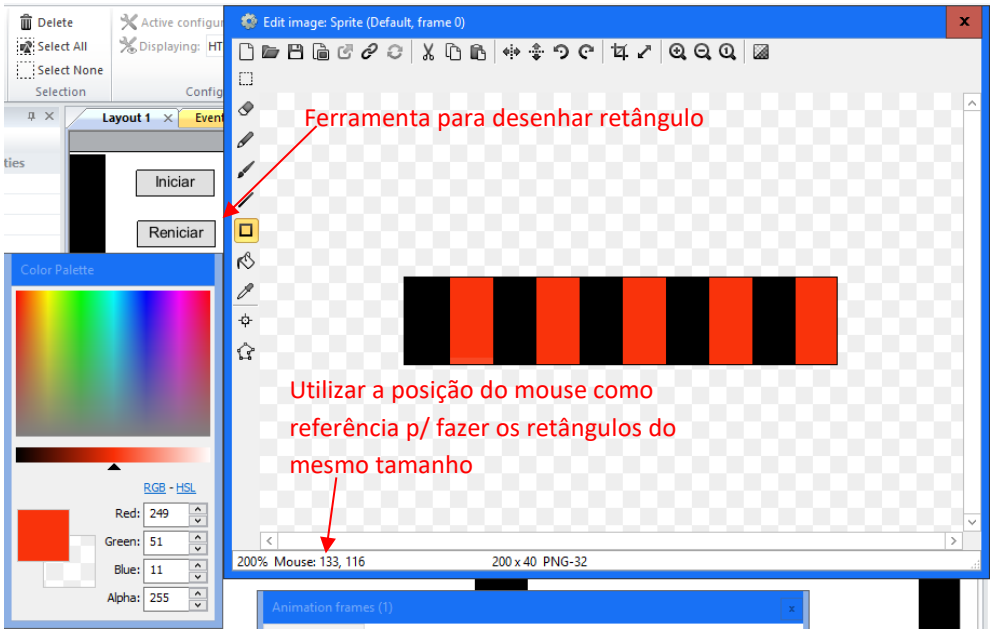


Figura 75 Editor de imagem

Fonte: Autoria Própria

Após inserir a escala colocar as legendas utilizando o objeto *text*. A pista deverá ficar parecida com a figura 76.

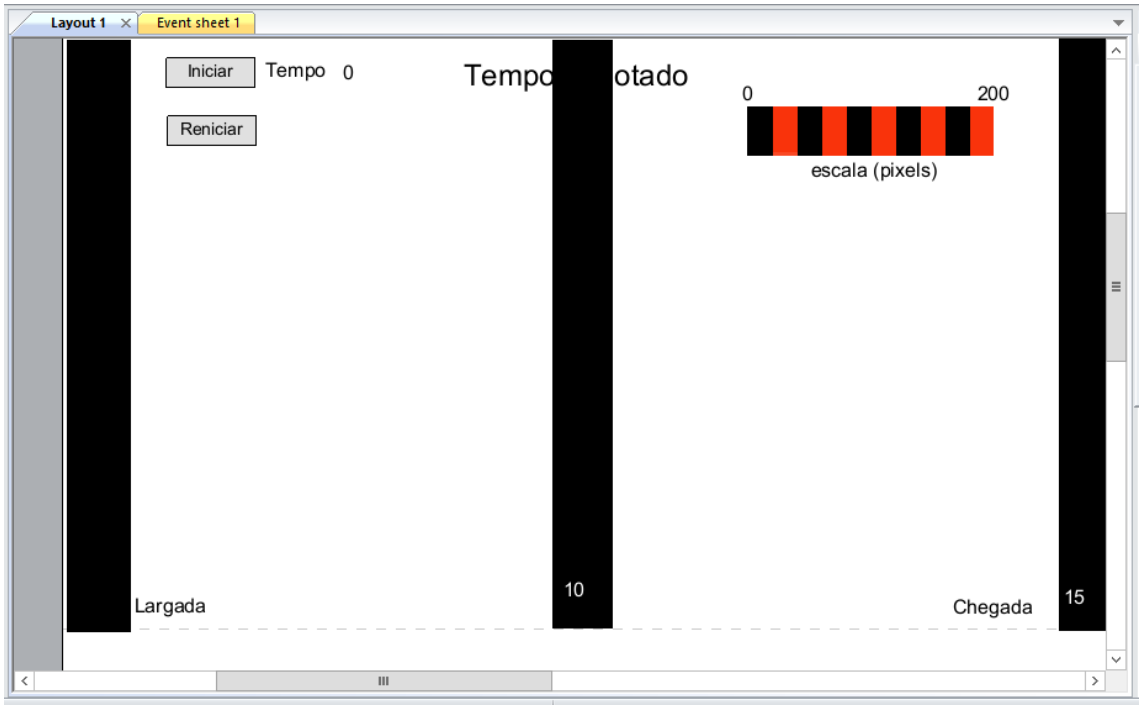


Figura 76 Aparência da pista com escala e legendas
Fonte: Autoria Própria

Para dar movimento a pista é necessário acrescentar comportamentos (*behaviors*) aos objetos que a compõe. Os comportamentos já estão, funcionalmente, pré-definidos no *Construct 2*. Na pista de exemplo pista será utilizado o comportamento (*behavior*) chamado *fade* que faz um objeto desaparecer gradativamente. A adição começa clicando no objeto para selecioná-lo. Na *properties bar*, ver a categoria *Behaviors*. Clicar em *Add / Edit*. A janela de comportamentos para o jogador, mostrada na figura 77, irá abrir. Nesse caso será mostrado o processo para o objeto “Intermediário”

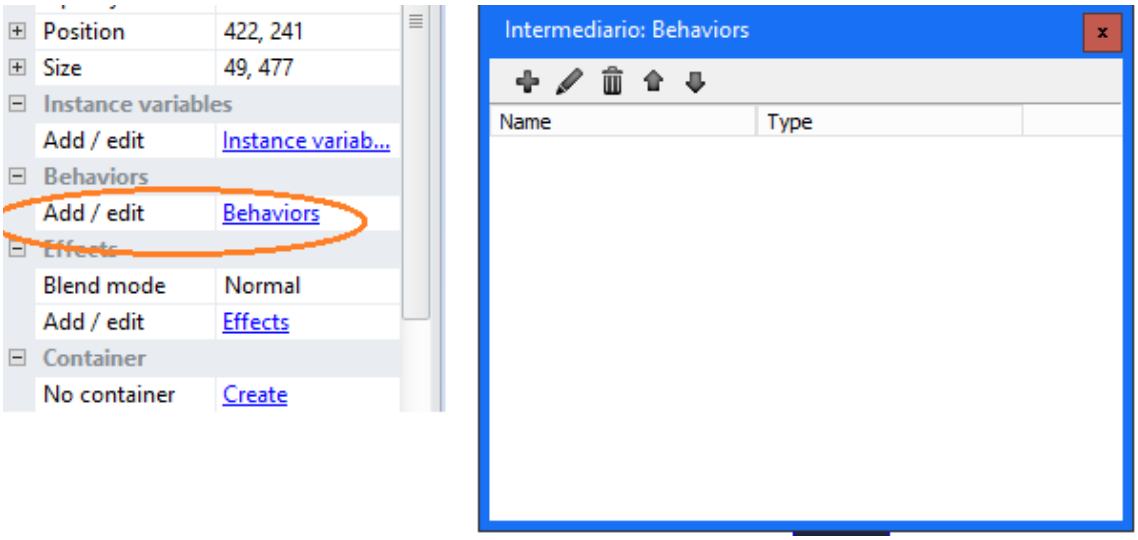


Figura 77 Adicionando um comportamento
Fonte: Autoria Própria

Clicar na cruz verde *'add behavior'*. Dar um clique duplo em *fade* (figura 78)

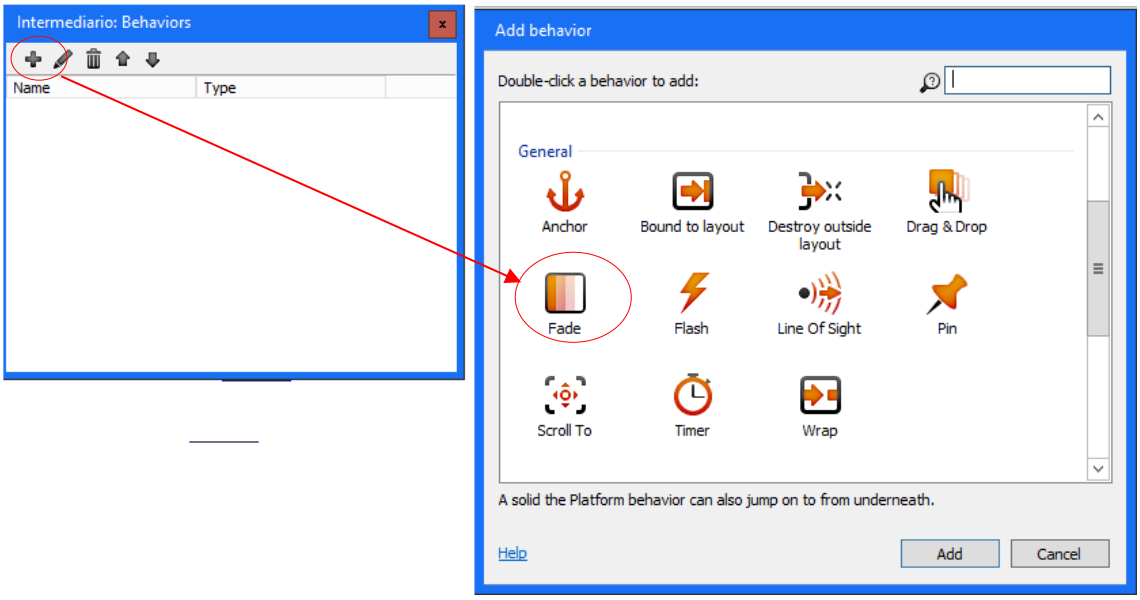


Figura 78 Adicionando o comportamento fade
Fonte :Autoria Própria

A janela de comportamentos ficará como a figura 79:

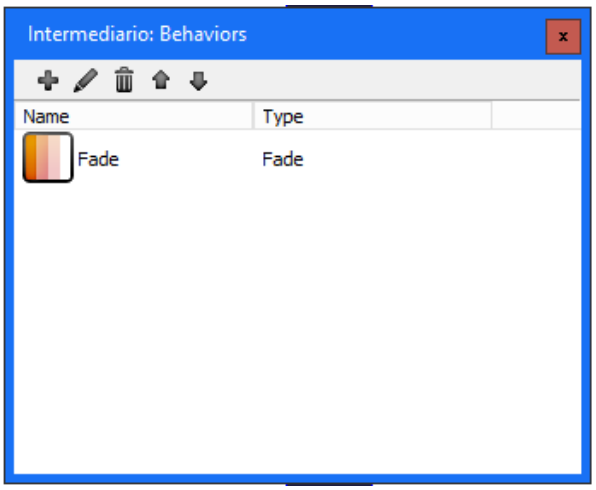


Figura 79 janela de comportamento após adicionar o fade
Fonte: Autoria Própria

Por definição o tempo para o *fade out,* responsável pelo desaparecimento do objeto, é ajustado para 1 segundo. Esse tempo deverá ser ajustado para 0,3 segundos, com isso o objeto desaparecerá mais rapidamente. Também deve ser mudado o parâmetro "*Active at start*" para "*No*", e o parâmetro "*Wait time*" para o valor 10. As propriedades ficarão como mostrado na figura 80.

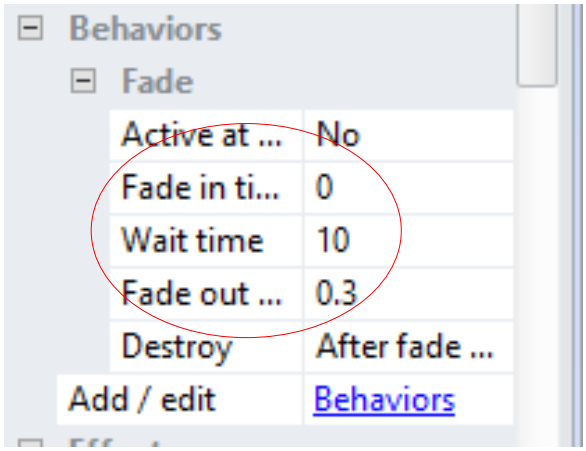


Figura 80 Ajuste de tempo do fade out
Fonte: Autor

O comportamento *fade* deverá ser aplicado, também, ao objeto "chegada". No objeto "Chegada" a propriedade "*wait time*" deverá ter o valor 15.

Após essa etapa o layout da pista está completo e deve ser feito a programação dos eventos relacionados a cada objeto.

8.1.1 Programando os eventos dos objetos da pista

Os eventos são inseridos na que se abre ao clicar em “*Event sheet*” conforme mostra a figura 81.

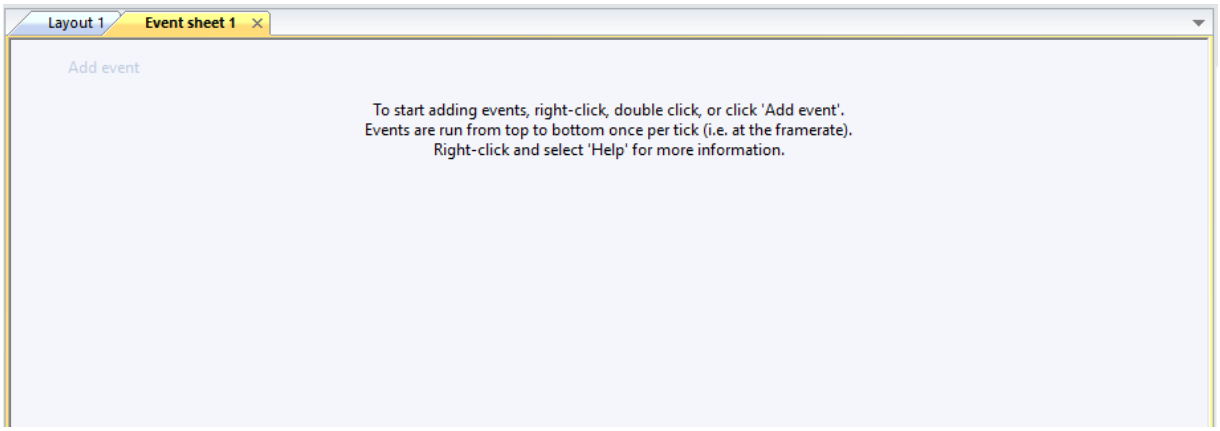


Figura 81 Aba Event sheet
Fonte: Autor

Para adicionar um evento clicar em “*Add event”*, então irá abrir uma janela para escolher o objeto no qual se deseja inserir o evento (figura 82). Escolher o botão iniciar dando um clique duplo sobre ele. Irá abrir a janela para escolha da condição a qual o evento irá ocorrer, no exemplo apresentado o evento deverá ocorrer ao clicar sobre o botão, portanto deve ser escolhida a opção “*On clicked”* clicando uma vez sobre ela (figura 82).

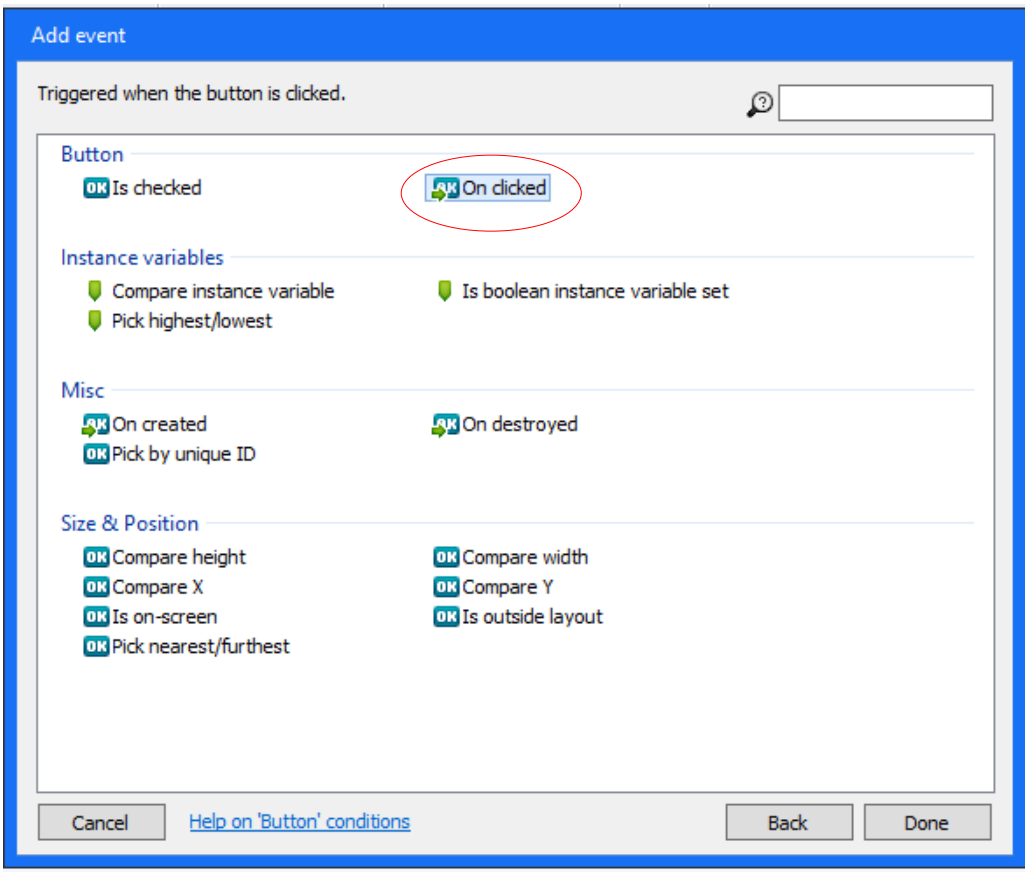


Figura 82 condição on clicked selecionada
Fonte: Autoria Própria

Após selecionado a condição clicar em "*Done"* conforme mostra a figura 82. A janela de ações ficará como mostrada na figura 83. A próxima etapa será adicionar a ação a ser executada quando o botão iniciar for clicado, para isso deverá ser dado um duplo clique na opção "*Add action*" também indicado na figura 83 pela seta vermelha.

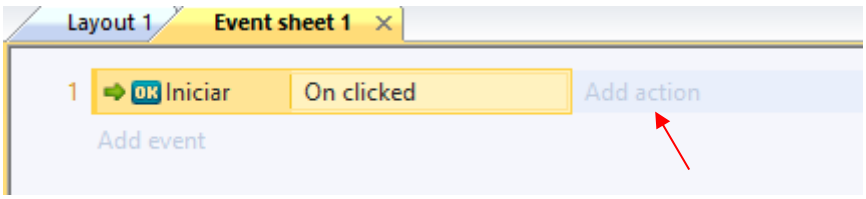


Figura 83 detalhe da janela de eventos
Fonte: Autoria Própria

Abrirá uma janela para a escolha do objeto no qual a ação será executada (figura 84).

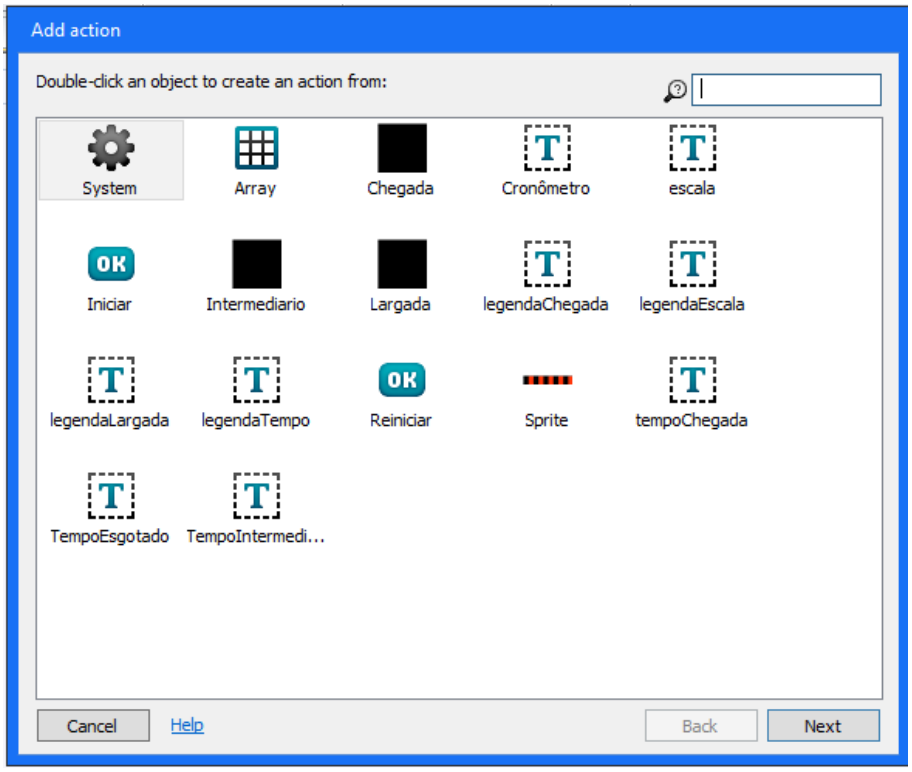


Figura 84 janela Add action
Fonte: Autoria Própria

Primeiro será acrescentado uma ação no objeto largada. A linha de largada é onde o robô será posicionado para iniciar o desafio, ao clicar no botão iniciar essa faixa deve sumir, pois o robô só irá andar quando o sensor fizer a leitura da cor branca no sensor. Ao dar um duplo clique no objeto largada irá abrir uma janela para escolha da ação, nessa tela escolher a opção "*set visible*" e na janela "*Parameters for Largada: set visible*", que irá abrir, mudar para "*invisible*" conforme mostram a figura 85. Feito isso clicar em "*done*" e a primeira ação será adicionada. Os passos seguintes serão adicionar as ações nos objetos intermediário e largada. A sequência de passos é igual a sequência executada para o objeto largada, porém a ação escolhida será "*start fade*" conforme mostra a figura 86. Nessa ação não abrirá a janela de parâmetros, pois a única tarefa será iniciar o tempo para o objeto desaparecer. Esse processo deve ser executado para os objetos "intermediário" e "chegada". Até esse momento a tela "*Event sheet*" deve estar igual à mostrada na figura 87. Ainda serão acrescentados outros eventos e ações a este botão, porém antes algumas variáveis devem ser acrescentadas na programação.

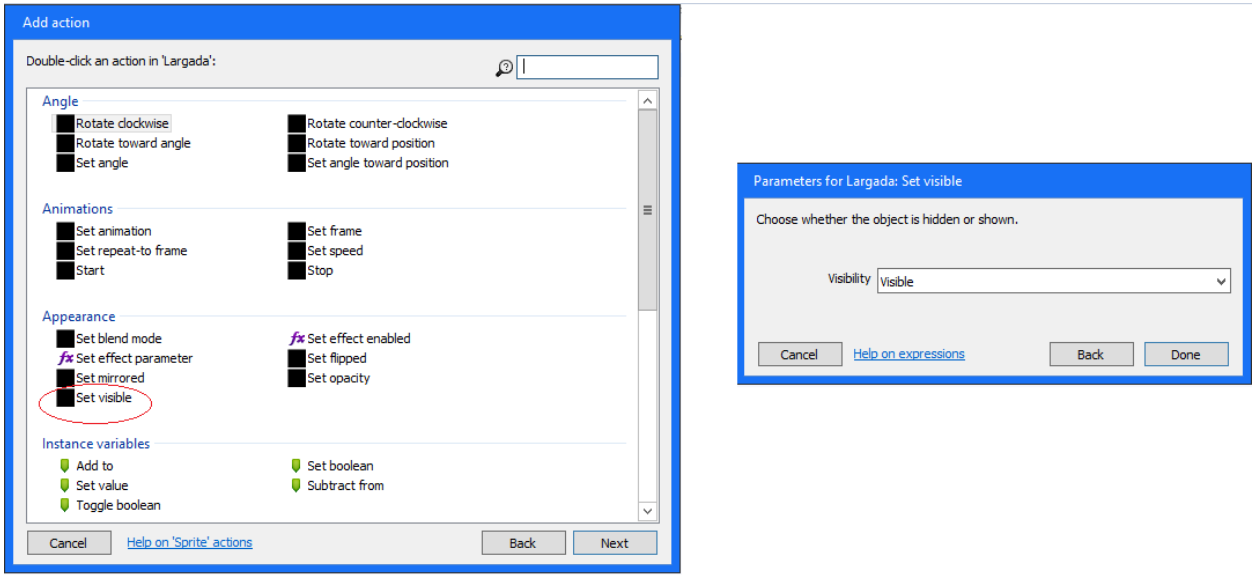


Figura 85 Escolha e parâmetro da ação
Fonte: Autor

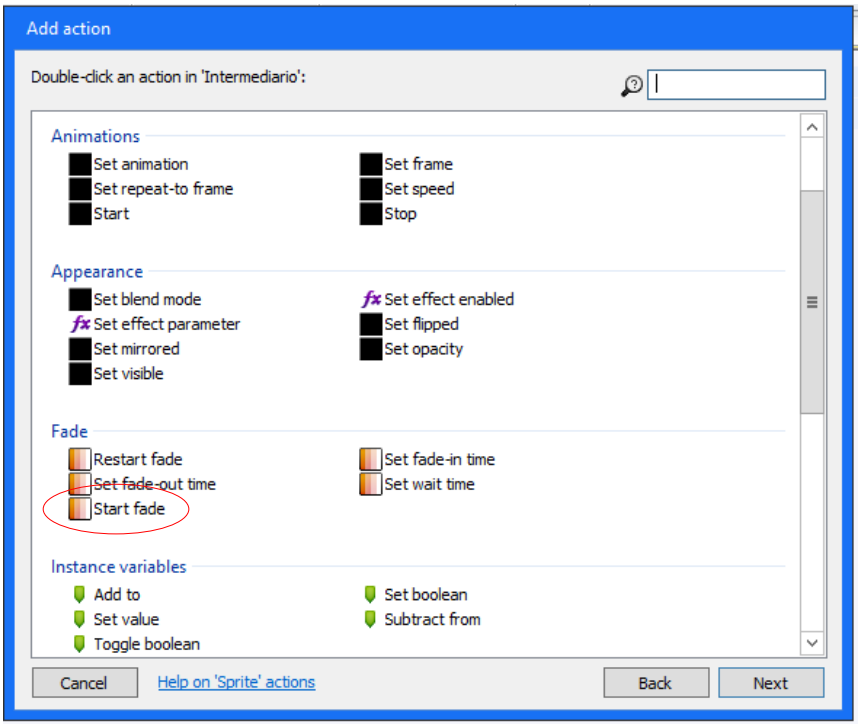


Figura 86 Ação escolhida para o objeto " intermediário"
Fonte: Autoria Própria

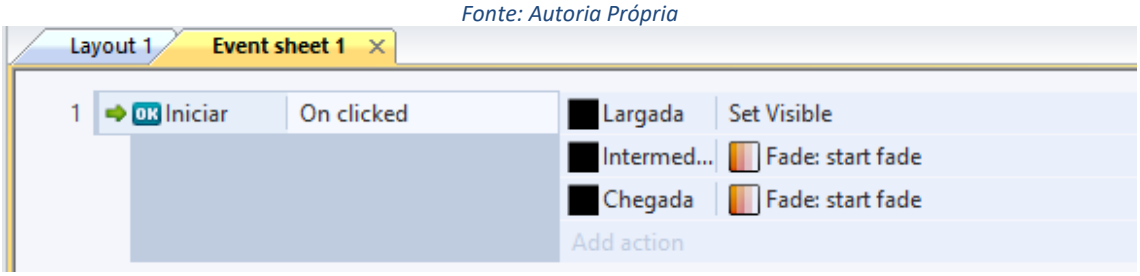


Figura 87 Aparência da tela "Event sheet" até agora
Fonte: Autoria Própria

Para o desafio que está sendo programado é necessário cumprir as tarefas em um determinado tempo, portanto é necessária uma contagem de tempo e a

sincronização dos eventos ao iniciar o desafio. Dessa forma será acrescentado uma variável de tempo e um "*cronômetro*" na programação.

Para acrescentar uma variável basta clicar a tecla "v" e irá abrir uma janela para configuração da nova variável. O nome da primeira variável, nesse caso, pode ser "*tempoTotal*", o tipo pode continuar como número *(number)* e o valor inicial zero. Essa variável será responsável pela contagem do tempo total do desafio.

Os objetos intermediários e chegada necessitam de um timer para que o aluno possa ver o tempo que falta para o objeto desaparecer, então deve ser acrescentado uma variável de tempo para cada um desses objetos. O processo é o mesmo da primeira variável, porém mudando os nomes e valor inicial das variáveis. A figura 88 mostra como devem ficar as configurações das variáveis "*tempoTotal*", tempo do objeto "*intermediário*" e do objeto "*chegada*" respectivamente.

Por essas variáveis é possível ver que o desafio deverá ser cumprido em 15 segundos, pois após esse tempo a linha de chegada irá desaparecer, e que o objeto intermediário desaparecerá dez segundos após o início do desafio.

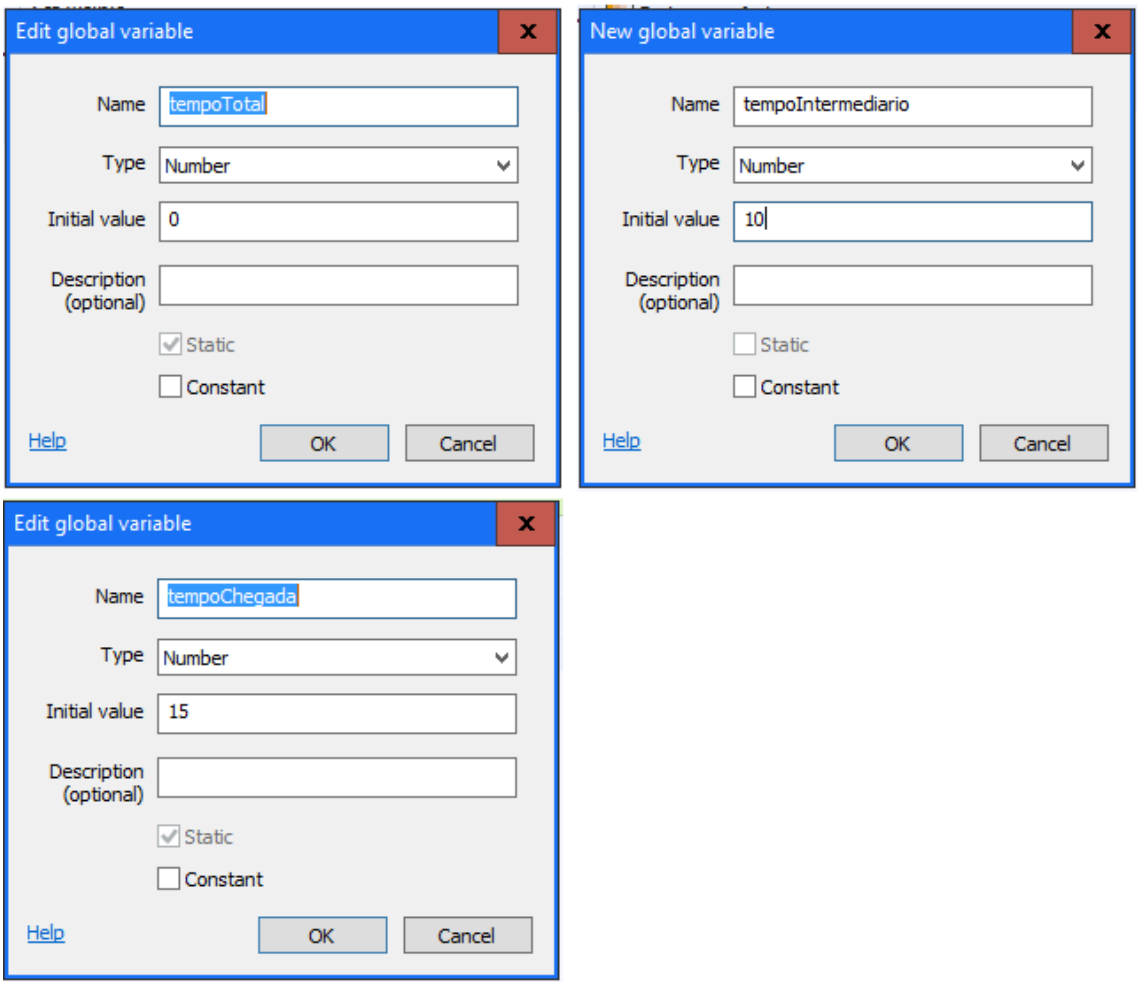


Figura 88 variáveis tempoTotal, tempoIntermediário e tempoChegada
Fonte: Autoria Própria

Acrescentado as variáveis necessárias, mais algumas ações serão acrescentadas ao botão iniciar. Seguir os passos vistos anteriormente para acrescentar eventos e ações e, no momento de escolha do objeto no qual a ação será executada, clicar em “*system”* (figura 89).

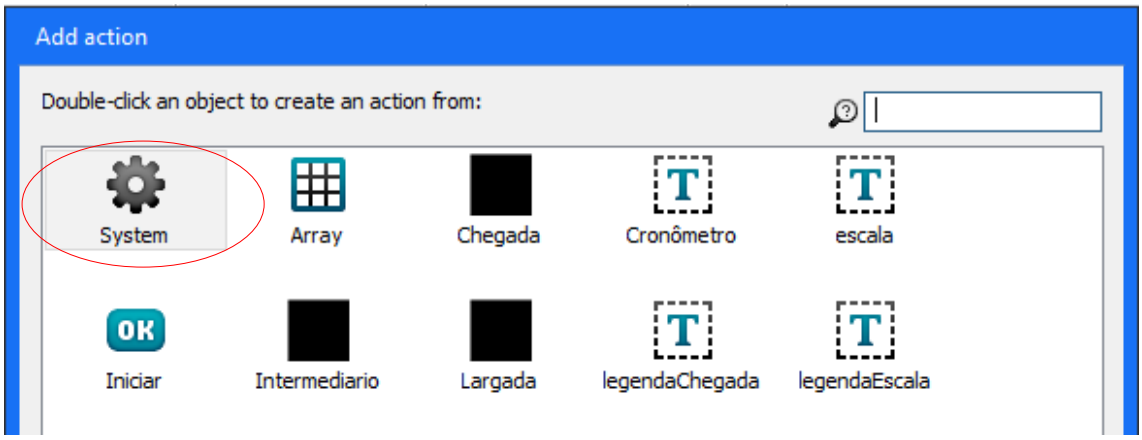


Figura 89 Escolha do objeto "system" para criar uma ação
Fonte: Autor

Na janela de escolha da ação, mostrada na figura 90, clicar em “*reset global variables*”, isso fará o sincronismo da contagem de tempo com o início do desafio.

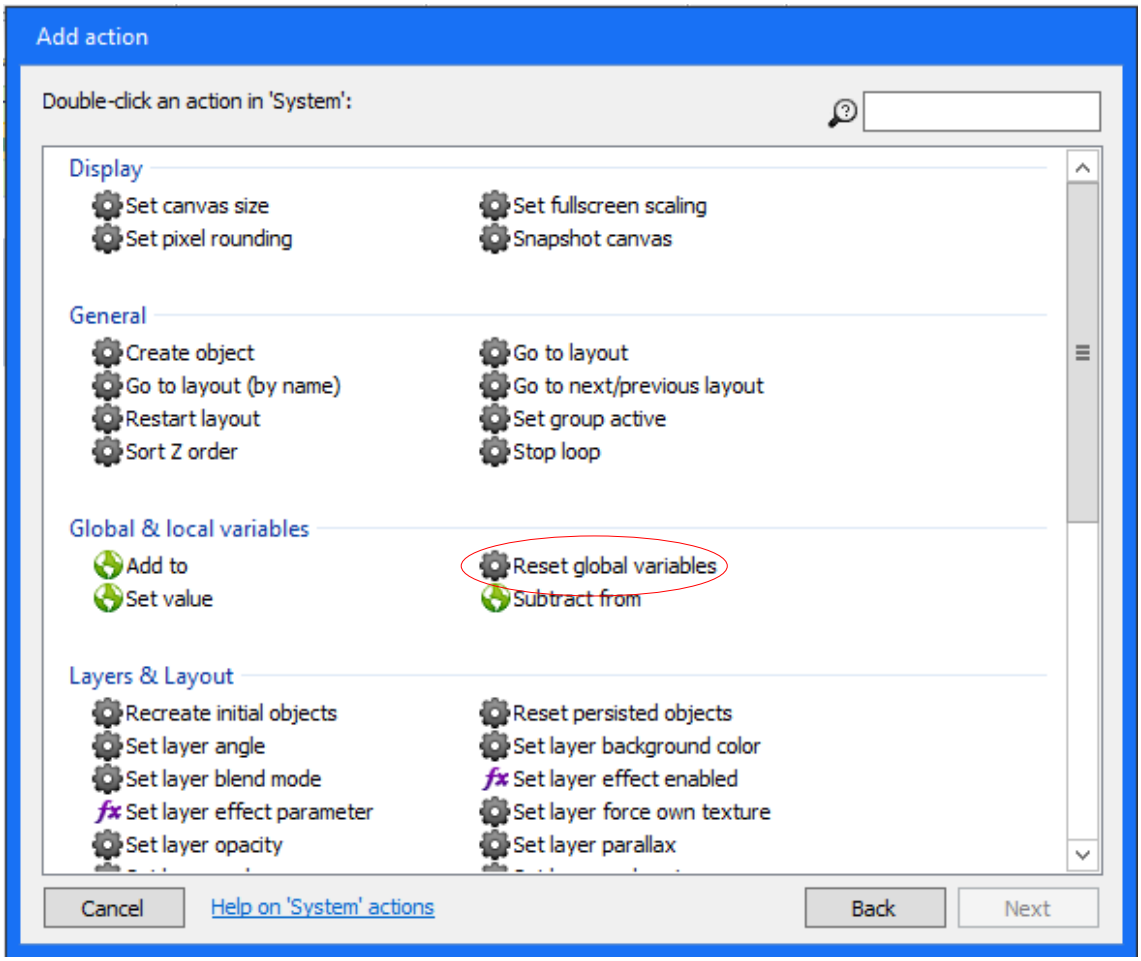


Figura 90 ação de reset nos tempos do desafio
Fonte: Autor

Para que os valores de tempo sejam mostrados na tela quando o desafio iniciar é necessário acrescentar uma ação aos objetos de texto referentes ao tempo de cada objeto. Para isso, dar um duplo clique no objeto “cronômetro” e na janela para escolha da ação escolher a opção “*set visible*” e na janela “*Parameters for Cronômetro: set visible*”, manter a opção “*visible*” conforme mostrado na figura 91.

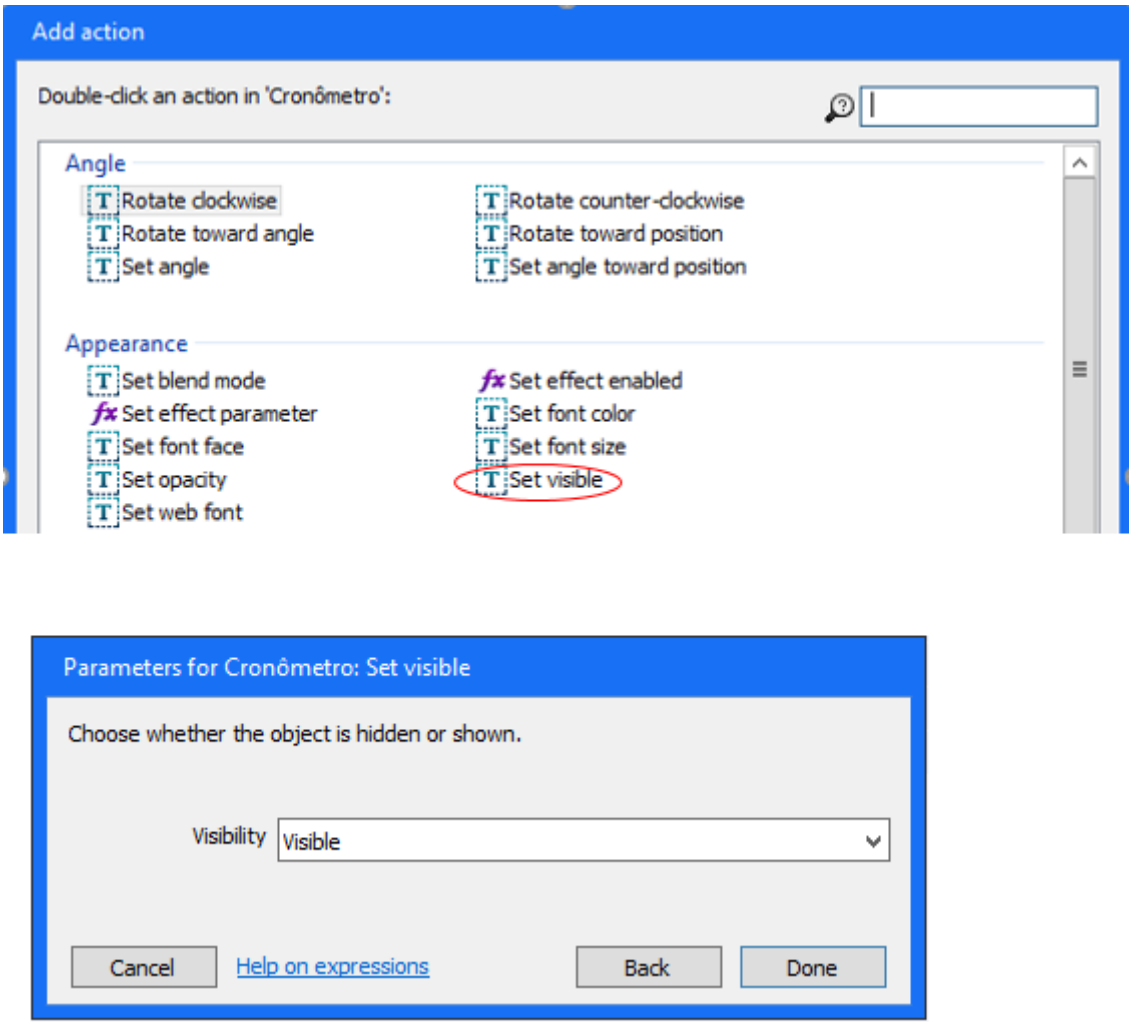


Figura 91 configuração da visualização do tempo total
Fonte: Autor

O cronômetro só irá contar o tempo quando um evento e uma ação forem configurados. Selecionando "*add event*" no "*event sheet"* escolhemos o objeto *"system"* e o evento "*every X seconds*" (figura 92). Na janela de parâmetros manter o intervalo de 1.0 segundos e clicar em "*done*" (figura 93). Feito isso deve ser adicionado uma ação para esse evento. Em "*add action*" escolher *System* novamente e a ação *"Add to" (figura 94)* e na janela de parâmetros escolher a variável *tempoTotal* e manter o valor em 1 (figura 95). Com esse processo o cronômetro irá fazer a contagem de tempo. Os tempos do objeto "Intermediario" e "Chegada" devem ser regressivos, portanto, deve ser adicionado a ação "*subtract from*" para as variáveis *tempoIntermediario* e *tempoChegada,* (figura 96*)* o valor dessa ação permanece 1 segundo.

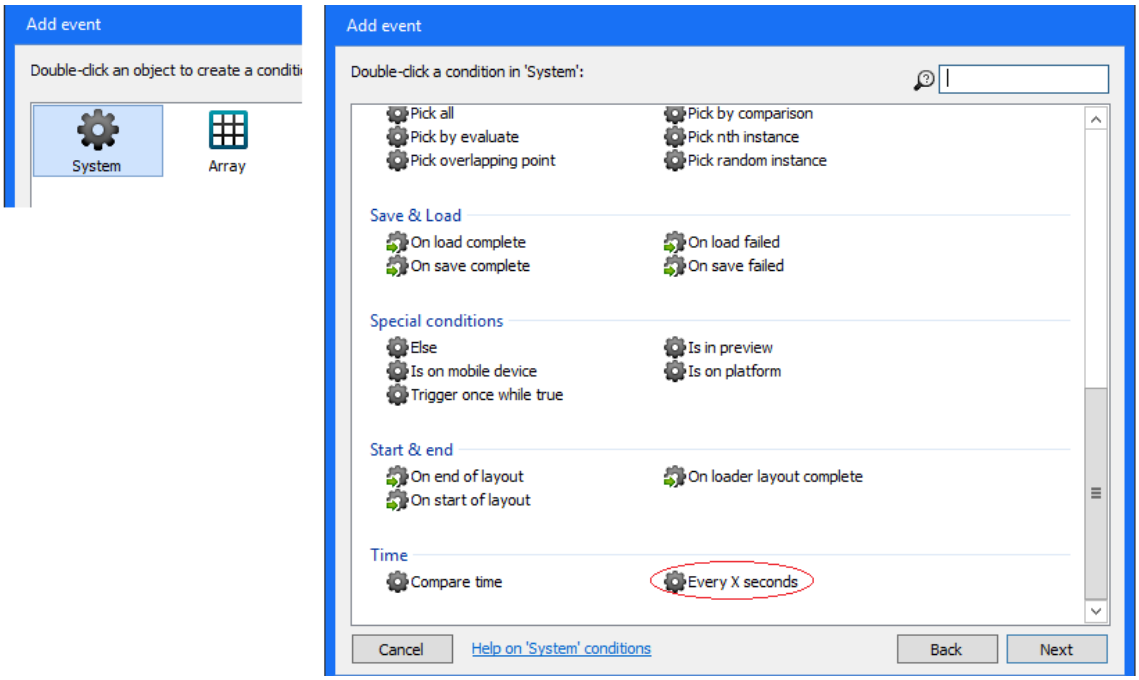


Figura 92 Escolha do evento "every X seconds"
Fonte: Autoria Própria

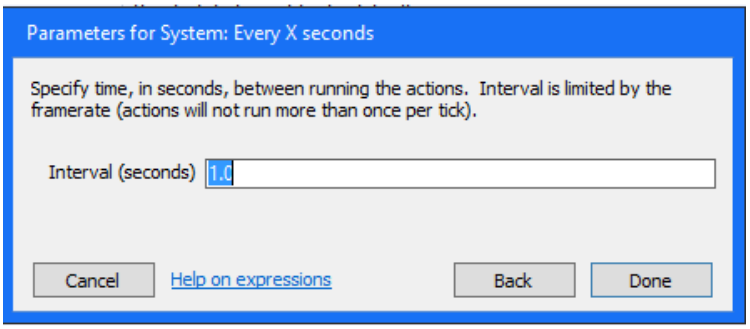


Figura 93 Configuração para contagem de 1 segundo
Fonte: Autoria Própria

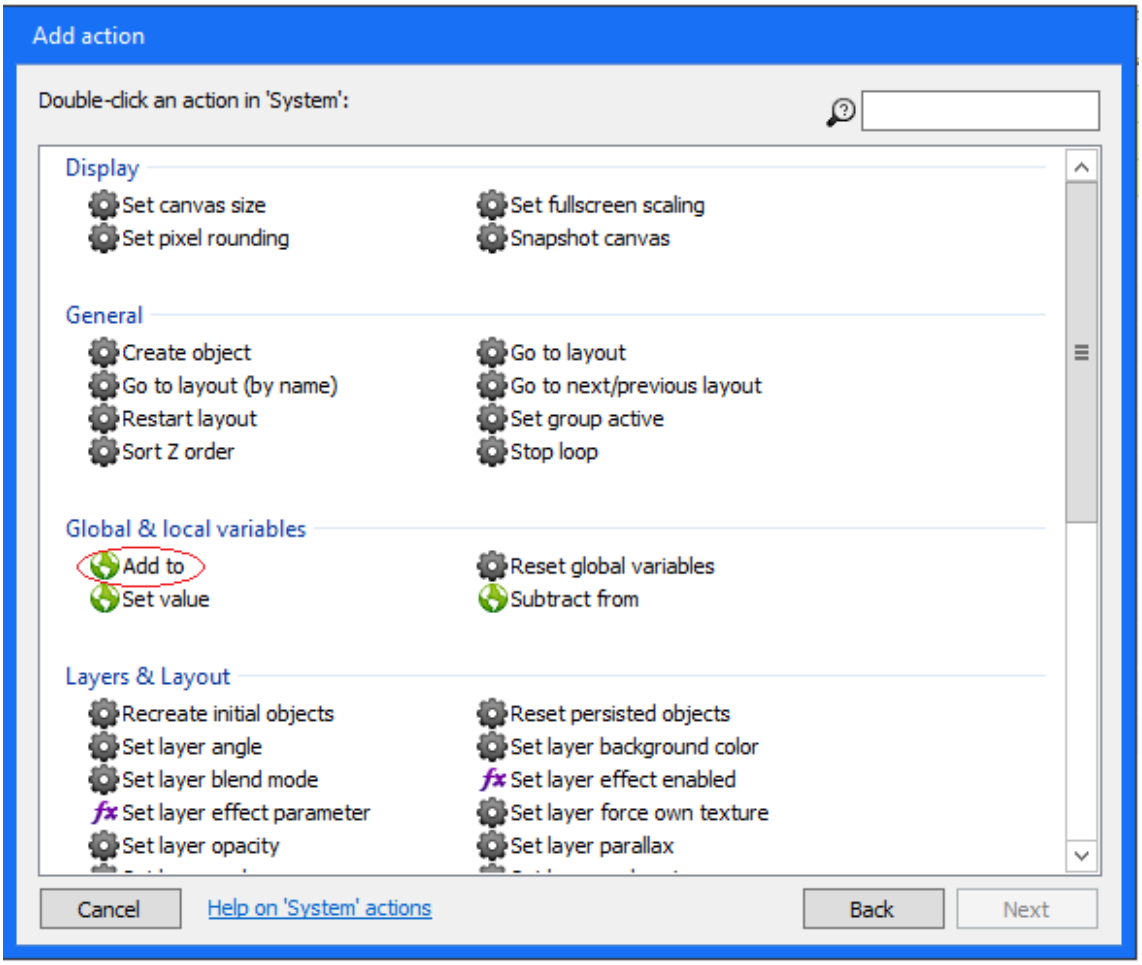


Figura 94 Escolha da ação Add to aplicado ao cronômetro.
Fonte: Autoria Própria

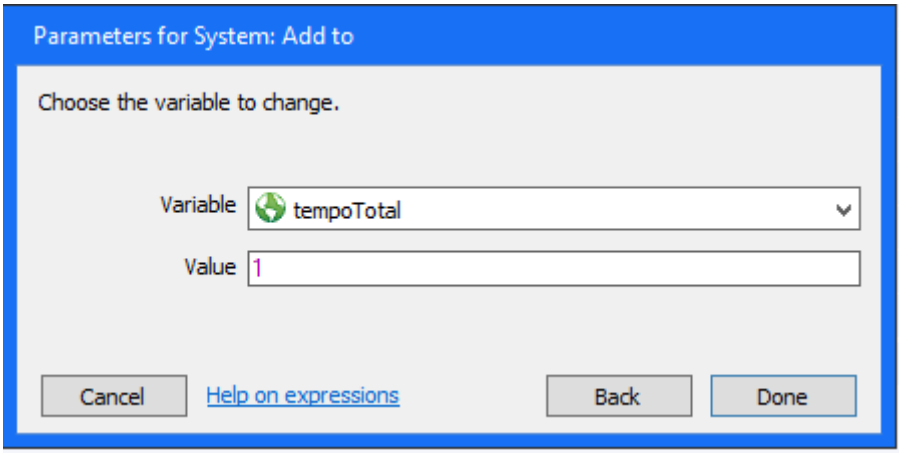


Figura 95 Parâmetros da ação Add to para o cronômetro
Fonte: Autoria Própria

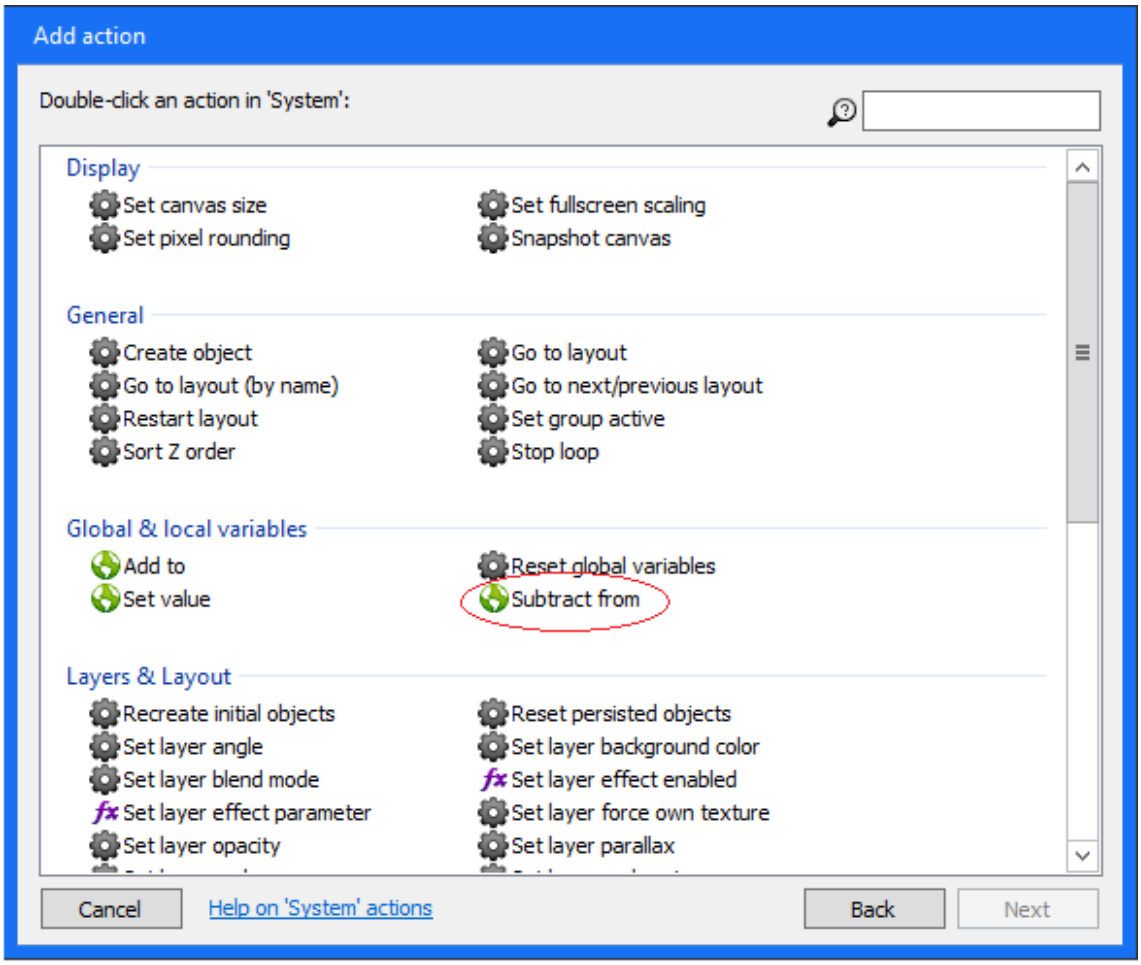


Figura 96 Escolha da ação subtract from para as variáveis tempoIntermediário e tempoChegada
Fonte: Autoria Própria

As próximas ações adicionadas ao evento *"every X seconds"* irá fazer com que os tempos sejam exibidos corretamente nos objetos de texto do cronômetro, *tempoIntermediário* e *tempoChegada*. Começando pelo cronometro, seguir os seguintes passos: vincular o valor da variável "tempoTotal" com o cronômetro, escolhendo a ação *"set text"* para esse objeto e na janela *"Parameters for Cronômetro*" digitar o nome da variável "*tempoTotal*", com isso irá abrir outra janela onde deverá ser escolhida a variável (figuras 97 e 98). Esses passos deverão ser repetidos para os objetos "*tempoIntermediário*" e "*tempoChegada*". Não esquecer de vincular as variáveis corretas para cada objeto.

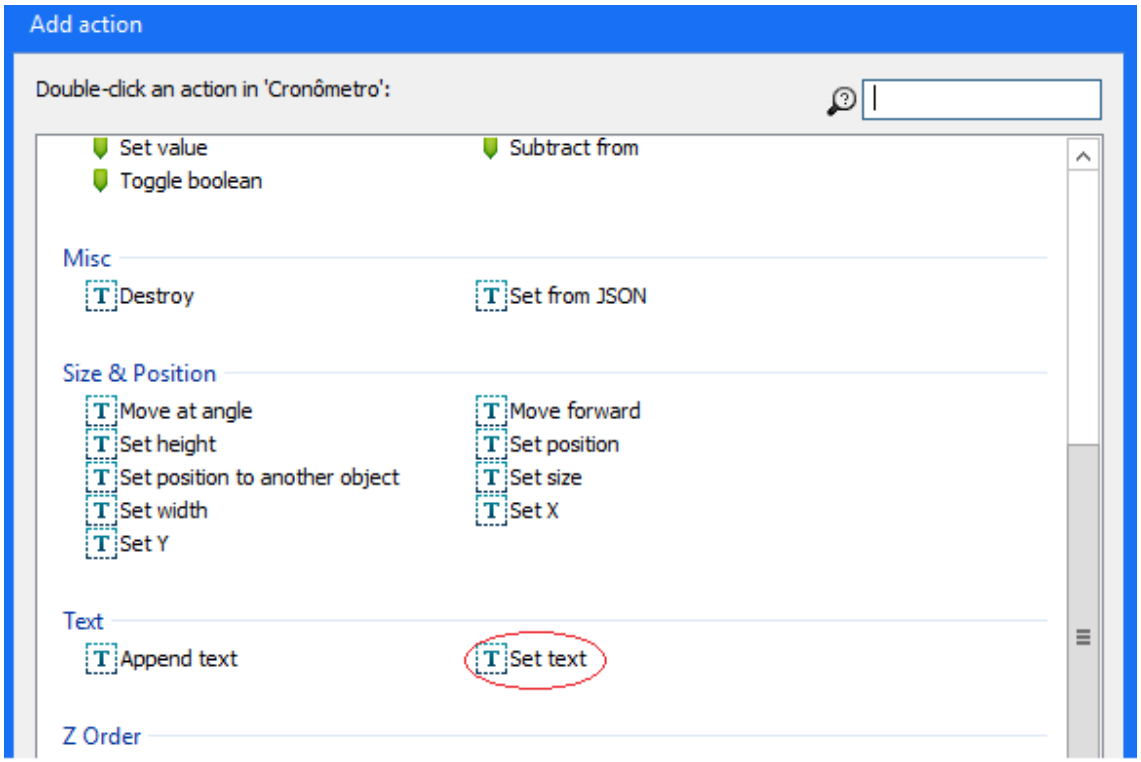


Figura 97 Escolha da ação set text
Fonte: autoria Própria

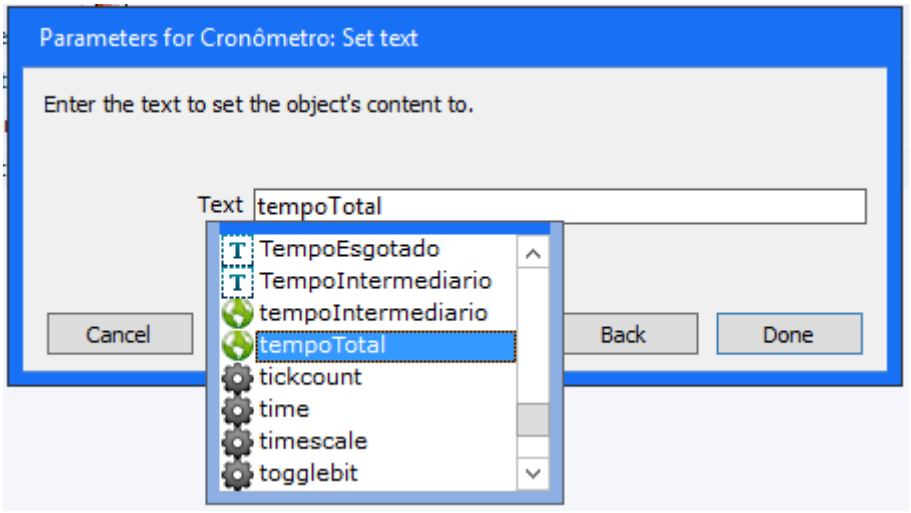


Figura 98 parâmetros de texto para o cronômetro
Fonte: Autoria Própria

Quando o objeto Chegada desaparecer, todos os elementos da pista desaparecerão e surgirá na tela a mensagem “Tempo Esgotado”. Será necessário acrescentar o evento “*On fade-out finished*” ao objeto Chegada. Nas ações para esse evento devem ser seguidos os passos da figura 91 para cada objeto de texto, ajustando o parâmetro para “*invisible*” para as legendas e tempos e “*visible*” para o objeto “tempoEsgotado”. A última etapa é adicionar um evento e ação para o botão reiniciar, seguindo o passo mostrado na figura 83, para esse botão, e adicionar a ação

"*Restart layout*" para o objeto System (figura 99). Essa ação irá fazer com que o desafio volte a condição inicial e possa ser executado novamente.

A tela "*Event sheet*" deve estar como mostrado na figura 100, com todos os eventos adicionados para esse primeiro desafio.

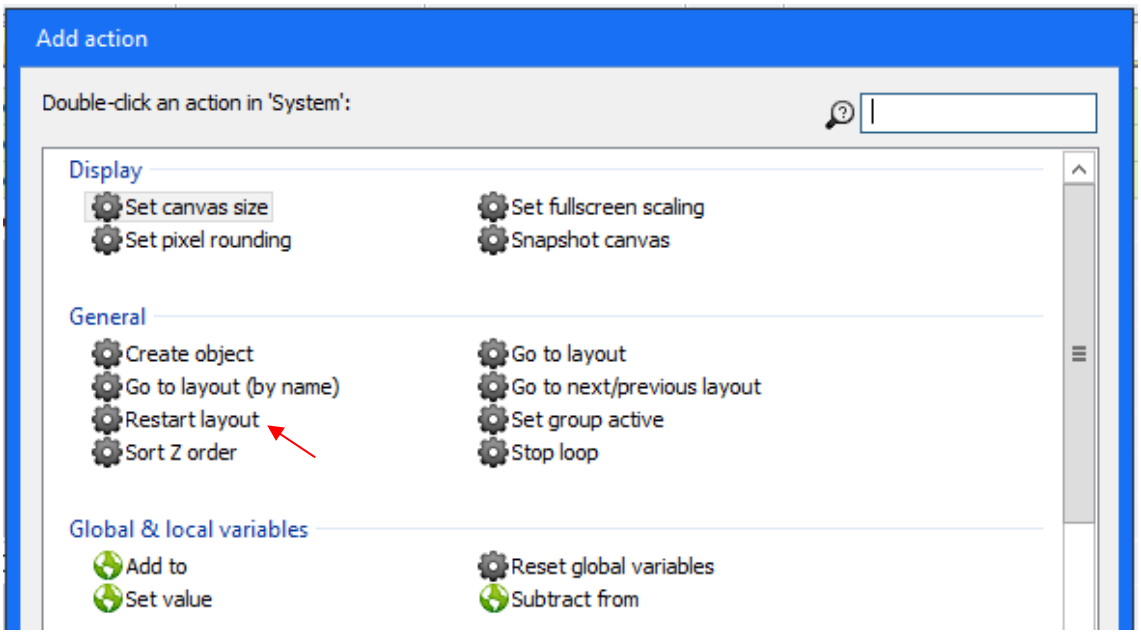


Figura 99 Ação para reiniciar o desafio.
Fonte :Autoria Própria

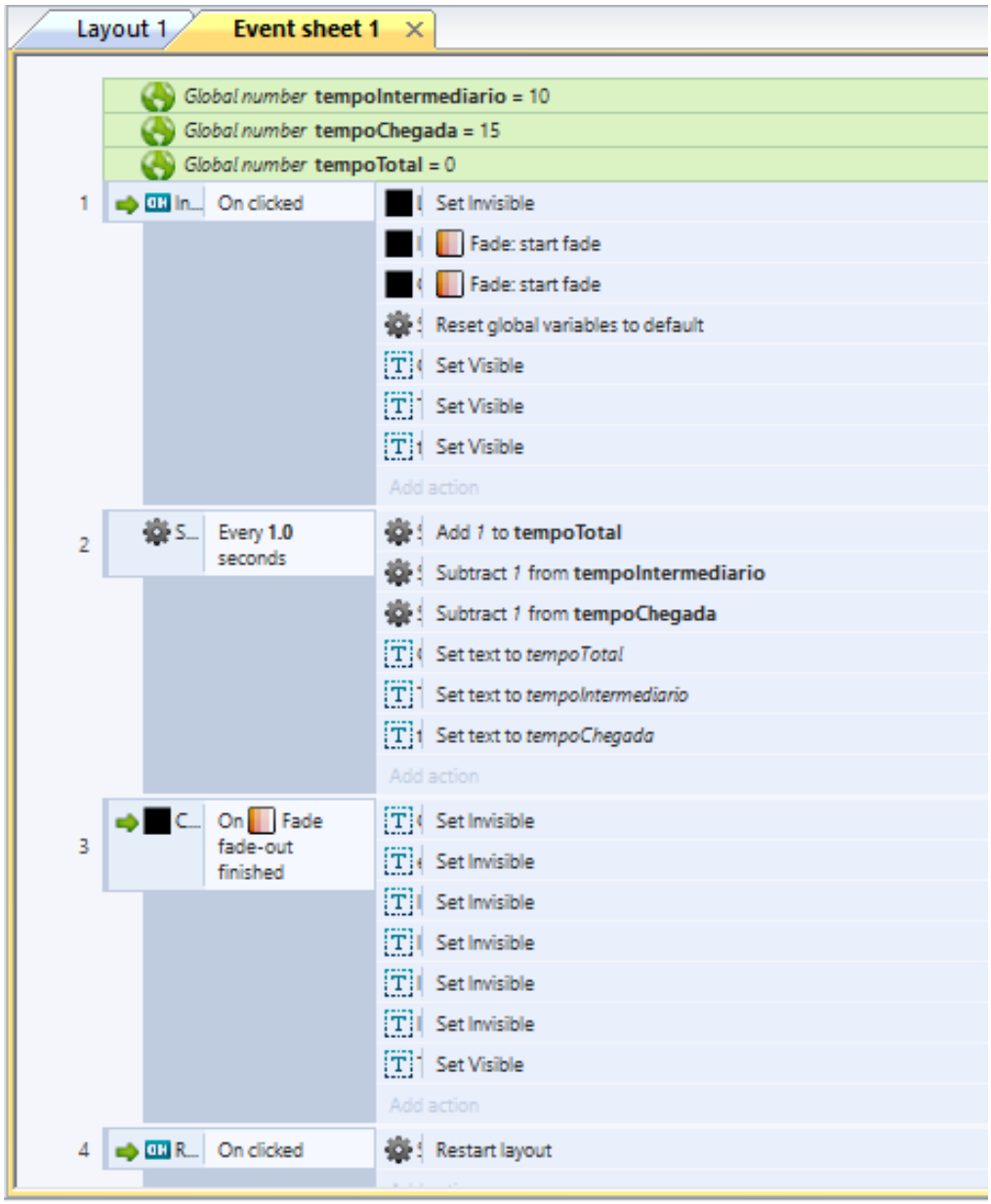


Figura 100 Aparência final da tela Event sheet
Fonte: Autoria Própria

A programação para o desafio Velocidade Média já está concluída, para executá-la basta clicar no botão *"Run layout"* localizado na barra de tarefas do software Construct 2, conforme mostra a figura 101. Uma janela do navegador irá abrir com a pista e os controles, nessa janela, poderá ser feito nos botões configurados durante esse projeto.

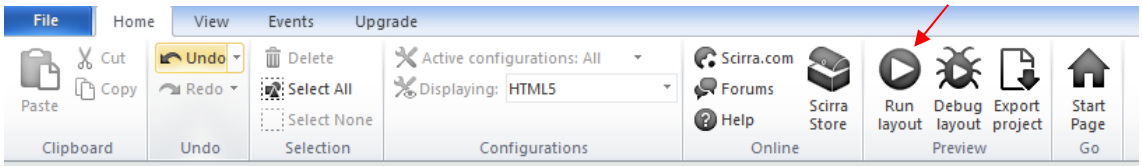


Figura 101 Botão para executar o projeto
Fonte: Autoria Própria

Para utilizar a pista basta projetá-la no piso com o projetor fixado a uma altura de aproximadamente 2 metros do chão para que a pista tenha um tamanho adequado. A figura 102 mostra uma sugestão de como fixar o projetor utilizando uma peça conhecida como sargento. Essa peça, mostrada na figura 103, pode ser encontrada em sofás antigos ou comprados em lojas de ferragem. Será necessário fazer um furo na peça para passar o parafuso de fixação do projetor. O sargento será parafusado no projetor no local utilizado para fixá-lo no teto.

Figura 102 fixação do projetor.
Fonte: Autoria Própria

Figura 103 "Sargento" utilizado para fixar o projetor.
Fonte: Autoria Própria

A figura 104 mostra uma sugestão de montagem do suporte para o projetor.

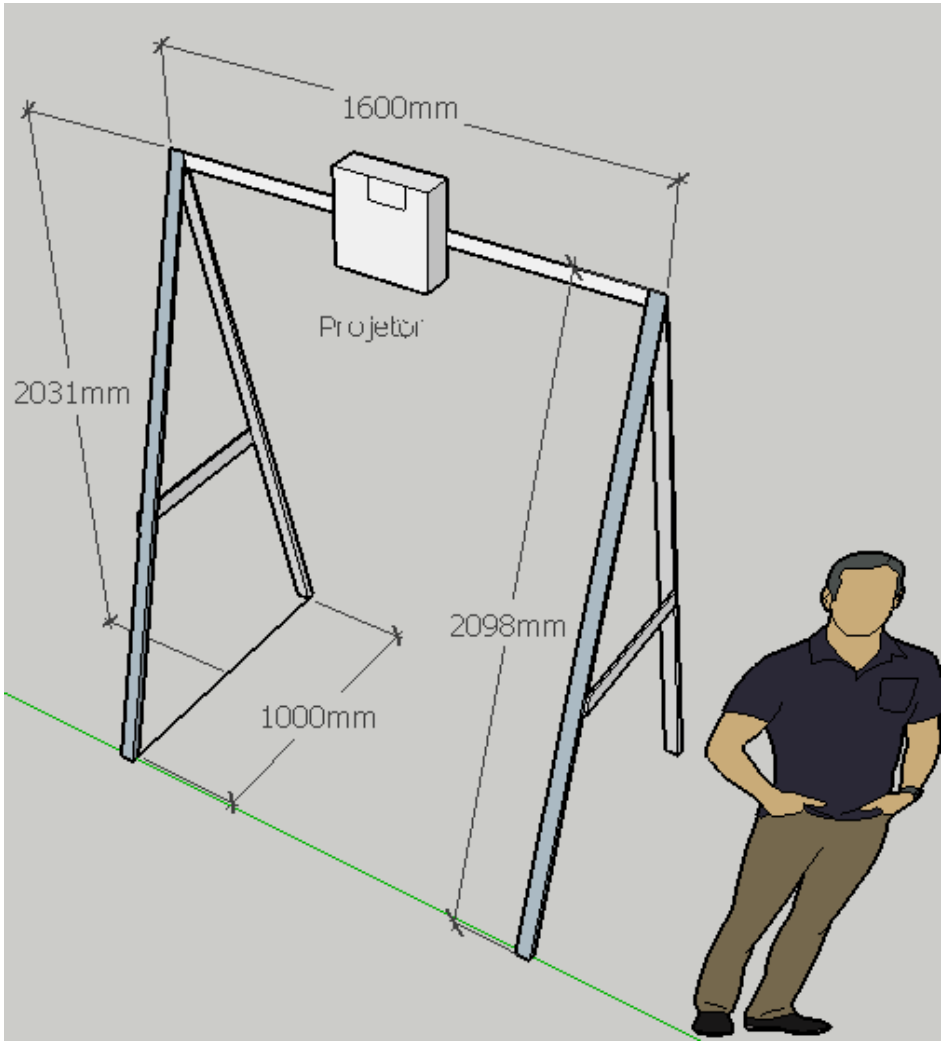


Figura 104 Suporte para o projetor
Fonte: Autoria Própria

Desafio 1 – Velocidade Média

8.1.2 Objetivo do desafio

Fazer o robô chegar até o final da pista, cumprindo os requisitos, no tempo estipulado.

8.1.3 Objetivo relacionado à Física

Conhecer o conceito de velocidade média e como efetuar esse cálculo;

Aprender a realizar conversões de medidas.

8.1.4 Descrição

Este desafio é composto por uma pista conforme mostrado nas figuras 105 e 106.

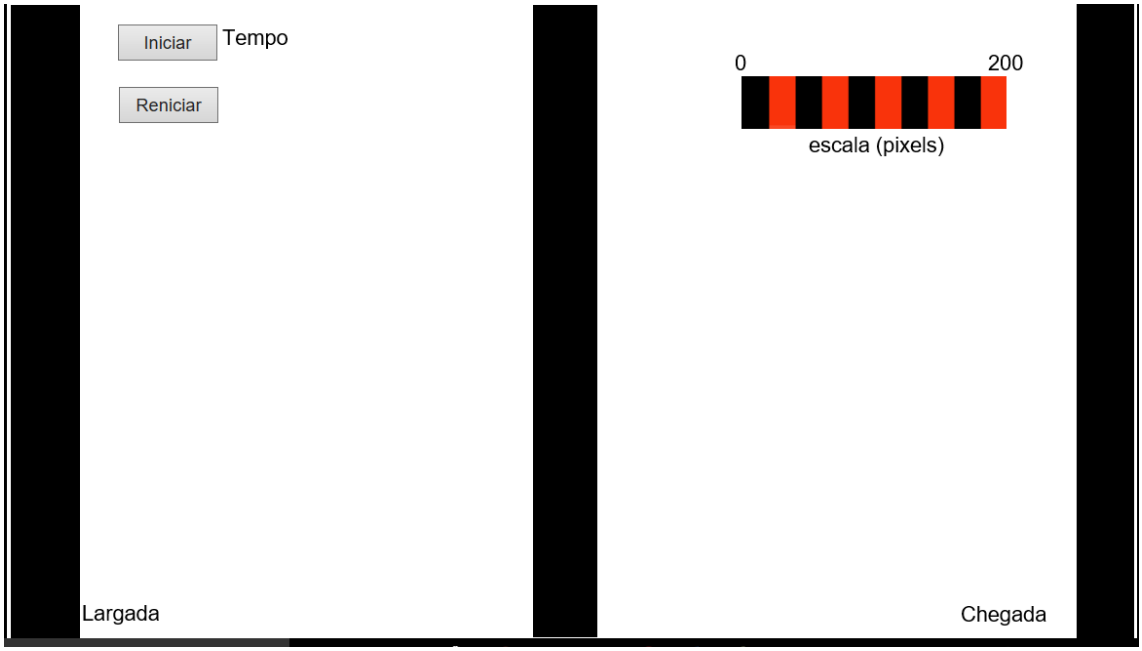


Figura 105 Aparência inicial da pista
Fonte: Autoria Própria

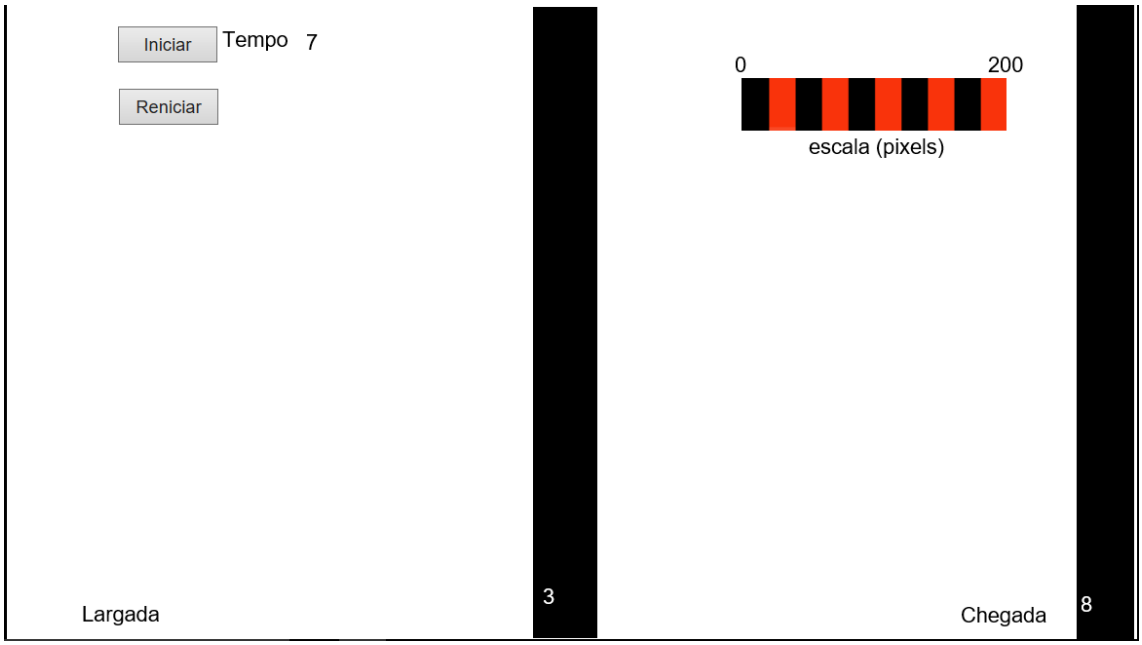


Figura 106 Aparência da pista ao ser iniciado o desafio
Fonte: Autoria Própria

Com relação ao desafio: existem na pista, nas linhas intermediária e de chegada, um timer que mostra o tempo restante até a respectiva linha desaparecer. Ao clicar em iniciar a linha de partida desaparecerá, será iniciada a contagem dos tempos e o robô deverá começar a andar. Próximo aos botões de iniciar e reiniciar tem um cronômetro com o tempo total do desafio. O botão reiniciar volta a pista a condição inicial. O objetivo é fazer o robô chegar à linha de chegada ao tempo de 15 segundos, porém ele deve fazer uma parada de 5 segundos na linha intermediária. Após os 15 segundos a pista desaparecerá e o robô ficará sem referência podendo sair da pista e nesse caso perder o desafio (Figura 107). O desafio será perdido também caso o robô não fique os 5 segundos na linha intermediária.

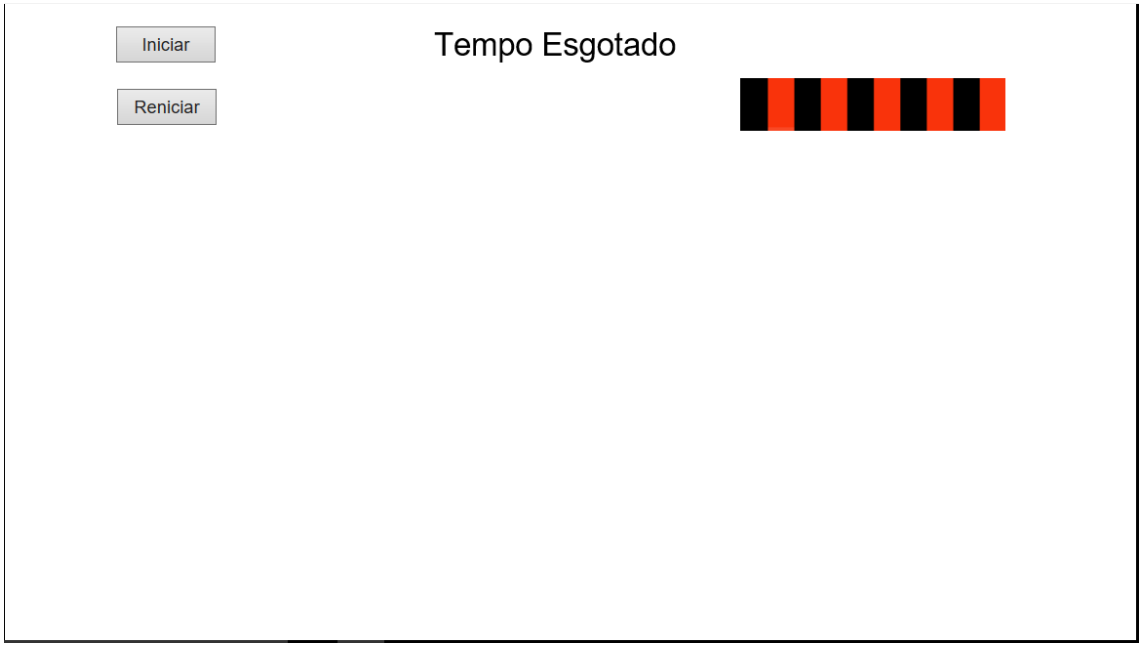


Figura 107 Como fica a pista ao final do tempo
Fonte: Autoria Própria

8.1.5 Para pensar

O que deve ser programado?

Você sabe o que é velocidade média?

É necessário fazer alguma conversão de unidades para cumprir o desafio?

Dica: antes de programar o robô, analisar o cenário, verificar quais as variáveis que deverão ser medidas e calculadas.

Após realizar o desafio, responder as seguintes questões:

1. Qual a velocidade média do robô do ponto de partida até a linha intermediária?
2. Qual a velocidade média do robô da linha intermediária até a linha de chegada?
3. Qual a velocidade média de todo o trajeto da pista?
4. Tendo em vista as respostas anteriores, o que se pode concluir em relação a velocidade média do trajeto realizado pelo robô? Essa conclusão pode ser utilizada para analisar uma viagem real entre duas cidades quaisquer?

8.2 Desafio 2 – Encontro de móveis

Para esse desafio criar um layout conforme mostrado na figura 108

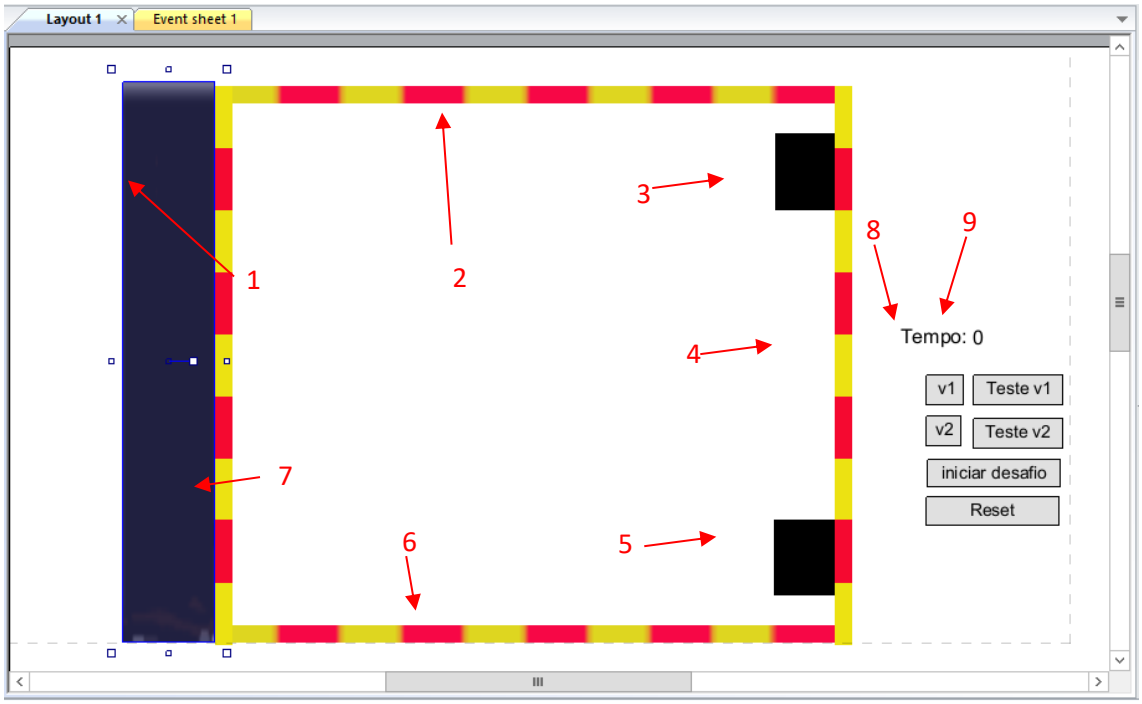


Figura 108 Layout da pista do desafio 2
Fonte: Autoria Própria

Os objetos de 1 a 7 são do tipo *Sprite* os objetos 8 e 9 são do tipo *text* e abaixo desses estão posicionados objetos do tipo *button.* As figuras 109, 110 e 111 mostram como devem ficar as propriedades dos objetos 1 a 9 do layout mostrado

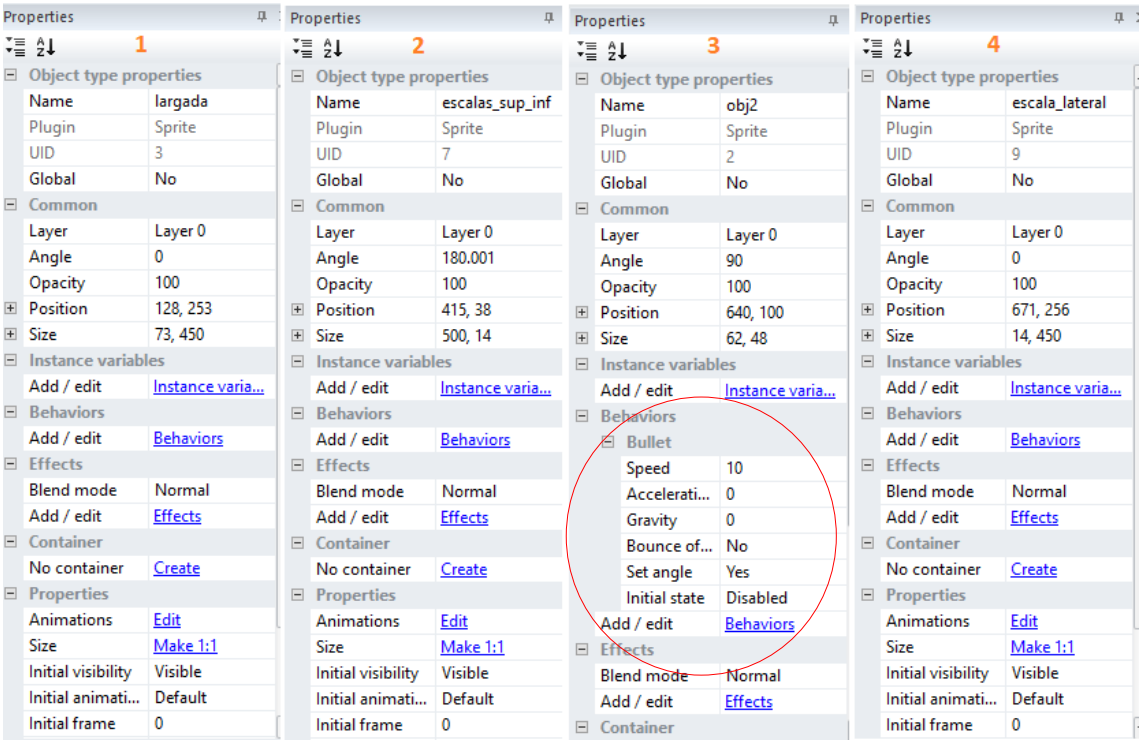


Figura 109 Propriedades dos objetos 1 a 4
Fonte: Autoria Própria

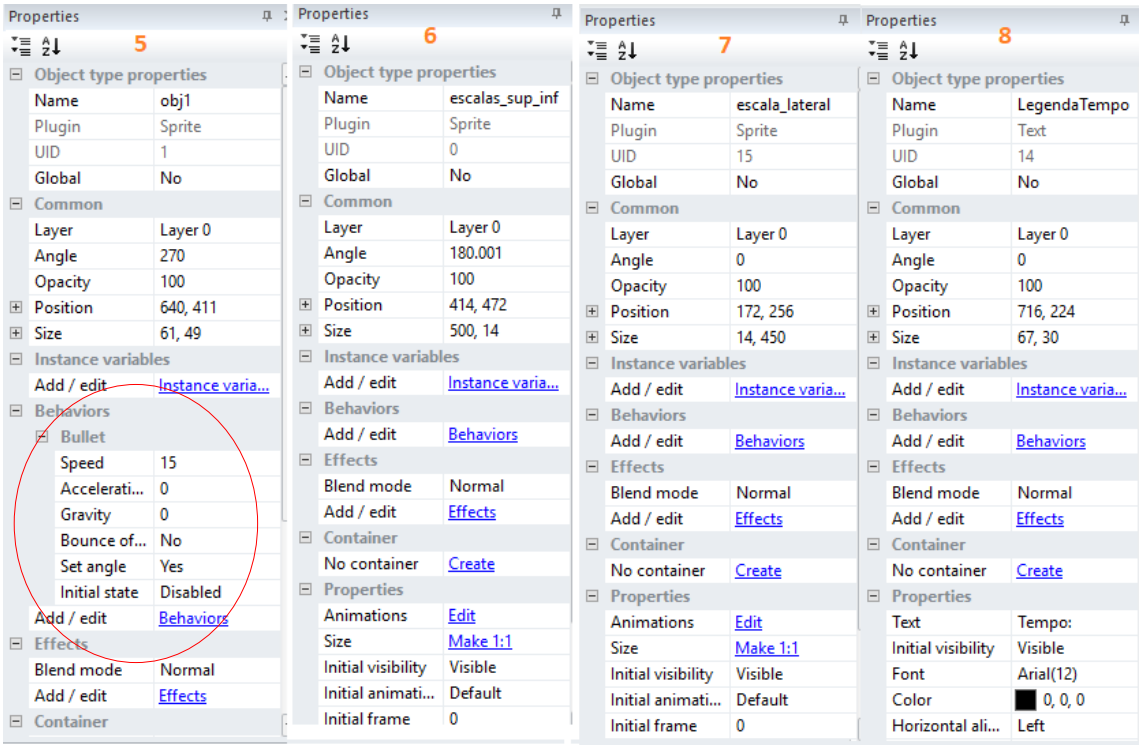


Figura 110 propriedades dos objetos 5 a 8
Fonte Autoria Própria

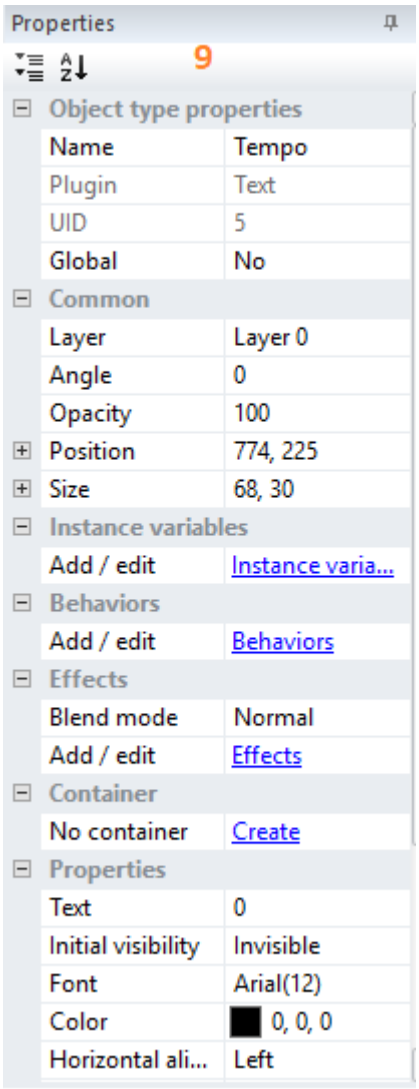


Figura 111 Propriedades do objeto 9
Fonte: Autoria Própria

Nos objetos 3 e 5 foi acrescentado o behavior "*bullet*", o processo para essa tarefa já foi explicado no desafio 1. As propriedades para esse *behavior* pode ser visto em destaque nas figuras 109 e 110 Abaixo dos objetos 8 e 9 do layout estão posicionados os botões de controle, a figura 112 mostra a distribuição desses botões. Para inserir esses botões basta seguir os procedimentos mostrados nas figuras 68 e 69, e modificá-los para ficar com a aparência da figura 112.

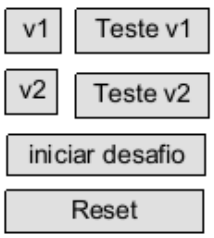


Figura 112 distribuição dos botões de comando Fonte: Autoria Própria

Os eventos serão acrescentados conforme explicado no desafio 1, velocidade média, ficando a janela “*event sheet*” como mostra a figura 85. Seguindo a sequência descrita no desafio 1, velocidade média, é possível programar esses eventos, sendo necessário apenas verificar as condições na janela de propriedades para cada ação de forma que fiquem como aparece na figura 113. Se o professor programou o desafio 1, não terá dificuldades em programar este desafio e os próximos descritos nesse trabalho. Em caso de dúvidas acessar a documentação do software onde poderá ser encontrado a descrição de todos os processos envolvidos na programação.

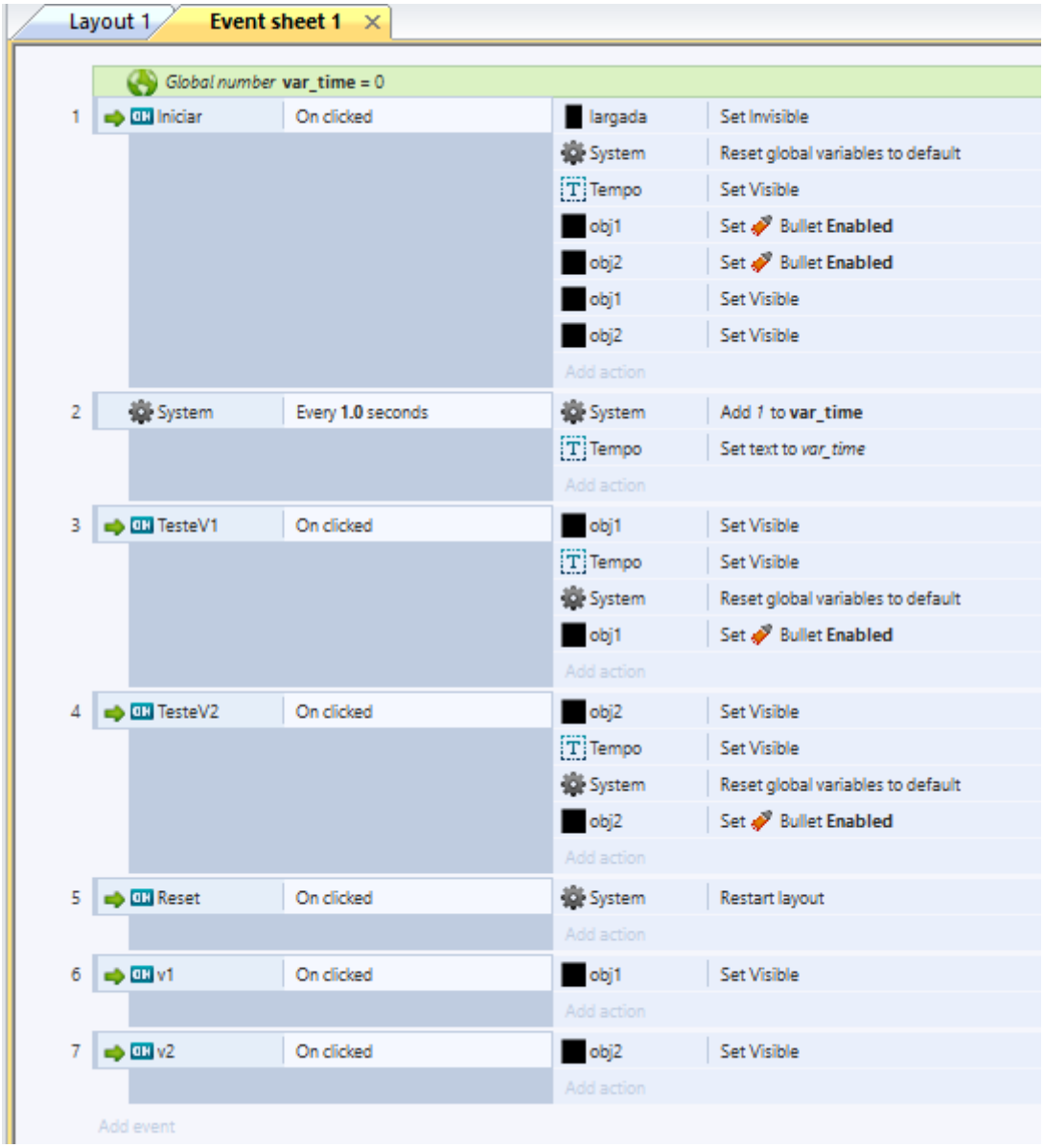


Figura 113 Eventos programados para o desafio 2.
Fonte: Autoria Própria

8.2.1 Objetivo do desafio

Fazer o robô encontrar dois objetos, em movimento, simultaneamente

8.2.2 Objetivo Voltado a Física

Conhecer os cálculos relacionados a velocidade, o instante e a posição em que dois objetos se encontram.

8.2.3 Descrição

Neste desafio o robô irá interagir com a pista mostrada nas figuras 114 e 115

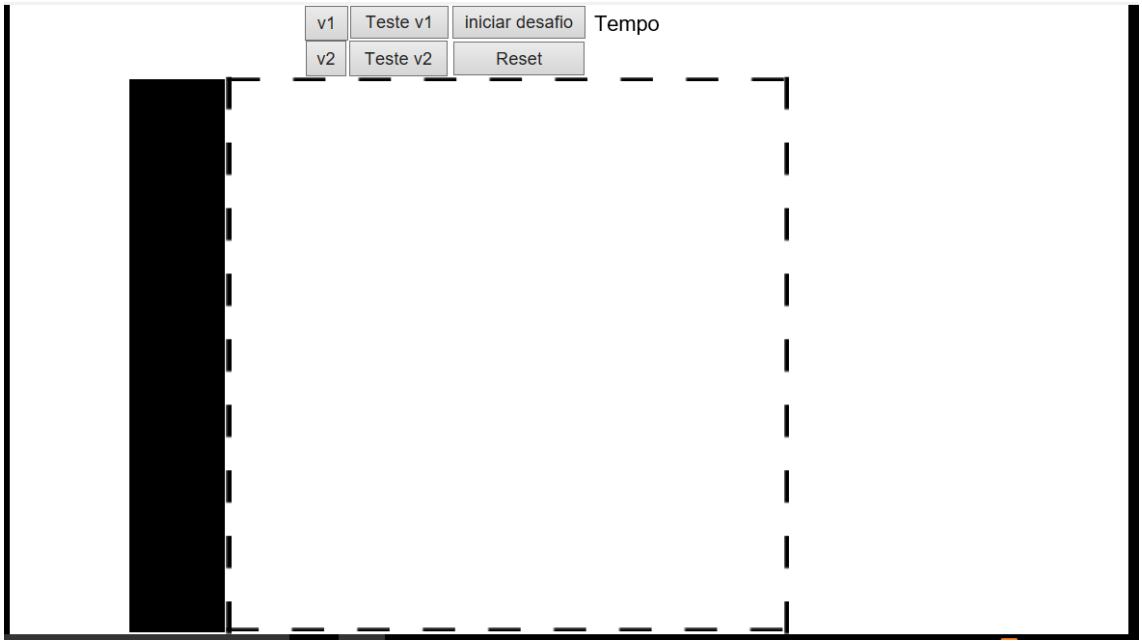


Figura 114 Pista antes de iniciar o desafio
Fonte: Autoria Própria

Com relação ao desafio: A pista é composta por uma linha de largada e dois objetos que se movem um em direção ao outro com velocidades diferentes, conforme visto na figura 115.

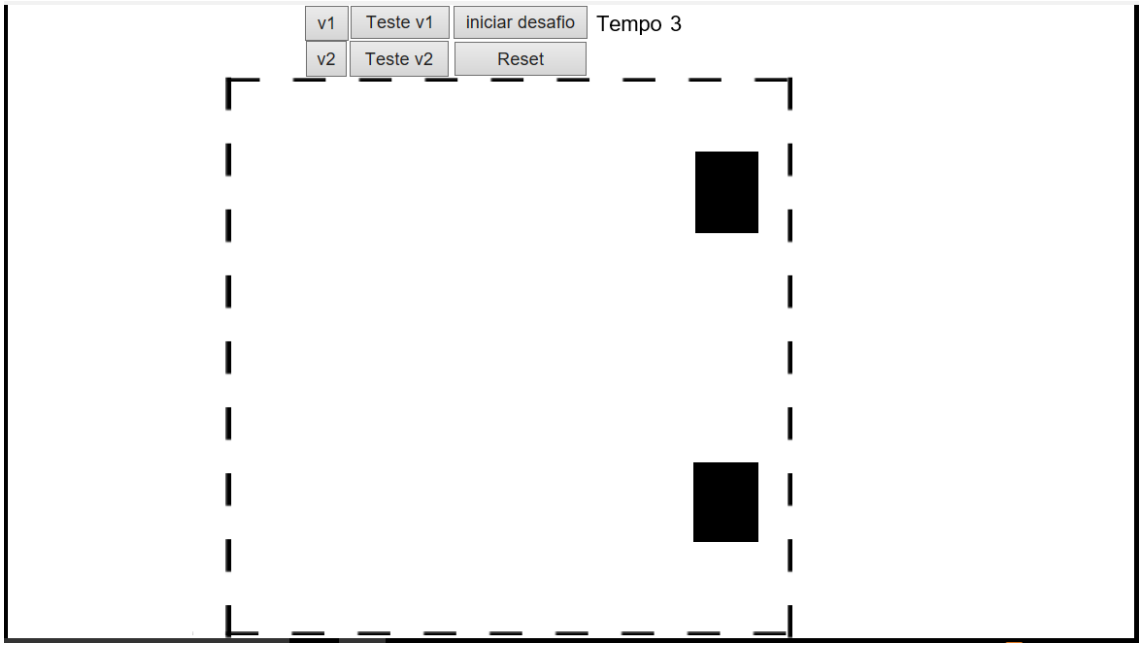


Figura 115 Pista ao iniciar o desafio
Fonte: Autoria Própria

Nas laterais da pista existem retângulos em preto e branco que podem auxiliar a efetuar medidas de comprimento da pista. Acima da pista estão os botões de controle do desafio conforme descrito abaixo:

- Botão v1: faz aparecer na pista o objeto 1;
- Botão v2: faz aparecer na pista o objeto 2;
- Teste v1: inicia o movimento do objeto 1;
- Teste v2; inicia o movimento do objeto 2;
- Iniciar desafio: inicia o movimento dos dois objetos e faz desaparecer a linha de largada que serve como sinal para o robô se movimentar;
- Reiniciar: reinicia o desafio.

Próximo aos botões existe um cronômetro, que se inicia ao clicar em iniciar desafio, teste v1 ou teste v2. Esse instrumento serve para auxiliar as tomadas de tempo do desafio. Os participantes do desafio deverão posicionar o robô de maneira que, ao se movimentar, ele possa encontrar simultaneamente os dois objetos, caso isso não ocorra o desafio não será cumprido.

8.2.4 Para pensar

O que deve ser programado?

Como saber o ponto exato em que os objetos se encontrarão?

É necessário fazer alguma conversão de unidades para cumprir o desafio?

Dica: antes de programar o robô, analisar o cenário, verificar quais as variáveis que deverão ser medidas e calculadas.

Após realizar o desafio, responder as seguintes questões:

1. Quais as variáveis envolvidas nesse desafio?
2. Qual o ponto em comum dessas variáveis para os dois objetos e o robô ao se encontrarem?
3. Em que situação real podemos aplicar a análise utilizada neste desafio?

8.3 Desafio 3 – Movimento Retilíneo Uniforme (MRU) e Movimento Retilíneo Uniformemente Variável (MRUV).

O layout para programação é mostrado na figura 116

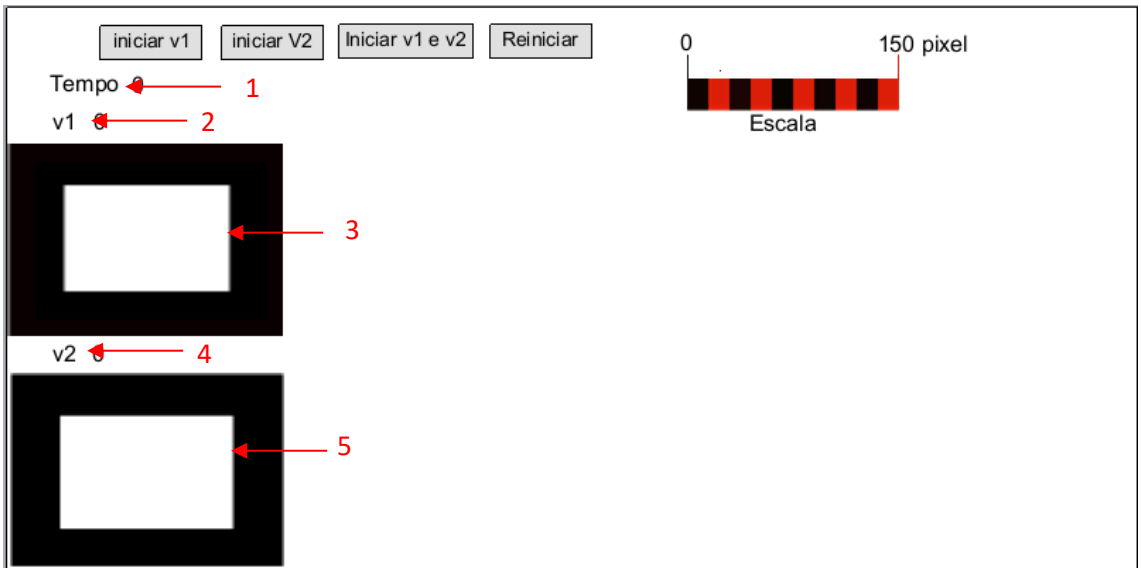


Figura 116 layout a ser construído para o desafio 3
Fonte: Autoria Própria

Na figura 117 são mostradas as propriedades dos objetos marcados na figura 88, os demais objetos não serão descritos pois são de fácil construção e já foram explicados casos parecidos nos desafios 1 e 2.

Nas propriedades é possível ver que foi adicionado o *"Behavior"* "*Bullet*" nos objetos 3 e 5, assim o professor deve estar atento para que as propriedades fiquem como mostra a figura 117. Os eventos programados estão na figura 118. É possível ver que o objeto MRUV é programado para acelerar quando iniciar seu movimento, para isso é acrescentado a ação de ajuste da aceleração (*set bullet acceleration to*) nos eventos 2 e 3. No exemplo a aceleração é ajustada para 2,5 pixels/segundo, porém nada impede que o professor programe outros valores a sua escolha. O objeto MRU deve ser ajustado para manter uma velocidade constante durante o trajeto, para isso é utilizado a ação *set bullet speed to* conforme o evento 3. Assim como a aceleração esse valor pode ser alterado sem problemas.

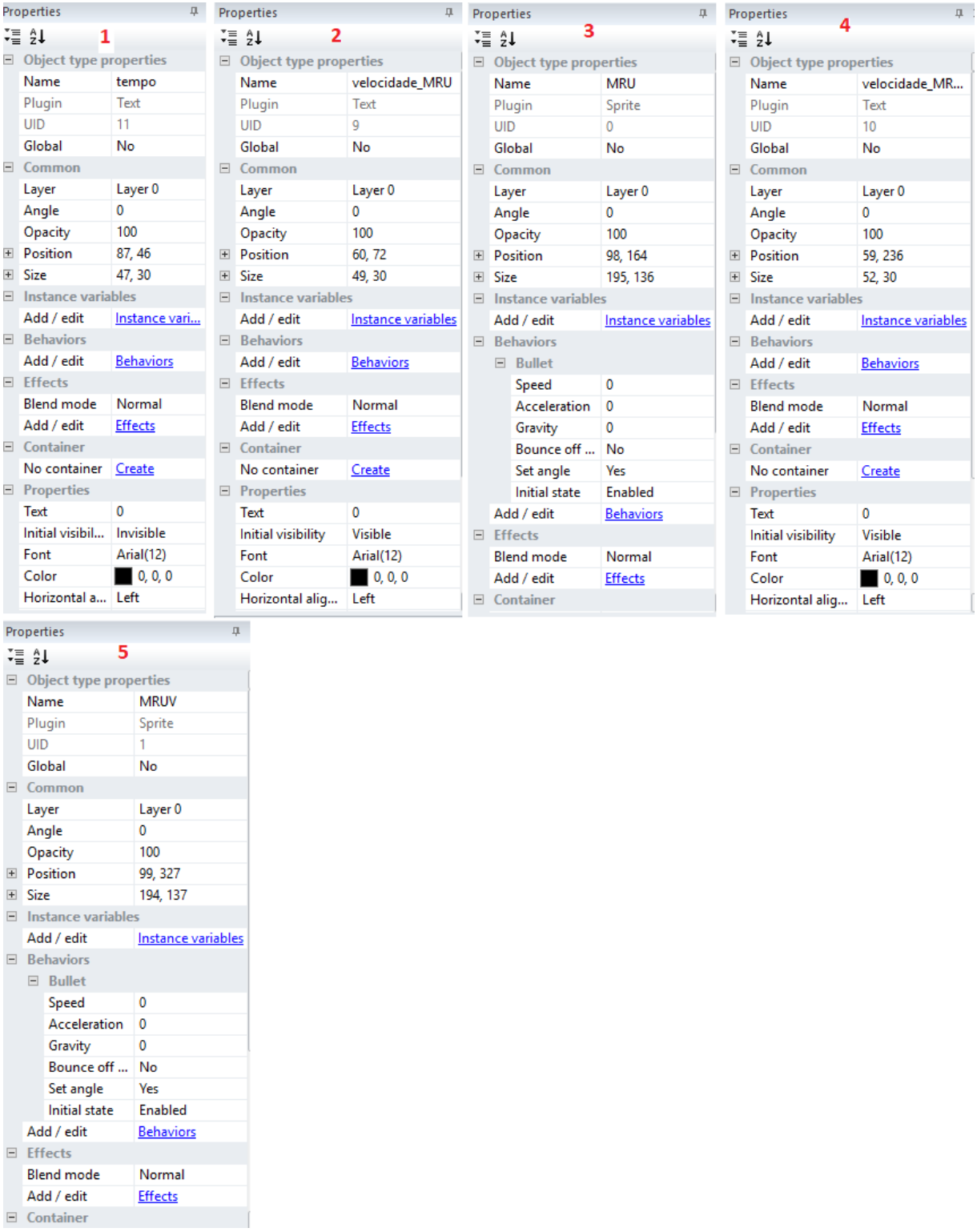


Figura 117 Propriedades dos objetos numerados
Fonte: Autoria Própria

.

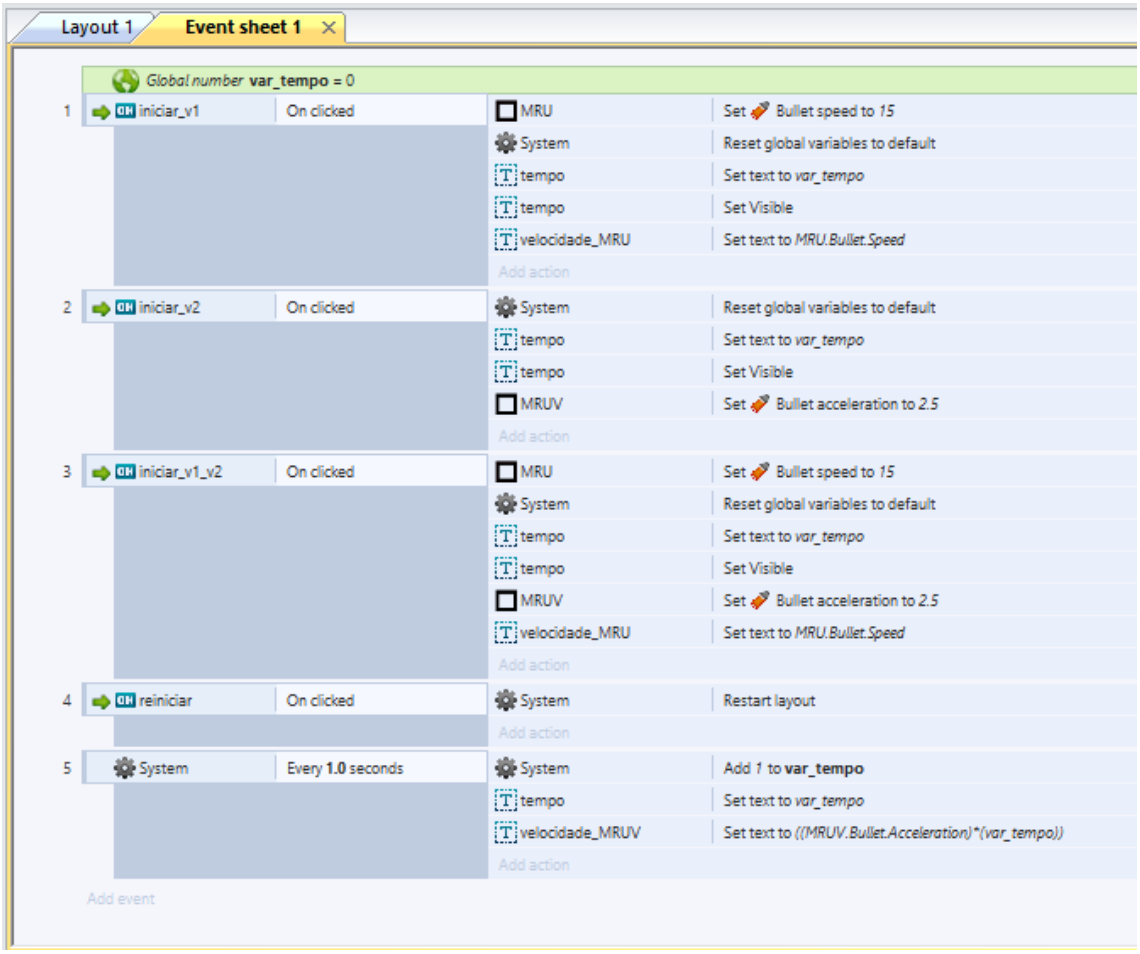


Figura 118 Eventos programados no desafio 3
Fonte: Autoria Própria

8.3.1 Objetivo do desafio

Fazer o robô acompanhar um objeto em Movimento retilíneo uniforme e em movimento retilíneo uniformemente variável.

8.3.2 Objetivo Voltado a Física

Conhecer os conceitos e cálculos relacionados ao Movimento Retilíneo Uniforme (MRU) e Movimento Retilíneo Uniformemente Variado (MRUV).

8.3.3 Descrição

Este desafio é composto por uma pista conforme mostrado na figura 119.

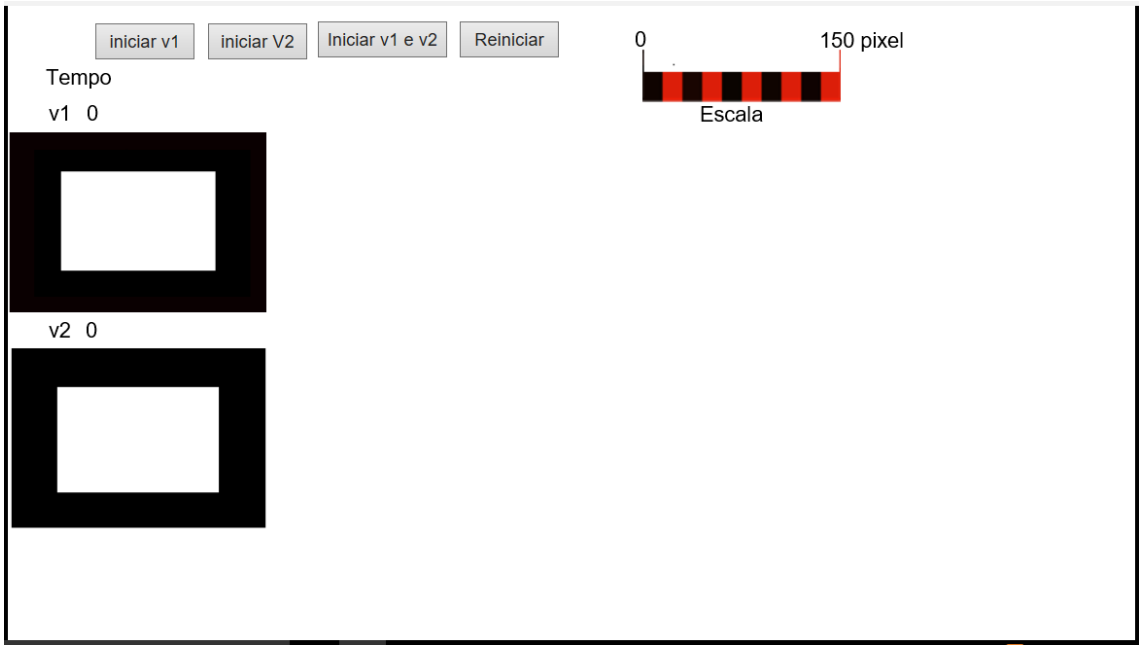


Figura 119 Layout da pista antes de iniciar o desafio
Fonte: Autoria Própria

Com relação ao desafio: A pista é composta por dois objetos, sendo que um se move em linha reta com velocidade constante e o outro se move em linha reta com certa aceleração, medida em pixels por segundo ao quadrado. Os botões acima da pista controlam os movimentos dos objetos. Acima de cada objeto encontra-se um "velocímetro", identificado por v1 e v2, indicando a velocidade dos objetos 1 e 2 respectivamente. As velocidades são medidas em pixels/segundo. Ao lado dos botões tem uma escala que poderá ser utilizada para medir as distâncias da pista.

Abaixo dos botões existe um cronômetro, que se inicia quando qualquer um dos objetos começar a se mover, este instrumento será útil durante o desafio. Os participantes do desafio deverão programar o robô para que esse acompanhe o objeto em movimento, sem que o sensor do robô encontre as bordas pretas do objeto, ou seja ele deverá acompanhar o movimento do objeto com o sensor sempre dentro do retângulo branco formado pelo objeto. Caso o sensor entre em contato com as bordas do objeto o desafio é considerado não concluído.

8.3.4 Para pensar

O que deve ser programado?

Como saber que o robô acompanhará o objeto sem tocar nas bordas?

É necessário fazer alguma conversão de unidades para cumprir o desafio?

Dica: antes de programar o robô, analisar o cenário, verificar quais as variáveis que deverão ser medidas e calculadas.

Após realizar o desafio, responder as seguintes questões:

1. Quais as variáveis envolvidas nesse desafio?
2. Qual a diferença entre o movimento do objeto 1 e do objeto 2?
3. Qual a dificuldade em fazer o robô seguir o objeto 1? E em relação ao objeto 2 qual a dificuldade?
4. Ao ligar um carro para sair da garagem, em linha reta, ele se comporta como o objeto 1 ou como o objeto 2? Justifique.

8.4 Desafio 4 – Queda livre

O layout a ser programado é mostrado na figura 120.

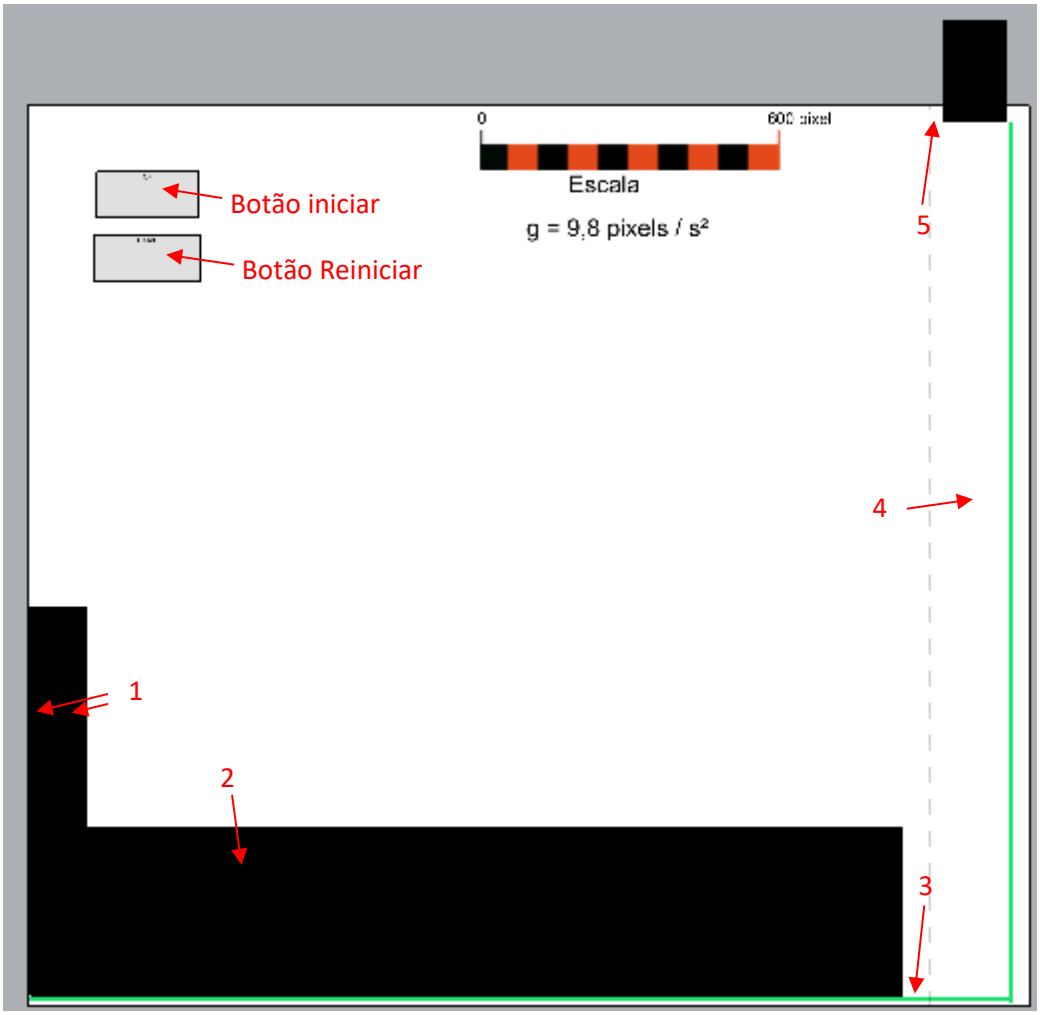


Figura 120 Distribuição dos objetos a serem programados
Fonte: Autoria Própria

Nesse layout o objeto 5 está posicionado acima da área visível na tela do navegador, assim como a resolução do layout que é maior, a posição do objeto acima do layout permite um tempo maior de queda, facilitando ao aluno a programação de velocidade do robô. Durante a construção do layout prestar atenção na posição do objeto 5, que deve estar com seu canto posicionado no início do objeto 4 (parede) que é a referência para o aluno medir a altura da queda. Os objetos não numerados nesse desafio são a escala e as legendas e dados adicionais, a programação desses objetos já foram ensinados anteriormente.

A propriedade dos objetos numerados é mostrada na figura 121. Nas propriedades dos objetos 3 e 5 estão destacados os parâmetros para os comportamentos chamados de *behaviors, Solid e Physics.* Os *behaviors* são

adicionados conforme visto nas pistas anteriores. Esses comportamentos que garantem a “precisão” da simulação de queda livre. Analisando as propriedades físicas acrescentadas por meio desses “*behaviors*” pode ser visto que há a possibilidade de “calibrar” a densidade, fricção, elasticidade para o objeto, dando mais realidade a simulação. O objeto chão (3) está com a propriedade *“Immovable*” ajustada para sim (yes) e o objeto (5) tem essa propriedade ajustada para não (no), esses ajustes garantem que o chão não irá “cair” e que o objeto sofra a queda livre. A propriedade *“Initial state”* do *behavior physics* também está desabilitada (*Disabled*) para o objeto 5 de forma a garantir que o objeto fique parado no alto da pista até o desafio ser iniciado

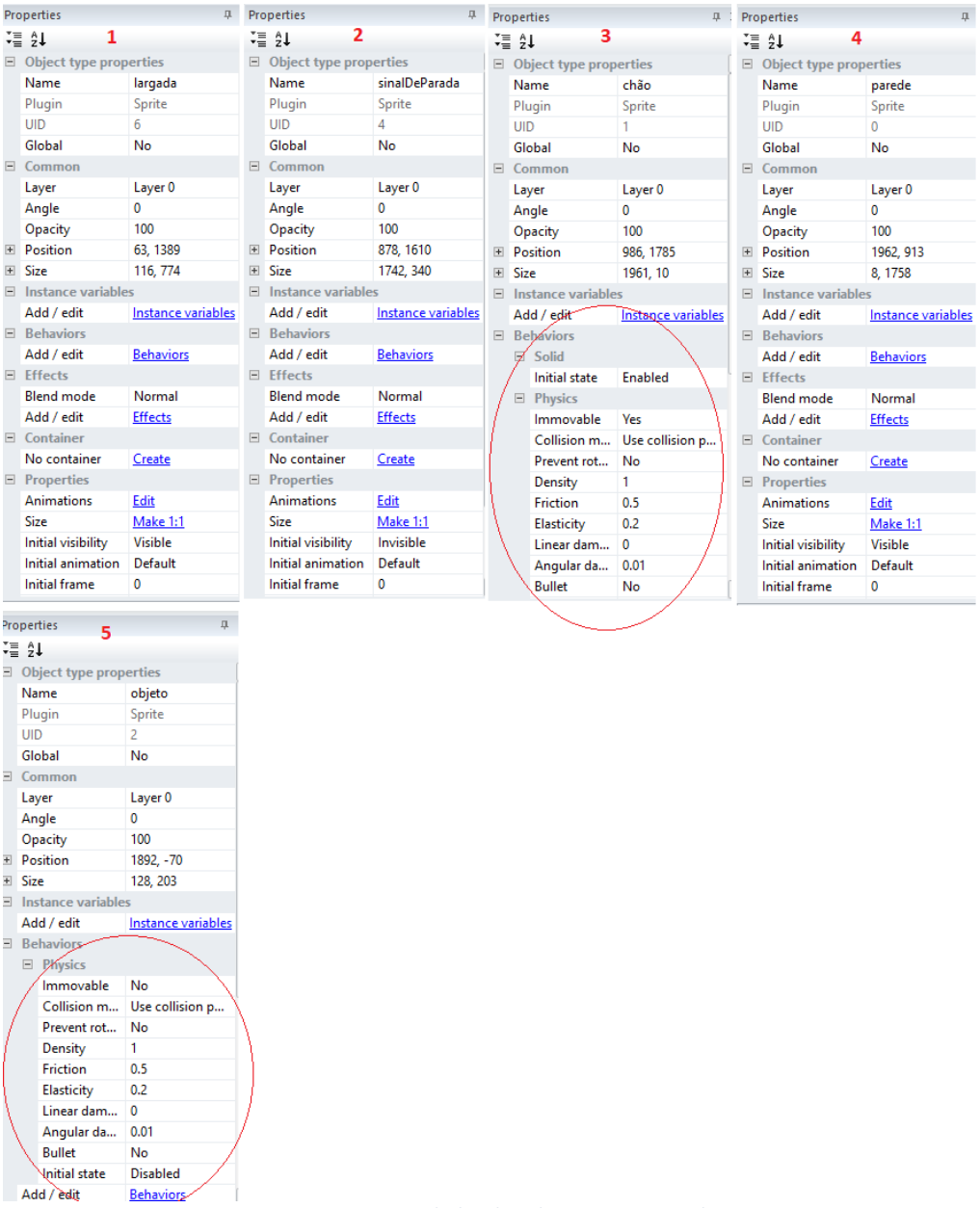


Figura 121 Propriedades dos objetos programados.
Fonte: Autoria Própria

Na figura 122 são mostrados os eventos para esse projeto. Para o objeto será habilitado a física da simulação e ajustado o valor da gravidade em 9,8 pixels/s² (evento 1). Nada impede de o professor simular outros valores de gravidade no experimento, inclusive pode ser testado com os alunos, por exemplo, a queda livre do objeto com a gravidade da lua. O evento 2 e responsável por aparecer o objeto 2 que fará o robô parar na posição que estiver quando o objeto tocar o “chão”.

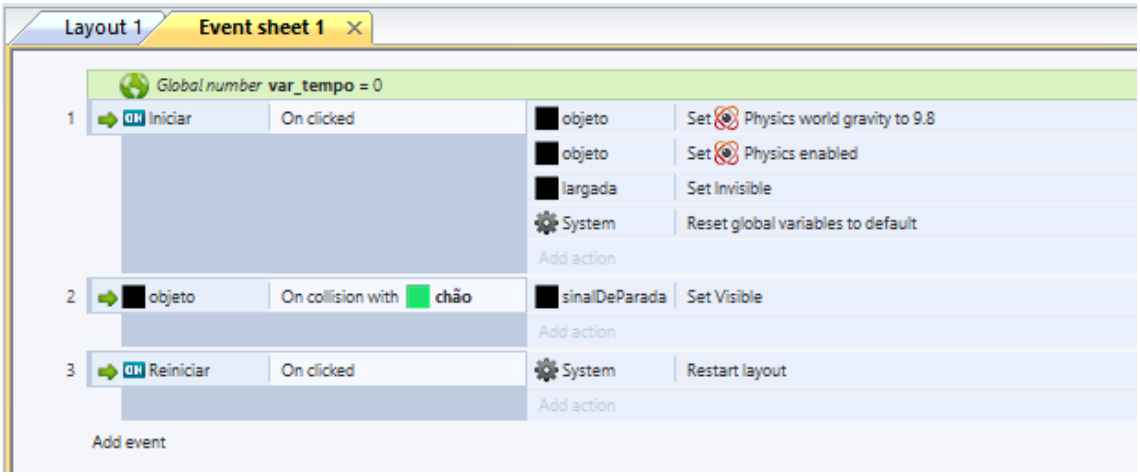


Figura 122 Evento do desafio queda livre.
Fonte: Autoria Própria

8.4.1 Objetivo do desafio

Fazer o robô interceptar um objeto em queda livre

8.4.2 Objetivo Voltado a Física

Entender o conceito de queda livre e os cálculos envolvidos.

8.4.3 Descrição

Este desafio é composto por uma pista conforme mostrado nas figuras 123 e 124.

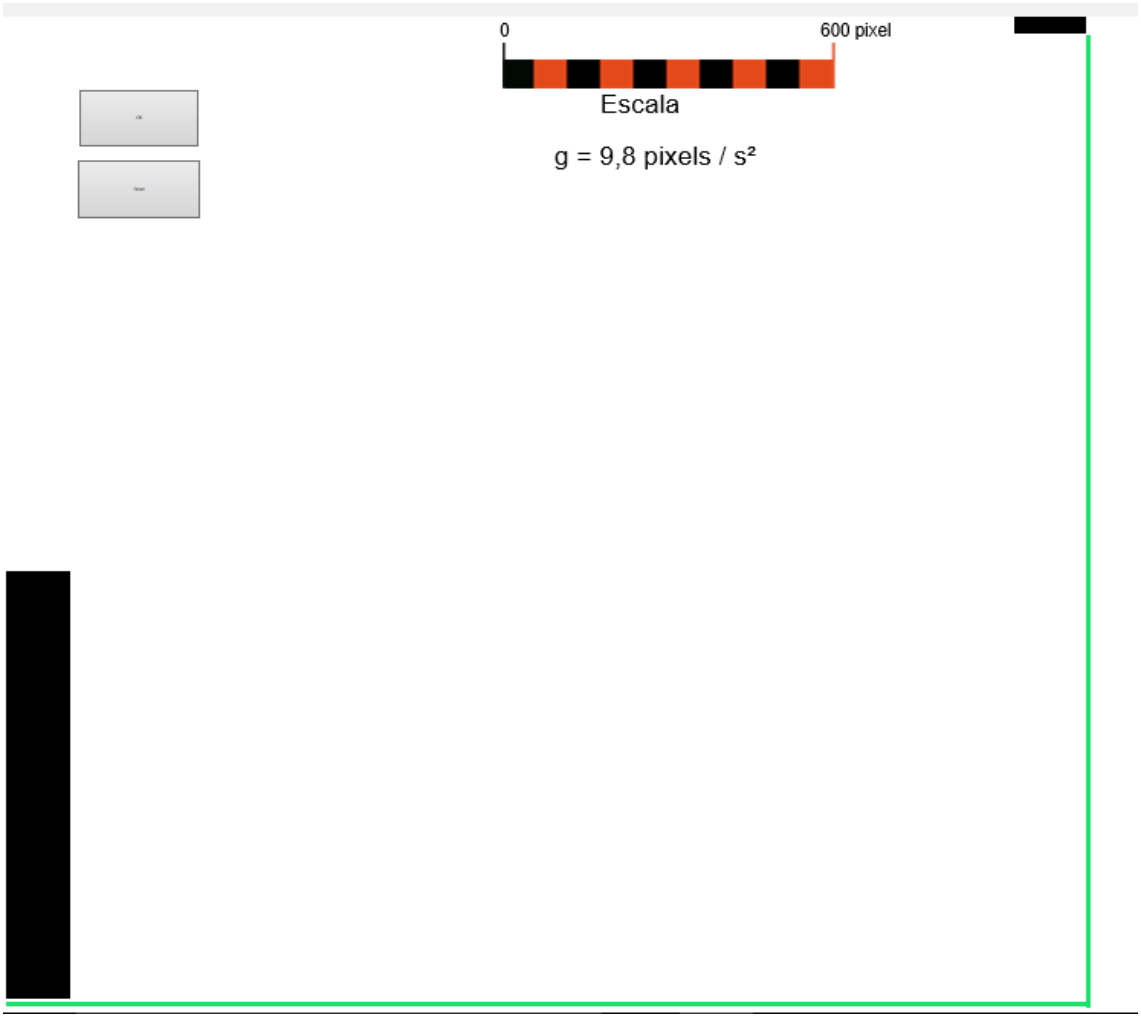


Figura 123 Layout da pista antes de iniciar ao desafio
Fonte: Autoria Própria

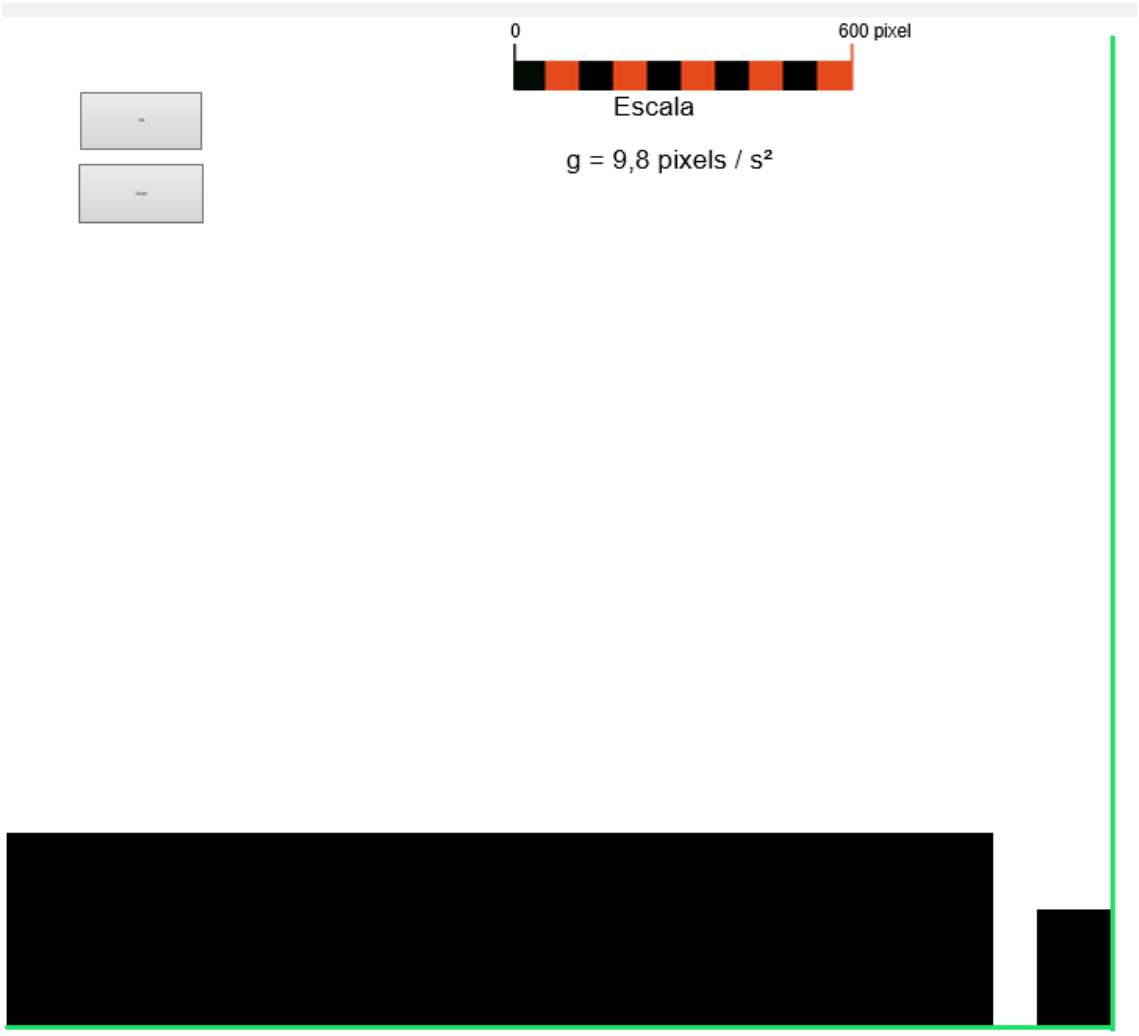


Figura 124 Layout da pista após o objeto tocar o "chão"
Fonte: Autoria Própria

.

Com relação ao desafio: A pista é composta por um objeto, que é lançado em queda livre com a aceleração da gravidade medida em pixels por segundo ao quadrado. Os botões acima da pista controlam o momento em que o objeto começa a cair e o reinício do desafio. No lado oposto ao da queda do objeto tem uma faixa onde o robô deverá ser posicionado antes do início do desafio, essa faixa desaparecerá simultaneamente com o início da queda do objeto. Ao lado dos botões tem uma escala que poderá ser utilizada para medir as distâncias da pista e uma legenda que mostra o valor adotado na pista virtual. A linha verde abaixo da tela serve como referência do solo para o objeto e a linha verde na lateral, próximo ao objeto em queda, pode ser utilizada como referência para medir a altura que o objeto se encontra. O jogador deve programar o robô para que seu sensor entre em contato com o objeto quando ele toca o solo, ou um pouco antes, caso contrário perde o desafio.

Quando o objeto tocar o solo aparecerá uma faixa da cor preta que fará o robô parar na posição que estiver quando esse evento ocorrer.

8.4.4 Para pensar

O que deve ser programado?

Como saber que o robô irá chegar no momento exato para interceptar o objeto?

É necessário fazer alguma conversão de unidades para cumprir o desafio?

Dica: antes de programar o robô, analisar o cenário, verificar quais as variáveis que deverão ser medidas e calculadas.

Após realizar o desafio, responder as seguintes questões:

1. Quais as variáveis envolvidas nesse desafio?
2. Para concluir o desafio em algum momento foi necessário saber a massa do objeto em queda livre? Justifique.

8.5 Desafio 5 – Lançamento de projéteis

A figura 125 mostra a disposição dos objetos no layout. As propriedades dos objetos numerados são mostrados na figura 126. O objeto 1 foi acrescentado o comportamento Bullet e ajustado a velocidade para o valor 80, a gravidade para 9,8 e deixado a condição inicial desabilitada. Prestar atenção na propriedade *Common,* do objeto 1, onde o ângulo foi ajustado paro o valor de 315°, isso garante uma inclinação de 45° em relação ao objeto 2 que serve de referência (chão). Os demais objetos sem numeração são legendas e escalas que podem ser construídas seguindo as instruções do primeiro desafio.

Os eventos aparecem na figura 127. São apenas dois eventos, tornando a programação muito simples.

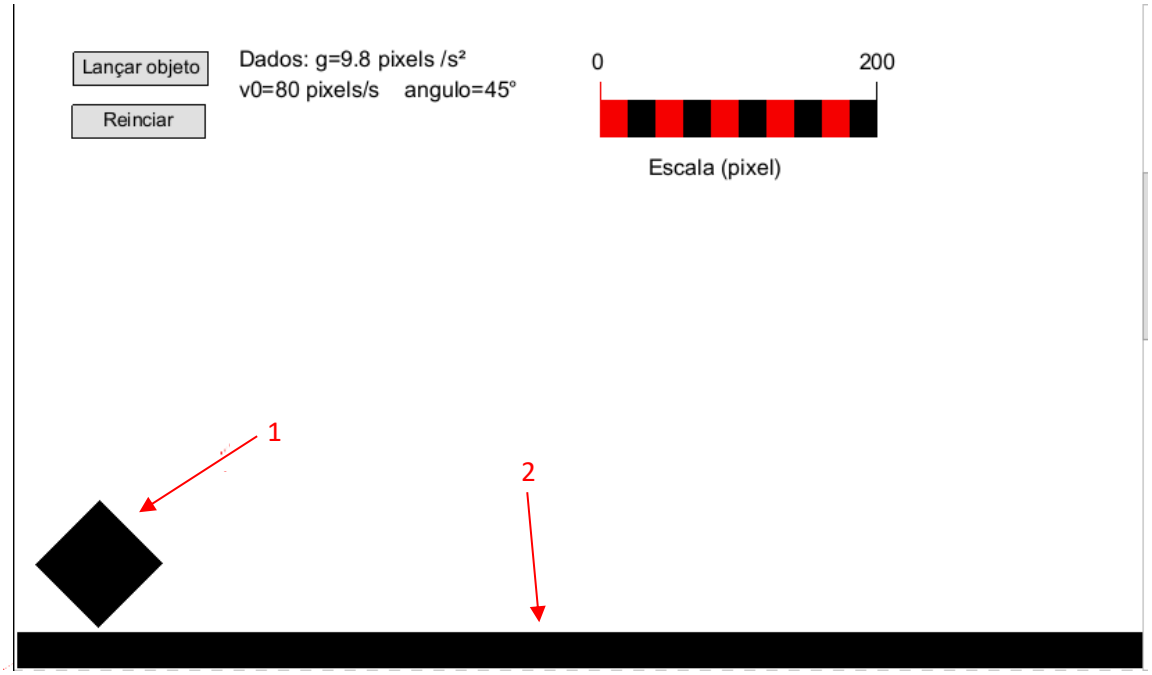


Figura 125 disposição dos objetos a serem programados.
Fonte: Autoria Própria

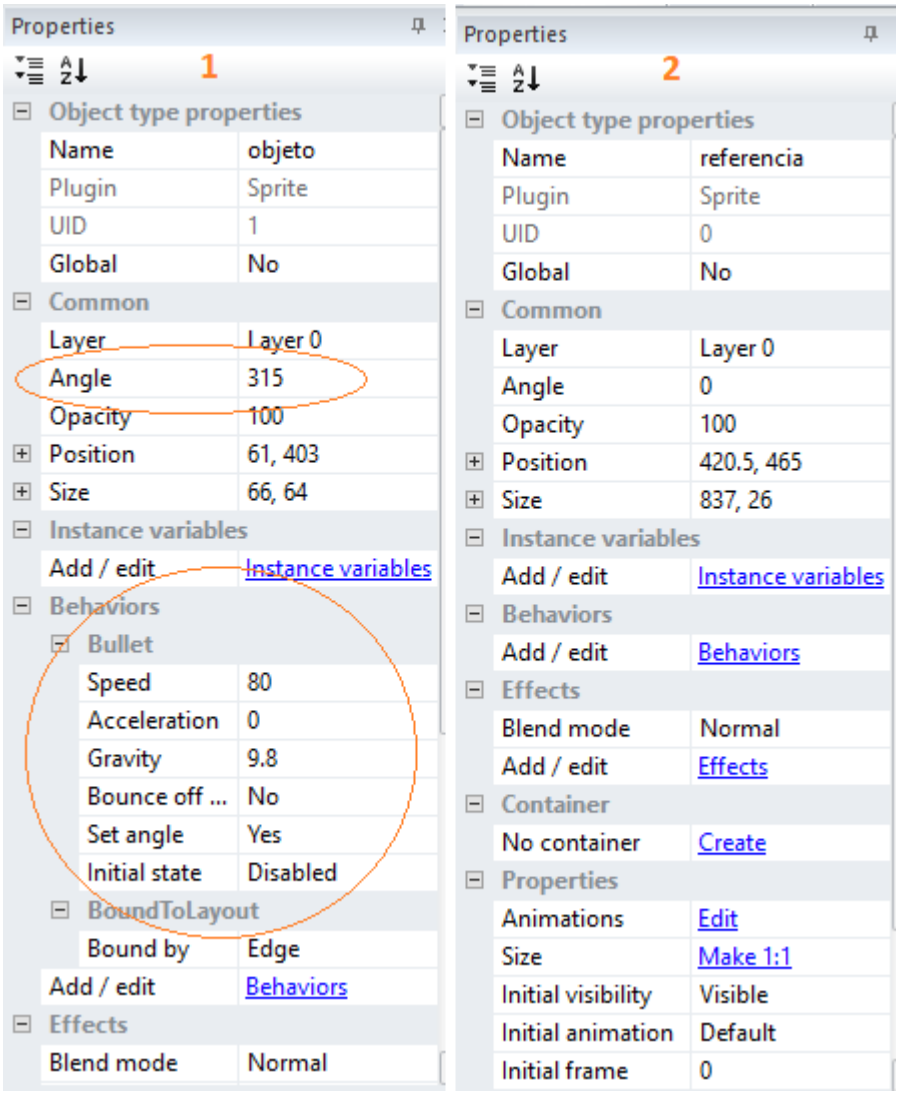


Figura 126 Propriedades dos objetos 1 e 2.
Fonte: Autoria Própria

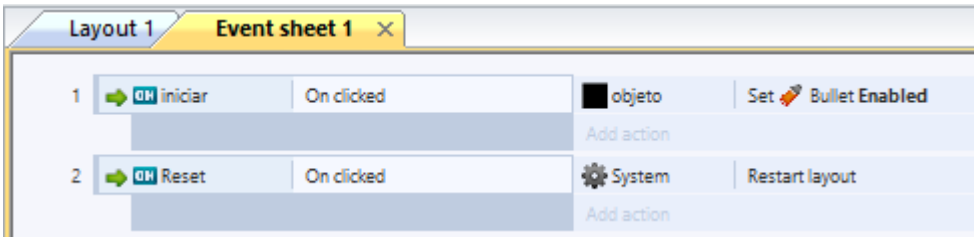


Figura 127 Eventos programados
Fonte: Autoria Própria

8.5.1 Objetivo do desafio

Fazer o robô se locomover simultaneamente com um objeto lançado a um certo ângulo e interceptá-lo no final do trajeto.

8.5.2 Objetivo Voltado a Física

Conhecer os conceitos relacionados ao lançamento de projéteis ou lançamento obliquo e os cálculos relacionados.

8.5.3 Descrição

Este desafio é composto por uma pista conforme mostrado na figura 128.

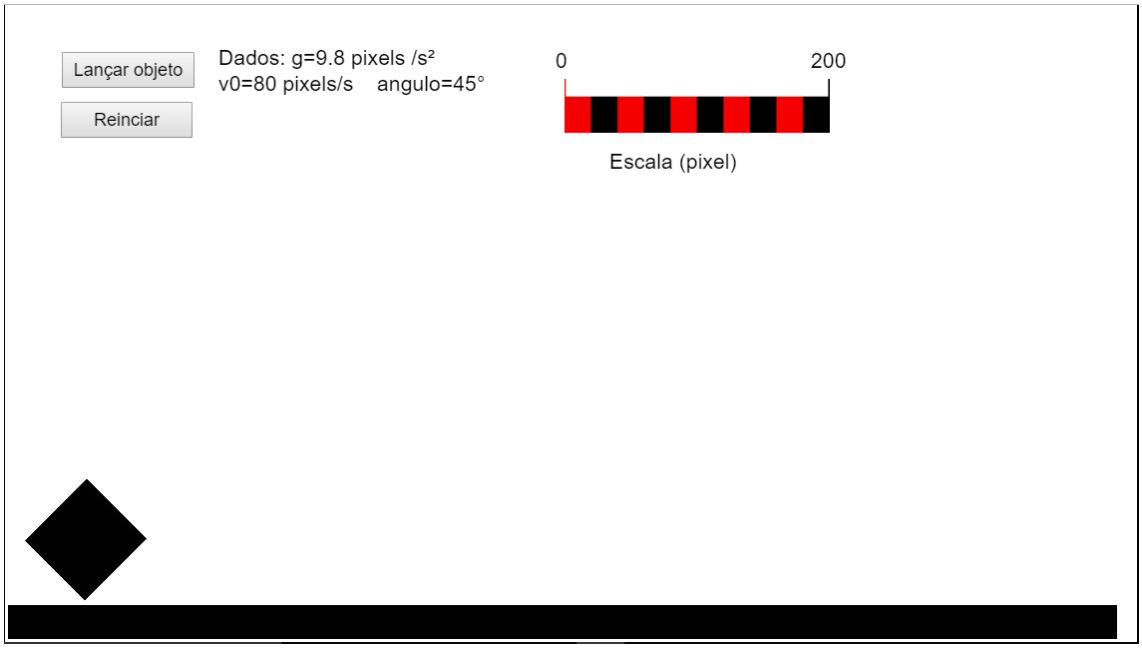


Figura 128 Layout da pista de lançamento de projéteis
Fonte: Autoria Própria

Com relação ao desafio: A pista é composta por um objeto que será lançado com uma velocidade inicial de 80 pixels por segundo e com um ângulo de 45° e por uma linha inferior que serve como referência do solo para o objeto.

Acima da pista estão os botões que controlam o desafio, sendo um para lançar o objeto e outro para reiniciar o desafio. Ao lado dos botões estão os dados iniciais do desafio e a escala utilizada na pista.

Para iniciar o desafio o participante deverá posicionar o robô com o sensor coberto pelo objeto. Ao lançar o objeto o robô deverá andar em linha reta e encontrar o objeto no final da pista.

8.5.4 Para pensar

O que deve ser programado?

Como saber que o robô irá chegar junto com o objeto no final da pista?

É necessário fazer alguma conversão de unidades para cumprir o desafio?

Dica: antes de programar o robô, analisar o cenário, verificar quais as variáveis que deverão ser medidas e calculadas.

Após realizar o desafio, responder as seguintes questões:

1. Quais as variáveis envolvidas nesse desafio?
2. Qual a diferença entre o movimento do objeto e do robô? Explique com suas palavras como eles chegam simultaneamente ao final da pista.
3. Qual a dificuldade em fazer o robô encontrar o objeto?
4. Cite duas modalidades esportivas que se utilizam dos conceitos vistos neste desafio. Justifique.

8.6 Desafio 6 – Movimento Circular Uniforme

Este desafio foi projetado no *site Construct 3*, que pode ser acessado pelo endereço www.construct.net. Esse site traz uma versão online, atualizada, do software utilizado para programar as pistas. Os comandos são os mesmos da versão utilizada nos outros desafios, foi escolhido esta plataforma simplesmente por permitir desenhar objetos circulares, que será útil nesse desafio. A disposição dos elementos da pista é mostrada na figura 129. Ao posicionar os objetos tomar cuidado com a simetria da pista para facilitar a programação da velocidade do robô de maneira que ele possa passar pela abertura do obstáculo circular. Fica a critério do professor a construção e medidas que serão utilizadas. As propriedades são exibidas na figura 130 e os eventos na figura 131. Em destaque nas propriedades estão os ajustes do comportamento (*behavior*) *Rotate* adicionado ao obstáculo circular. Nos eventos pode se ver que o valor da rotação do obstáculo será de 10°/s, porém o professor pode mudar esse valor. Ao começar a rotação esse valor aparecerá na tela por meio da ação programada no evento 1 "*set text to obstáculo.Rotate.Speed*"

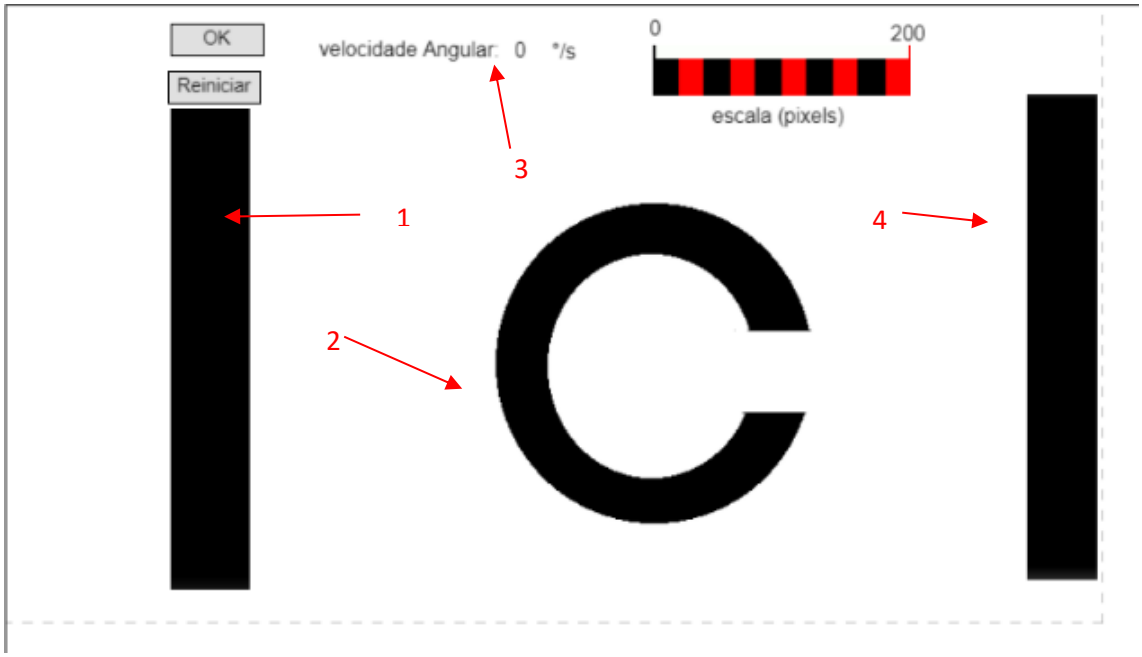


Figura 129 Disposição dos objetos
Fonte: Autoria Própria

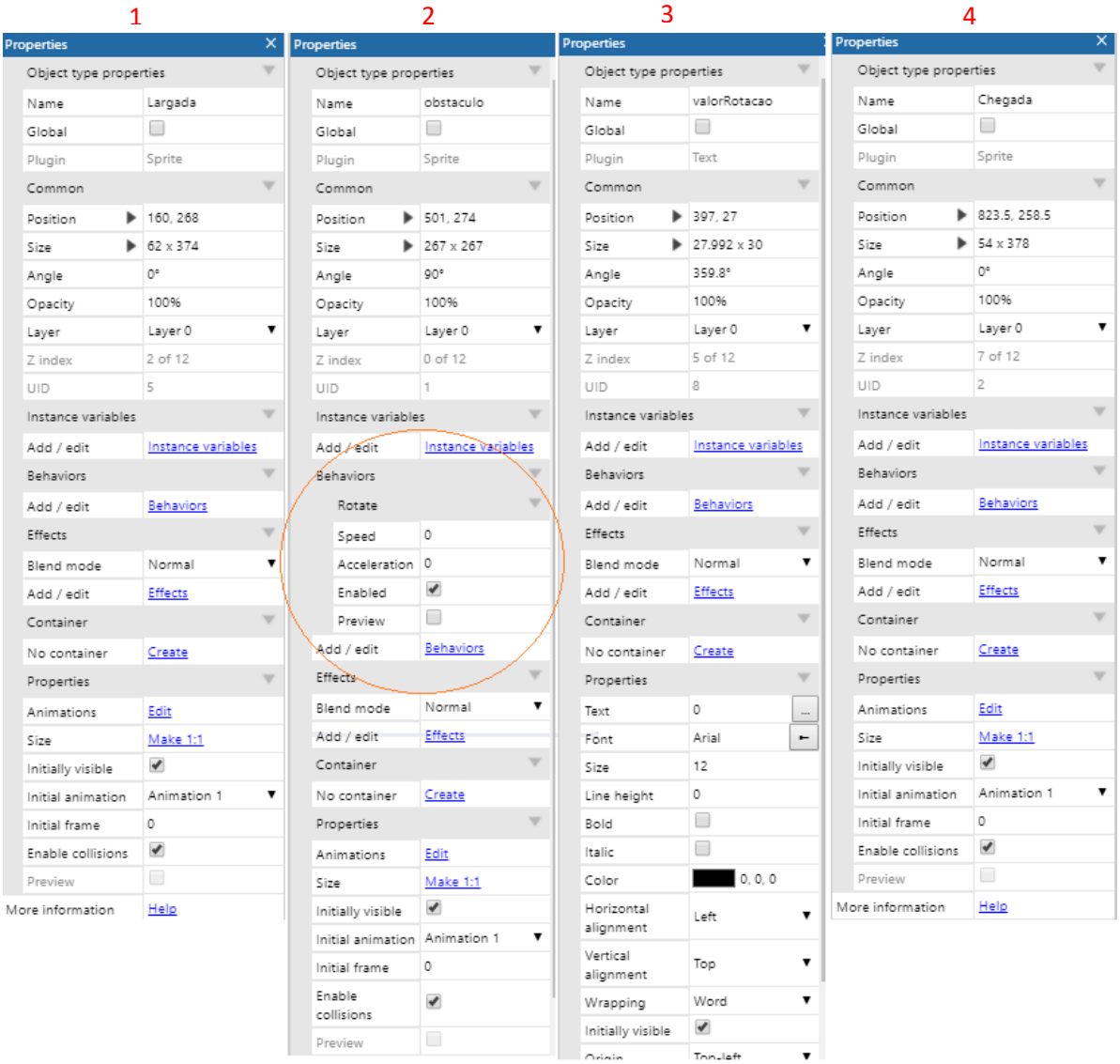


Figura 130 Propriedades dos objetos.
Fonte: Autoria Própria

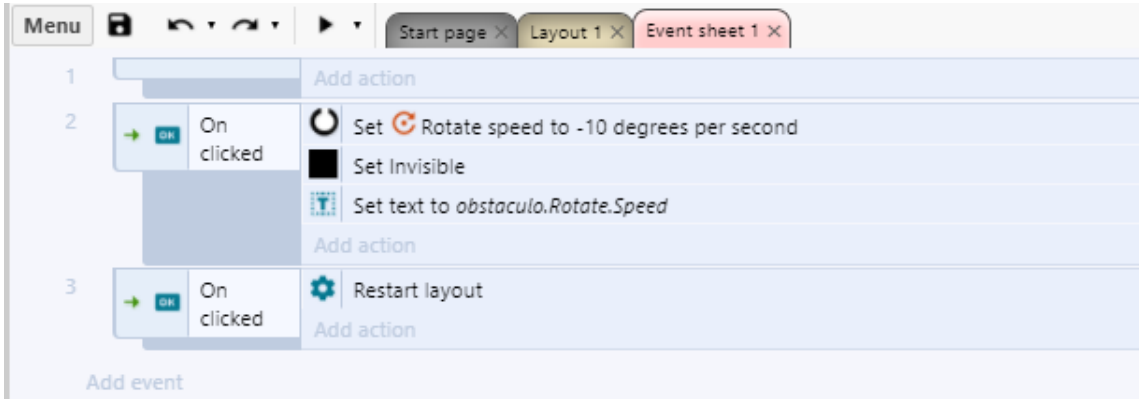


Figura 131 Eventos programados.
Fonte: Autoria Própria

8.6.1 Objetivo do desafio

Fazer o robô percorrer a pista em linha reta, sem tocar nas bordas do objeto em rotação no centro da pista.

8.6.2 Objetivo Voltado a Física

Conhecer o movimento circular uniforme, os cálculos envolvidos e a sua relação com o movimento linear.

8.6.3 Descrição

Este desafio é composto por uma pista conforme mostrado nas figuras 132 e 133.

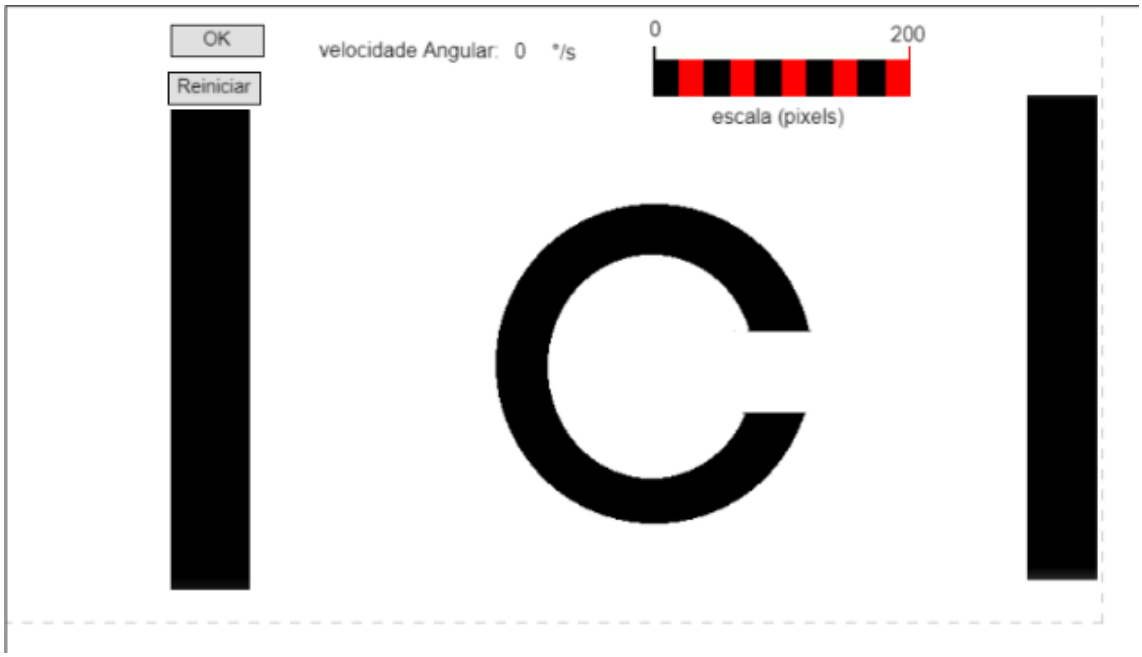


Figura 132 Layout da pista antes de iniciar o desafio
Fonte: Autoria Própria

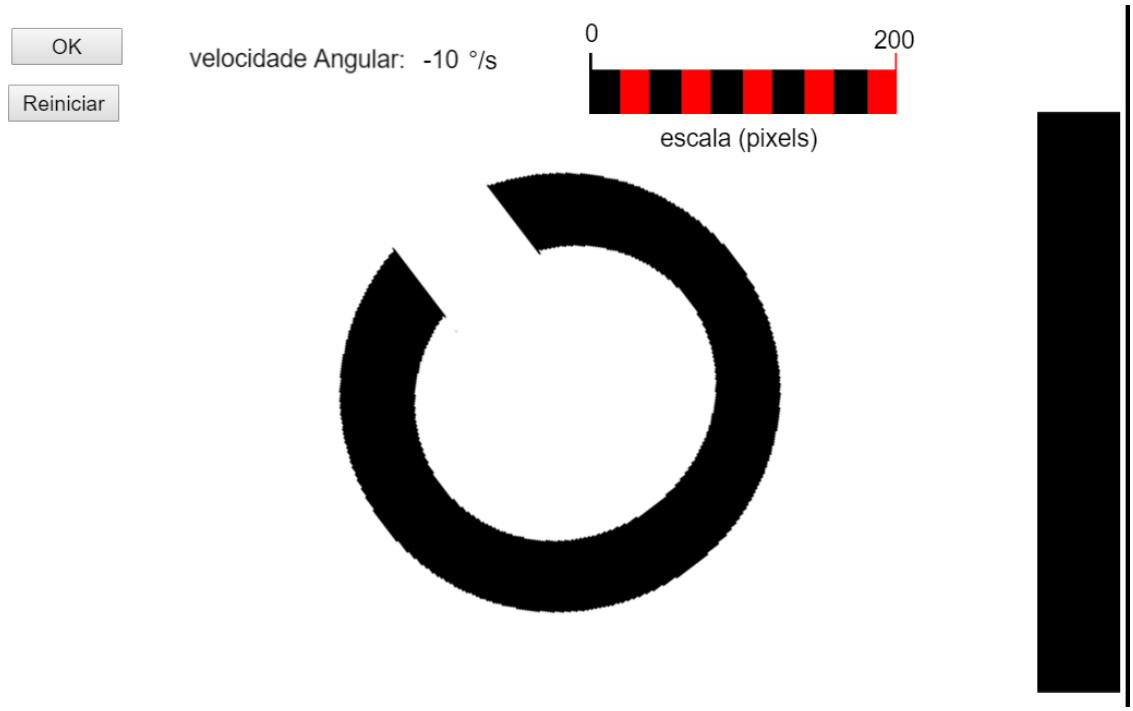


Figura 133 Layout da pista após iniciar o desafio
Fonte: Autoria Própria

Com relação ao desafio: A pista é composta por um círculo com uma abertura lateral, uma linha de partida e uma linha de chegada. O círculo fica localizado no centro da pista e as linhas de partida e chegada ficam nas laterais.

O robô deverá ser programado para sair da linha de partida, passar por dentro do círculo, sem que o sensor encontre nas bordas fechadas, e chegar à linha de chegada, passando pela pista em linha reta. Ao clicar no botão “OK” a linha de partida desaparecerá e o círculo começará o movimento de rotação. Será mostrada na pista a velocidade angular do círculo em graus por segundo. Além do botão para iniciar, existe um botão para reiniciar e uma escala para ajudar a medir a pista. A ideia é fazer o robô entrar e sair pela mesma abertura, cruzando a pista.

8.6.4 Para pensar

O que deve ser programado?

Como fazer o robô cruzar toda pista sem tocar nas bordas do círculo?

É necessário fazer alguma conversão de unidades para cumprir o desafio?

Dica: antes de programar o robô, analisar o cenário, verificar quais as variáveis que deverão ser medidas e calculadas.

Após realizar o desafio, responder as seguintes questões:

1. Quais as variáveis envolvidas nesse desafio?

2. O movimento do círculo é uniforme? O que ocorre com o módulo da velocidade de rotação do círculo? Existe aceleração nesse movimento?
3. Em um trem se movimentando em linha reta as rodas percorrem a mesma distância. Ao fazer uma curva qual roda, em relação ao raio da curva, percorre o caminho maior, a roda interna ou a externa?
4. Cite alguns exemplos de movimento circular uniforme que temos no nosso dia a dia. Justifique.

9 INFORMAÇÕES PARA RESOLUÇÃO DOS DESAFIOS

Conversão de graus para radianos

$$1° = \frac{\pi rad}{180}$$

Portanto 1°/s é igual a $\frac{\pi}{180} rad$/s ou 0,0174533rad/s.

O perímetro da roda pode ser obtido através do seguinte cálculo:

$$P = \pi * D$$

Onde P é o perímetro da roda e D é o diâmetro da roda.
Essas conversões serão úteis para todos os desafios.

Utilização da Escala presente nos desafios

Exemplo:
Ao projetar a pista, o tamanho da escala, medido com uma trena, é de 30 cm. Como a escala possui 150 pixels, através de uma regra de três calcula-se o valor de cada pixel da pista:

$$\frac{150\ pixels}{1\ pixel} = \frac{30\ cm}{x}$$

Nesse caso 1pixel equivale a 5 cm. Esse valor mudará conforme se variar a distância de projeção.

Desafio 1 – Velocidade média

Para calcular a velocidade do robô deve ser medido as distâncias entre os obstáculos, ver o tempo necessário para chegar ao obstáculo e ficar os 5 segundos parado. De posse desses dados basta aplicá-los diretamente na expressão da velocidade média:

$$v = \frac{\Delta x}{\Delta t}$$

Após o cálculo converter a velocidade em graus por segundo para programar o robô

Desafio 2 – Encontro de móveis

Nesse desafio o aluno espera-se que o aluno mova um objeto de cada vez e dessa forma possa calcular a velocidade média de cada objeto.

Com esses dados os alunos podem calcular o instante em que os objetos irão se encontrar da seguinte forma

$$posição\ do\ objeto\ 1\colon\ S_1 = S_{01} + v_1.t$$

$$posição\ do\ objeto\ 2\colon\ S_2 = S_{02} - v_2.t$$

Igualando as posições dos objetos temos:

$$S_{01} + v_1.t = S_{02} - v_2.t$$

Isolando o tempo t, encontra-se o momento em que os objetos irão se encontrar:

$$t = \frac{s_{02} - s_{01}}{v_1 + v_2}$$

Para facilitar os cálculos, considerando a origem na posição inicial do objeto 1 temos:

$$t = \frac{s_{02}}{v_1 + v_2}$$

Sabendo o tempo em que os objetos irão se encontrar e substituindo esse tempo em uma das equações da posição, encontra-se o ponto em que haverá esse encontro.

Com o tempo calculado para o encontro e a distância da largada até os objetos, calcula-se a velocidade que o robô deverá avançar pela expressão:

$$v = \frac{\Delta x}{\Delta t}$$

O robô deverá ser posicionado na linha de largada de forma a encontrar os objetos na posição calculada para o encontro dos objetos.

Não esquecer de transformar a velocidade em graus por segundo para programar o robô corretamente.

Desafio 3 – Movimento Retilíneo Uniforme (MRU) e Movimento Retilíneo Uniformemente Variável (MRUV).

Nesse desafio o aluno deverá utilizar as equações do MRU e MRUV para calcular a velocidade e aceleração necessária para o robô acompanhar os objetos.

Para MRU:

$$x = x_0 + v_0.t$$

Para MRUV:

$$x = x_0 + v_0.t + \frac{1}{2}.at^2$$

Lembrando que a aceleração está em pixel/segundo ao quadrado e a velocidade dos objetos em pixels por segundo. Para as conversões utilizar a escala. A velocidade do robô deverá ser programada em graus por segundo.

Desafio 4 – Queda livre

Nesse desafio o aluno irá utilizar a equação do MRUV, com a aceleração da gravidade, para calcular o tempo que o objeto levará para tocar o solo.

Com esse tempo e com a medida da pista calcular a velocidade que o robô deverá assumir para encontrar esse objeto. Lembrando de converter para graus por segundo após os cálculos. A aceleração da gravidade da pista está em pixel por segundo ao quadrado e deverá ser convertida para centímetros ou metros para realizar o cálculo.

Desafio 5 – lançamento de projéteis

Nesse desafio utilizar as equações do movimento oblíquo para calcular a velocidade que o robô deverá se movimentar para encontrar o objeto na outra extremidade da pista:

Na horizontal:

$$v_{0x} = |\overrightarrow{v_0}|cos\theta$$

$$x = x_0 + |\overrightarrow{v_0}|cos\theta.t$$

Na vertical:

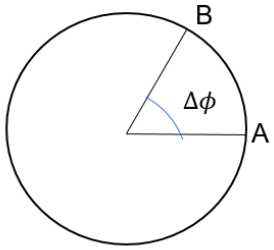
$$v_{0y} = |\overrightarrow{v_0}|sen\theta$$

$$y = y_0 + |\overrightarrow{v_0}|sen\theta . t - \frac{gt^2}{2}$$

Deverão ser realizadas as conversões de pixels para centímetro ou metros para efetuar os cálculos. A velocidade do robô também deverá ser convertida para graus por segundo.

Desafio 6 - Movimento Circular Uniforme

Utilizar as equações do movimento circular uniforme para converter a velocidade angular em velocidade escalar e programar o a velocidade do robô conforme os cálculos realizados:



$$\omega = \frac{\Delta\phi}{\Delta t}$$

$$v = \omega . R$$

Onde v é a velocidade escalar, ω é a velocidade angular e R o raio da circunferência, nesse caso o objeto que gira no centro da pista.

Cálculo da velocidade angular utilizando o período:

$$\omega = \frac{2\pi(rad)}{T}$$

Onde T é o intervalo de tempo do período. Assim como nos demais desafios a velocidade do robô deve ser programado em graus por segundo.

10 REFERÊNCIAS

ARDUINO IDE. Disponível em < https://www.arduino.cc/en/Main/Software>. Acesso em 01 de agosto de 2020

CONSTRUCT 3. Disponível em: < https://editor.construct.net/>. Acesso em 01 de agosto de 2020

DORCA, Ricardo Helou; BISCUOLA, Gualter José; BÔAS, Newton Vilas. Física 1: Mecânica. 2. ed. São Paulo: Saraiva, 2013.

OLIVEIRA, Mauricio Pietrocola Pinto de et al. Física em Contextos: Movimento, Força, Astronomia. São Paulo: Ftd, 2010.

RIBEIRO, Igor Souza. Et al. A plataforma Arduino: Princípios de Funcionamento e Demonstração Prática com um Controlador de Ventiladores. VII Congresso Brasileiro de Engenharia de Produção. Ponta Grossa, 2017.

ROBOCORE, Controlando Motores com o Módulo L298N. Disponível em: https://www.robocore.net/tutoriais/motor-dc-arduino-ponte-h-l298n. Acesso em 08 de agosto de 2020

SCIRRA. Disponível em: https://www.scirra.com/tutorials/262/comeando-com-o-construct-2-tutorial-in-portuguese>. Acesso em 01 de agosto de 2020