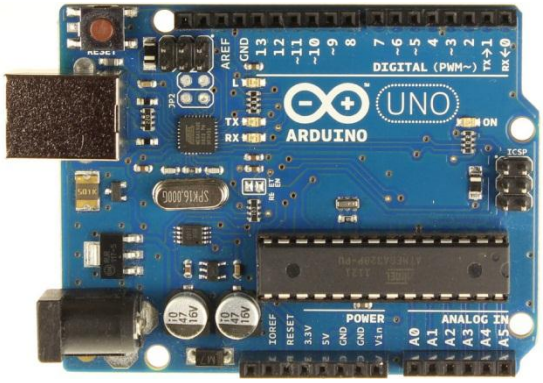
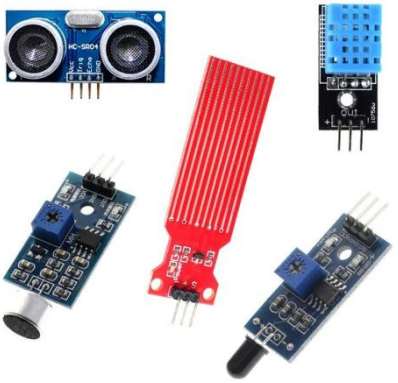
PPEFis Programa de Pós-Graduação em Ensino de Física - MNPEF

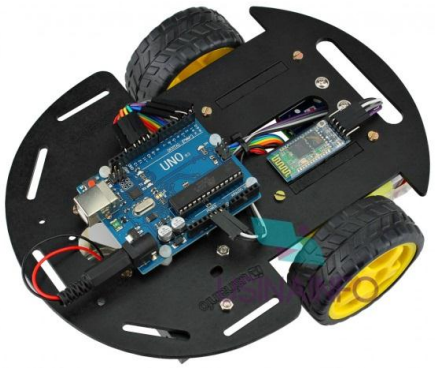
INSTITUTO FEDERAL
Espírito Santo

SBF
SOCIEDADE BRASILEIRA DE FÍSICA

# ROTEIRO DE ATIVIDADES NO ENSINO DA MECÂNICA UTILIZANDO ROBÓTICA EDUCACIONAL





**AUTOR:** EWERTON DE BARCELLOS JÚNIOR

**ORIENTADOR:** PROF. DR. CLEITON KENUP PIUMBINI

**COORIENTADOR:** PROF. ME. ROBSON LEONE EVANGELISTA

CARIACICA

2019

**SUMÁRIO**

CADERNO DO ALUNO

# FÍSICA MECÂNICA - CINEMÁTICA

## ATIVIDADE 1 – ESPAÇO PERCORRIDO, DESLOCAMENTO E POSIÇÃO

### OBJETIVO:

- Diferenciar espaço percorrido, deslocamento e posição com o objetivo de introduzir Física e seus objetos de estudo.

### MATERIAIS:

- Um robô utilizando a plataforma Arduíno, com a programação pré-determinada;
- Circuito impresso em lona.

### FUNDAMENTOS

- Espaço percorrido;
- Posição;
- Deslocamento;
- Cinemática.

### PROCEDIMENTO

- Esta atividade será realizada em quatro etapas.
- **Primeira etapa da atividade:** Libere o robô, já com a programação "Atividade _1.1", do bloco 0 do circuito.
- Observe em qual bloco o robô irá parar.
- Neste momento será feita, pelo mediador da atividade, uma indagação. Discuta com os outros visitantes e encontre uma possível resposta.
- **Segunda etapa da atividade:** Libere o robô, agora já com a programação "Atividade _1.2", do bloco 0 do circuito.
- Observe em qual bloco o robô irá parar.
- Neste momento será feita, pelo mediador da atividade, uma nova indagação. Discuta com os outros visitantes e encontre uma possível resposta.
- **Terceira etapa da atividade:** Libere o robô, agora já com a programação "Atividade _1.3", do bloco 0 do circuito.
- Observe o movimento do robô e em qual bloco ele irá parar.
- Neste momento será feita, pelo mediador da atividade, duas novas indagações. Discuta com os outros visitantes e encontre as possíveis respostas.
- **Quarta etapa da atividade:** Libere o robô, agora já com a programação "Atividade _1.4", do bloco 2 do circuito.
- Observe o movimento do robô e em qual bloco ele irá parar.
- Neste momento será feita, pelo mediador da atividade, três novas indagações. Discuta com os outros visitantes e encontre as possíveis respostas.

# FÍSICA MECÂNICA - CINEMÁTICA

## ATIVIDADE 2 – DETERMINAÇÃO DA VELOCIDADE

**OBJETIVO:**

- Relacionar os conceitos deslocamento e tempo com a grandeza velocidade escalar média;
- Coletar dados relativos a um movimento.

**MATERIAIS:**

- Um robô utilizando a plataforma Arduíno com a programação pré-determinada;
- Cronômetro (presente em celulares, relógios de pulso, etc.);
- Circuito impresso em lona.

**FUNDAMENTOS**

- Deslocamento;
- Tempo;
- Velocidade;
- Cinemática.

**PROCEDIMENTO**

- Esta atividade será realizada em uma etapa.
- Observe que o circuito apresenta 9 blocos divididos igualmente.
- Libere o robô, já com a programação “Atividade _2.1”, do bloco 0 do circuito.
- Utilizando um cronômetro, marque o tempo em que o robô se movimento do bloco 0 até o bloco em que o robô para.
- Observe em qual bloco o robô irá parar.
- Neste momento será proposta, pelo mediador, uma atividade. Realize essa atividade com os outros visitantes e encontre uma possível solução.

# FÍSICA MECÂNICA - CINEMÁTICA

## ATIVIDADE 3 – PONTO DE ENCONTRO

**OBJETIVO:**

- Relacionar movimento dos corpos com a velocidade média escalar com o objetivo de introduzir Física e seus objetos de estudo;
- Demonstrar que os corpos iguais se movimentam diferentes devido a grandeza velocidade escalar média.

**MATERIAIS:**

- Dois robôs utilizando a plataforma Arduíno, com a programação pré-determinada;
- Circuito impresso em lona.

**FUNDAMENTOS**

- Deslocamento;
- Tempo;
- Velocidade;
- Cinemática.

**PROCEDIMENTO**

- Esta atividade será realizada em duas etapas.
- **Primeira etapa da atividade:** Libere os robôs, já com as programações “Atividade _3.1” e “Atividade _3.2”, em cada um dos robôs, do bloco 0 do circuito.
- Observe os movimentos dos robôs e em qual bloco eles irão parar.
- Neste momento será feito, pelo mediador da atividade, uma indagação. Discuta com os outros visitantes e encontre uma possível resposta.
- **Segunda etapa da atividade:** Neste momento será proposto, pelo mediador da atividade, um desafio.
- Utilizando os robôs com as mesmas programações da etapa anterior, realize esse desafio com os outros visitantes e encontre uma possível solução.

# FÍSICA MECÂNICA - CINEMÁTICA

## ATIVIDADE 4 – ACELERAÇÃO DOS CORPOS

**OBJETIVO:**

- Identificar a variação da velocidade no decorrer do tempo e associar tal fenômeno a aceleração;
- Observar os movimentos dos corpos acelerados e desacelerados.

**MATERIAIS:**

- Dois robôs utilizando a plataforma Arduíno com a programação pré-determinada;
- Circuito impresso em lona.

**FUNDAMENTOS**

- Deslocamento;
- Tempo;
- Velocidade;
- Aceleração;
- Cinemática.

**PROCEDIMENTO**

- Esta atividade será realizada em uma etapa.
- Libere os robôs, já com as programações “Atividade _4.1” e “Atividade _4.2”, em cada um dos robôs, do bloco 0 do circuito.
- Observe os movimentos dos robôs e em qual bloco eles irão parar.
- Neste momento será feita, pelo mediador da atividade, três novas indagações. Discuta com os outros visitantes e encontre as possíveis respostas.

# FÍSICA MECÂNICA - DINÁMICA

## ATIVIDADE 5 – MOVIMENTO COM ATRITO

**OBJETIVO:**

- Observar diferentes movimentos dos corpos sobre diferentes superfícies de contato.
- Identificar que diferentes superfícies apresentam diferentes atritos com outros corpos.

**MATERIAIS:**

- Um robô utilizando a plataforma Arduíno com a programação pré-determinada;
- Uma toalha;
- Circuito impresso em lona.

**FUNDAMENTOS**

- Aceleração;
- Atrito;
- Dinâmica.

**PROCEDIMENTO**

- Esta atividade será realizada em duas etapas.
- **Primeira etapa da atividade:** Libere o robô, já com a programação "Atividade _5", do bloco 0 do circuito.
- Observe o movimento do robô sobre o circuito, que apresenta uma superfície com uma característica diferente (uma toalha de banho).
- **Segunda etapa da atividade:** Libere novamente o robô, já com a programação "Atividade _5" do bloco 0 do circuito.
- Observe novamente o movimento desse robô sobre o circuito, que apresenta uma superfície com uma característica diferente (uma toalha de banho).
- Neste momento será feito, pelo mediador da atividade, uma indagação. Discuta com os outros visitantes e encontre uma possível resposta.

# FÍSICA MECÂNICA - DINÁMICA

## ATIVIDADE 6 – CABO DE GUERRA

**OBJETIVO:**

- Identificar as condições de equilíbrio quando a resultante das forças é igual e diferente de zero.

**MATERIAIS:**

- Dois robôs utilizando a plataforma Arduíno, com a programação pré-determinada;
- Uma toalha;
- Um barbante;
- Circuito impresso em lona.

**FUNDAMENTOS**

- Força;
- Velocidade;
- Dinâmica.

**PROCEDIMENTO**

- Esta atividade será realizada em duas etapas.
- **Primeira etapa da atividade:** Libere os robôs, já com as programações “Atividade _6.1” e “Atividade _6.2”, em cada um dos robôs, no meio do circuito, em sentidos opostos.
- Observe o movimento do robô sobre o circuito.
- Neste momento será feito, pelo mediador da atividade, uma indagação. Discuta com os outros visitantes e encontre uma possível resposta.
- **Segunda etapa da atividade:** Libere novamente os robôs, já com as programações “Atividade _6.3” e “Atividade _6.4”, em cada um dos robôs, no meio do circuito, em sentidos opostos.
- Observe o movimento do robô sobre o circuito.
- Neste momento será feito, pelo mediador da atividade, uma indagação. Discuta com os outros visitantes e encontre uma possível resposta.

# CADERNO DO PROFESSOR

## FÍSICA MECÂNICA - CINEMÁTICA

## ATIVIDADE 1 – ESPAÇO PERCORRIDO, DESLOCAMENTO E POSIÇÃO

**OBJETIVO:**
O objetivo da atividade é compreender os conceitos posição, espaço percorrido e deslocamento.
Os objetivos específicos são:

- Diferenciar espaço percorrido, deslocamento e posição com o objetivo de introduzir Física e seus objetos de estudo.

**MATERIAIS:**

- Um robô utilizando a plataforma Arduíno com a programação pré-determinada;
- Circuito impresso em lona.

**FUNDAMENTO**

- Espaço percorrido;
- Posição;
- Deslocamento;
- Cinemática.

**PROCEDIMENTO**
No primeiro momento, baixe no robô a programação "Atividade _1.1". Os visitantes são orientados a liberar o robô do bloco zero da pista. Neste primeiro momento o robô apresenta uma programação em que ele anda do bloco zero até o bloco 8. Nesta hora é feita uma indagação: "Quais as posições inicial e final no circuito o robô se encontra?". Espera-se que os visitantes discutam entre si sobre o questionamento proposto e cheguem numa possível resposta. O mediador deve guiá-los na elaboração da resposta, sem fornecê-la diretamente.

Em um segundo momento, baixe no robô a programação "Atividade_1.2" e oriente os visitantes a liberará-lo novamente do bloco zero. O robô deve andar até um ponto qualquer do percurso, localizado antes do final do bloco 9. Neste momento faça uma segunda indagação: "Nesse segundo movimento, qual é o deslocamento realizado pelo robô?". Novamente, espera-se que os visitantes discutam e alcancem a possível resposta interagindo entre si. A função do mediador continua sendo de guiá-los na elaboração da resposta.

Em um terceiro momento, baixe no robô a programação “Atividade_1.3” e oriente os visitantes a liberar o robô do bloco zero. Com essa nova programação, o robô se desloca até o bloco 9 e retorna até certo ponto ao longo do percurso. Neste momento faça duas indagações: “Qual o deslocamento realizado pelo robô?” e “Qual o espaço percorrido realizado pelo robô?”. Mais uma vez, espera-se que os visitantes discutam e alcancem as possíveis respostas interagindo entre si. A função do mediador continua sendo de guiá-los na elaboração da resposta.

Em um quarto momento, baixe no robô a programação “Atividade_1.4” e oriente os visitantes a liberará-lo do bloco 2 do circuito. Com essa nova programação, o robô se desloca até o bloco 9, retorna até certo ponto ao longo do percurso. Neste momento, faça três indagações: “Quais as posições inicial e final no circuito o robô se encontra”, “Qual o deslocamento realizado pelo robô?” e “Qual o espaço percorrido realizado pelo robô?”. Neste momento é criado um debate, onde os visitantes devem compreender que o espaço percorrido depende da trajetória que o robô realiza e que o deslocamento depende apenas do ponto de partida e do ponto de chegada, não dependendo da trajetória.

# FÍSICA MECÂNICA - CINEMÁTICA

## ATIVIDADE 2 – DETERMINAÇÃO DA VELOCIDADE

**OBJETIVO:**

O objetivo dessa atividade é, através do trabalho em equipe, associar a velocidade ao tempo que um corpo gasta para percorrer certa distância.

Os objetivos específicos são:

- Coletar dados relativos a um movimento;
- Relacionar os conceitos deslocamento e tempo com a grandeza velocidade escalar média.

**MATERIAIS:**

- Um robô utilizando a plataforma Arduíno com a programação pré-determinada;
- Cronômetro (presente em celulares, relógios de pulso, etc.);
- Circuito impresso em lona.

**FUNDAMENTO**

- Deslocamento;
- Tempo;
- Velocidade;
- Cinemática.

**PROCEDIMENTO**

Inicialmente, baixe a programação “Atividade _2” no robô. Os visitantes são orientados a descobrir a velocidade do robô designados a eles. Inicialmente o robô é liberado do bloco 0. Informando sobre as divisórias em blocos presentes no circuito, faça os visitantes observarem o deslocamento (sendo os blocos do circuito) e o tempo (utilizando qualquer equipamento que tenha a função cronometro, como, por exemplo, celulares, relógios de pulso, etc.) do robô do bloco 0 até onde eles param.

Com as informações adquiridas, faça uma indagação: “Qual a relação que existe entre o deslocamento do robô, o tempo gasto por ele para esse deslocamento e a sua velocidade?”. Espera-se que os visitantes discutam entre si sobre o questionamento proposto e cheguem numa possível resposta. O mediador deve guiá-los na elaboração da resposta, sem fornecê-la diretamente. Após essa conclusão o mediador pode expor a equação de velocidade média escalar para os visitantes.

## FÍSICA MECÂNICA - CINEMÁTICA

## ATIVIDADE 3 – PONTO DE ENCONTRO

**OBJETIVO:**
O objetivo da atividade é compreender o conceito de velocidade escalar média.
Os objetivos específicos são:

- Relacionar movimento dos corpos com a velocidade média escalar com o objetivo de introduzir Física e seus objetos de estudo;
- Demonstrar que os corpos iguais se movimentam diferentes devido à grandeza velocidade escalar média.

**MATERIAIS:**

- Dois robôs utilizando a plataforma Arduíno, com a programação pré-determinada;
- Circuito impresso em lona.

**FUNDAMENTO**

- Deslocamento;
- Tempo;
- Velocidade;
- Cinemática.

**PROCEDIMENTO**

No primeiro momento, inicialmente, baixe a programação "Atividade _3.1" e "Atividade _3.2", cada programação em um robô diferente. Em seguida, faça a divisão dos visitantes em dois grupos e entregue um robô para cada grupo. Os dois grupos soltam os dois robôs do bloco 0, com o objetivo de chegarem ao bloco 9.

É observado que um dos robôs chega ao bloco 9 antes do outro. Neste momento, faça a seguinte indagação: "O que há de diferente de um robô para outro para que houvesse essa diferença de tempo para chegar ao bloco 9?" Neste momento é criado um debate, onde deve ser observado, pelos grupos, que os robôs apresentam algum parâmetro diferente (velocidades diferentes), onde o robô que tiver a maior velocidade fará o mesmo deslocamento, em menor tempo. A função do mediador é de auxiliar os grupos a alcançarem essa conclusão nessa etapa do processo.

No segundo momento, faça uma segunda indagação: "Sabendo que os robôs apresentam a mesma estrutura e que eles têm velocidades diferentes, o que pode ser feito para que eles passem pelo bloco 8 ao mesmo tempo, permanecendo com suas velocidades

constantes e sem alterar as suas estruturas?” Outra vez é criado um debate, onde deve ser observado, pelos grupos, que os robôs devem ser lançados em posições diferentes para que possam passar pelo bloco 8 no mesmo instante. Mais uma vez, o mediador tem a função de auxiliar os grupos a alcançarem essa conclusão. Após as possíveis respostas, o mediador propõe aos grupos que soltem os robôs em posições diferentes para que eles tenham uma visão das suas conclusões.

## FÍSICA MECÂNICA - CINEMÁTICA

## ATIVIDADE 4 – ACELERAÇÃO DOS CORPOS

**OBJETIVO:**
O objetivo da atividade é compreender o conceito de aceleração.
Os objetivos específicos são:

- Identificar a variação da velocidade no decorrer do tempo e associar tal fenômeno a aceleração;
- Observar os movimentos dos corpos acelerados e desacelerados.

**MATERIAIS:**

- Dois robôs utilizando a plataforma Arduíno, com a programação pré-determinada;
- Circuito impresso em lona.

**FUNDAMENTO**

- Deslocamento;
- Tempo;
- Velocidade;
- Aceleração;
- Cinemática.

**PROCEDIMENTO**

Inicialmente, baixe a programação “Atividade _4.1” e “Atividade _4.2”, cada programação em um robô diferente. Em seguida, faça a divisão dos visitantes em dois grupos e entregue um robô para cada grupo. Então, os grupos são orientados a soltarem os robôs do bloco 0.

É observado que os robôs se se movimentam cada vez mais rápido no decorrer do percurso. Neste momento, faça a seguinte indagação: “O que ocorre com o robô para que haja o aumento da sua velocidade?”. Logo em seguida, no momento de discursão entre os integrantes dos grupos, os visitantes devem observar que os robôs apresentam uma variação de sua velocidade por apresentarem aceleração. O mediador tem a função de auxiliar os grupos a alcançarem essa conclusão.

No final da atividade inicie uma discussão sobre esse fenômeno em veículos presentes no dia a dia dos visitantes.

## FÍSICA MECÂNICA - DINÁMICA

## ATIVIDADE 5 – MOVIMENTO COM ATRITO

**OBJETIVO:**
O objetivo da atividade é identificar o conceito de atrito e sua influência no movimento dos corpos.
Os objetivos específicos são:

- Observar diferentes movimentos dos corpos sobre diferentes superfícies de contato.
- Identificar que diferentes superfícies apresentam diferentes atritos com outros corpos.

**MATERIAIS:**

- Dois robôs utilizando a plataforma Arduíno com a programação pré-determinada;
- Uma toalha;
- Um par de calotas de plástico;
- Circuito impresso em lona.

**FUNDAMENTO**

- Aceleração;
- Atrito;
- Dinâmica.

**PROCEDIMENTO**
Inicialmente, modifique as rodas de um robô. Baixe a programação "Atividade _5" em apenas um robô. Em seguida, oriente os visitantes a soltarem o robô no circuito, partindo sempre do bloco 0 e se movendo até o bloco 9. Oriente os visitantes a observarem o movimento do robô no circuito.

Em seguida, oriente os visitantes a soltarem o mesmo robô sobre uma superfície com uma característica diferente da primeira (uma toalha de banho) e os oriente a observar o movimento do robô nessa nova superfície.

É possível afrirmar que, para cada superfície, o robô, se movimenta com velocidades diferentes. Neste momento, faça uma indagação: "O que influenciou na mudança do movimento do carrinho nas duas situações observadas?". Logo em seguida, no momento de discursão dos visitantes, deve ser observar que o carrinho apresenta uma variação de sua velocidade devido ao atrito entre suas rodas e a superfície que estão se movimentando. O mediador tem a função de auxiliar os visitantes a alcançarem essa conclusão.

No final da atividade faça uma discussão sobre esse fenômeno em veículos, com rodas diferentes e se movimentando em diferentes superfícies presentes no dia a dia dos visitantes.

## FÍSICA MECÂNICA - DINÁMICA

## ATIVIDADE 6 – CABO DE GUERRA

**OBJETIVO:**
O objetivo da atividade é compreender o conceito de equilíbrio dos corpos.
Os objetivos específicos são:

- Identificar as condições de equilíbrio quando a resultante das forças é igual e diferente de zero.

**MATERIAIS:**

- Dois robôs utilizando a plataforma Arduíno, com a programação pré-determinada;
- Uma toalha;
- Um barbante;
- Circuito impresso em lona.

**FUNDAMENTO**

- Força;
- Velocidade;
- Dinâmica.

**PROCEDIMENTO**

No primeiro momento, baixe nos robôs as programações “Atividade _6.1” e “Atividade _6.2”. Os visitantes são orientados a liberar os robôs que estão conectados por um barbante, ao mesmo tempo, em sentidos opostos, no meio do tapete. Neste primeiro momento os robôs, ao serem acionados, não conseguirão sair da posição em que foram liberados. Neste momento é feita uma indagação: “O que esses robôs apresentam de semelhante para que não saiam da posição?”. Espera-se que os visitantes discutam entre si sobre o questionamento proposto e cheguem numa possível resposta. O mediador deve guiá-los na elaboração da resposta, sem fornecê-la diretamente.

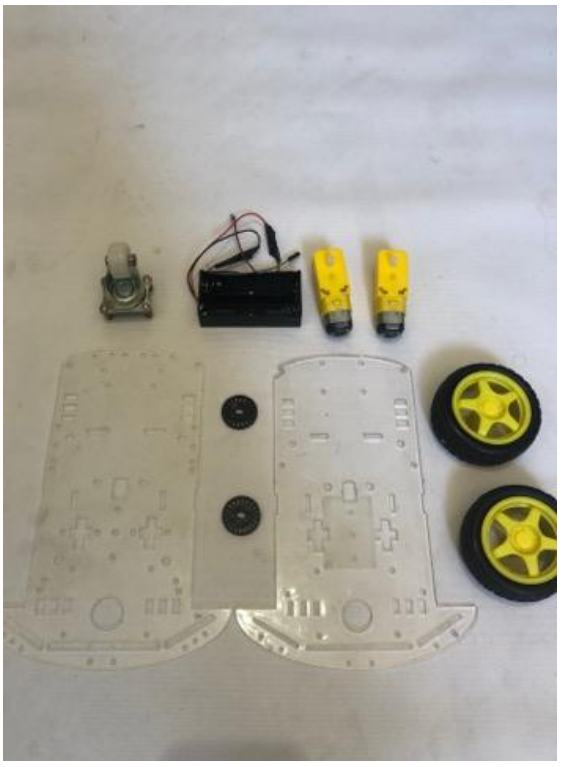
Em um segundo momento, baixe nos robôs as programações “Atividade_6.3” e a “Atividade _6.4” e oriente os visitantes a liberá-los novamente do centro da pista, em sentidos contrários. Um dos robôs irá puxar ou outro. Neste momento, faça uma segunda indagação: “Nessa segunda situação, o que levou um dos robôs sair do repouso e ser puxado pelo outro?”. Novamente, espera-se que os visitantes discutam e alcancem a possível resposta interagindo entre si. A função do mediador continua sendo de guiá-los na elaboração da resposta.

# CONSTRUÇÃO DO ROBÔ

O robô que é utilizado nas atividades é composto pelos seguintes componentes:

- Um kit Chassi para Robô duas rodas contendo dois chassi de acrílico, dois motores DC (3~6 V), duas rodas de borracha, uma roda articulada, dois discos de Encoder e suporte para duas baterias (Figura 1);
- Um microcontrolador Arduíno Uno R3 (Figura 2);
- Uma placa de ensaio mini (Figura 3);
- Dois sensores de velocidade (Figura 4);
- Um módulo ponte H L298N (Figura 5);
- Duas baterias de Lítio de 3.7 V (Figura 6).

Figura 1: Kit Chassi para Robô duas rodas.

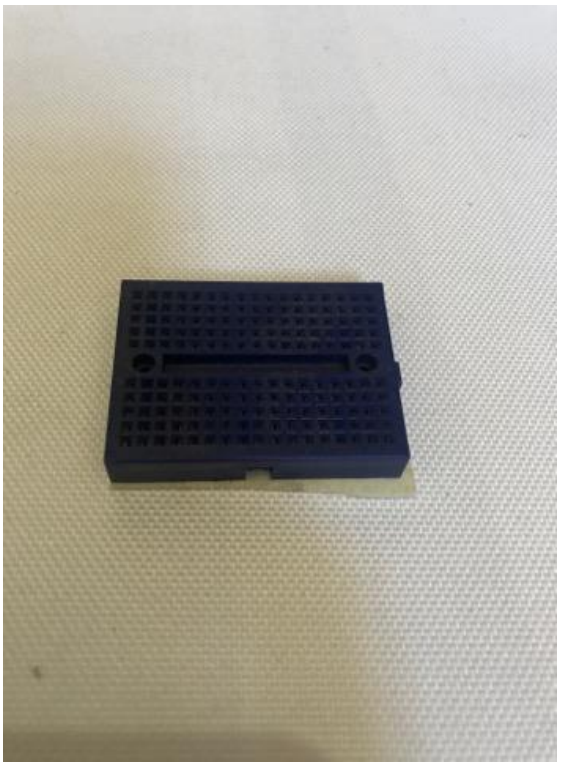


Fonte: Elaborado pelo autor.

Figura 2: Microcontrolador Arduíno R3.

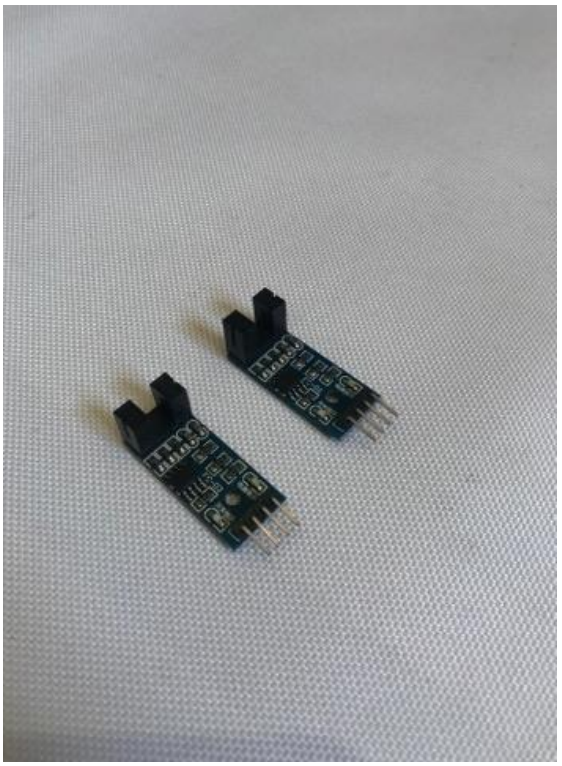

Fonte: Elaborado pelo autor.

Figura 3: Protoboard mini.



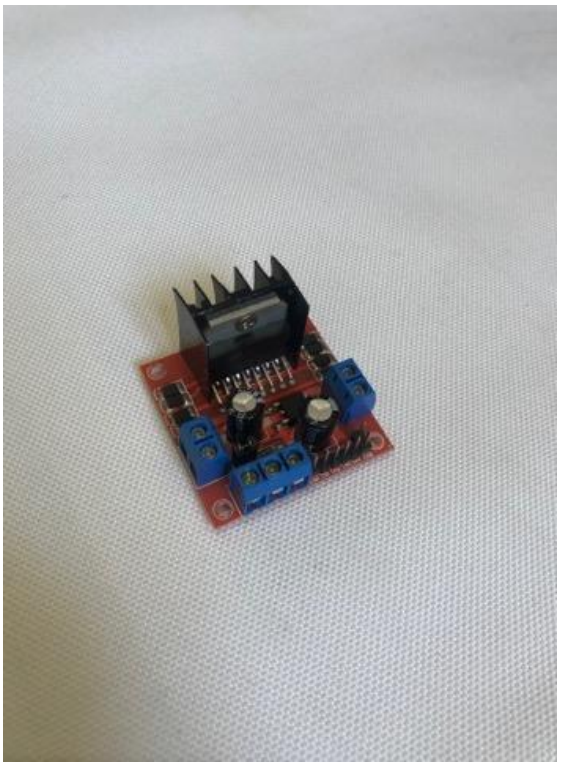
Fonte: Elaborado pelo autor.

Figura 4: Sensores de velocidade Encoder.



Fonte: Elaborado pelo autor.

Figura 5: Módulo ponte H L298N.



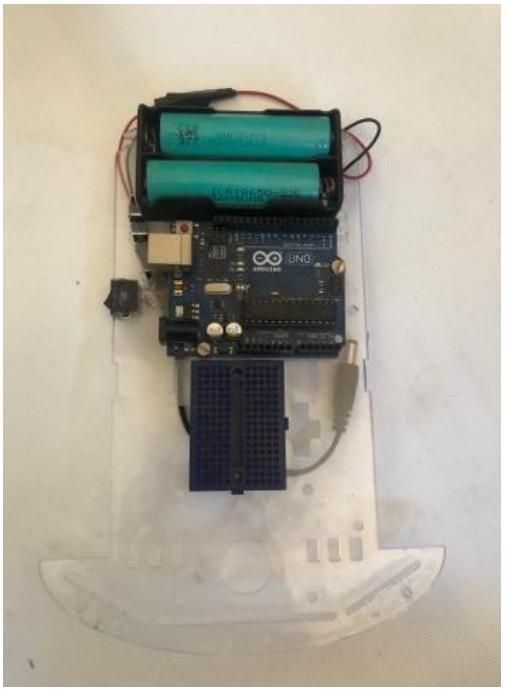
Fonte: Elaborado pelo autor.

Figura 6: baterias de Lítio de 3.7 V.

Fonte: Elaborado pelo autor.

Anexamos em uma das placas de acrílico presente no kit o microcontrolador Arduíno Uno R3, a placa de ensaio mini e o suporte duplo de bateria. Essa esquematização está demonstrada na Figura 7.

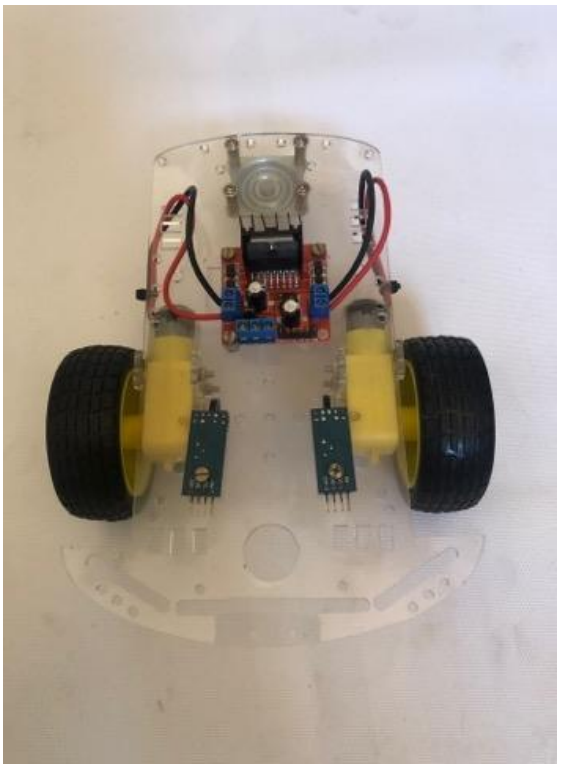
Figura 7: Chassi superior do robô.



Fonte: Elaborado pelo autor.

Em seguida anexamos na segunda placa de acrílico o módulo ponte H, os dois motores DC com as rodas emborrachadas e os dois discos de Encoder, os dois sensores de velocidade e a roda articulada. Essa esquematização está demonstrada na Figura 8.

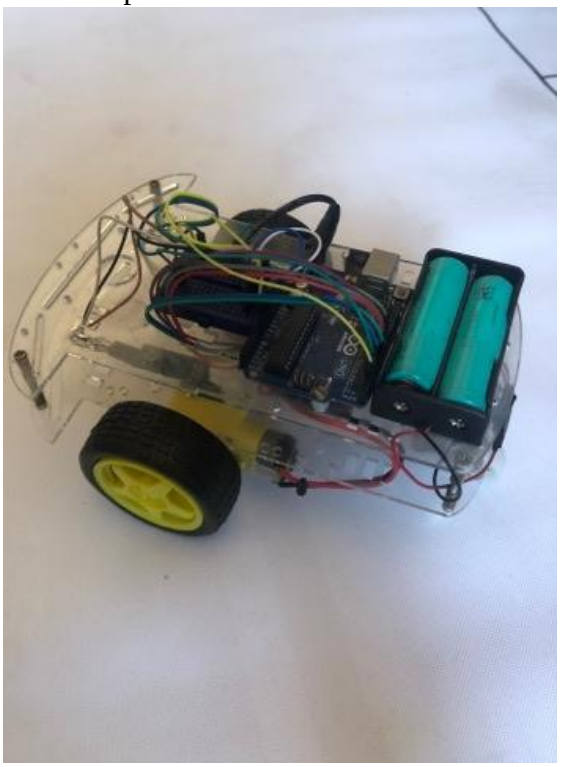
Figura 8: Chassi inferior do robô.



Fonte: Elaborado pelo autor.

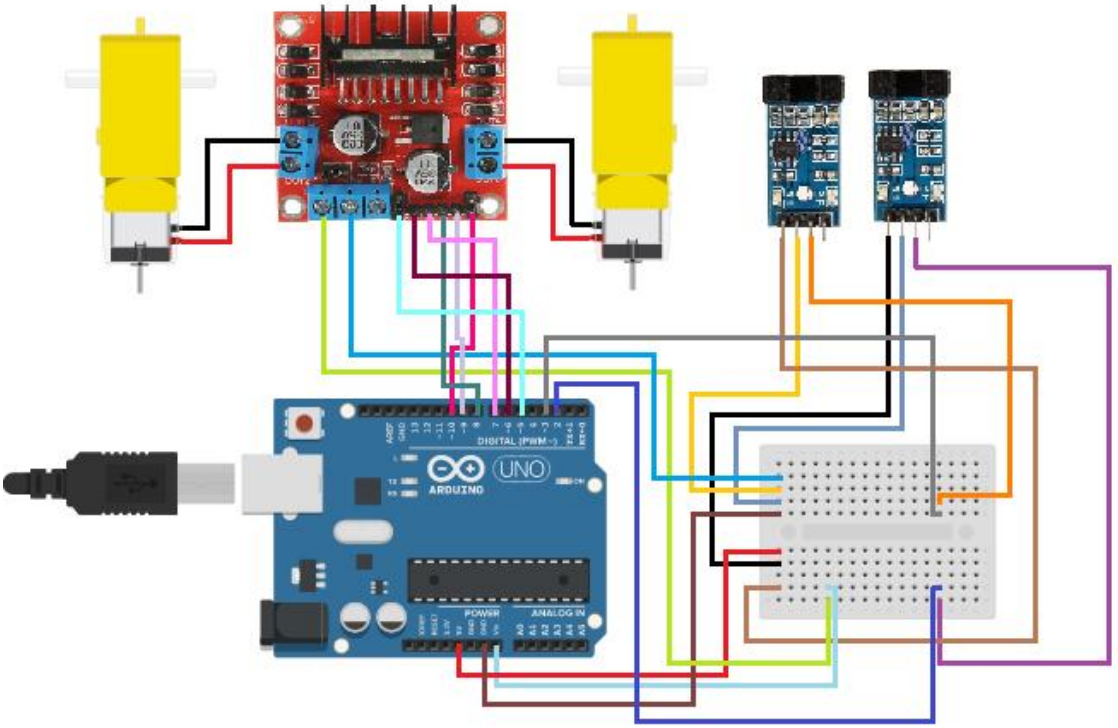
Logo após, unimos o chassi inferior com o superior e conectamos os fios (Fig. 9). Na Figura 10 é possível observar a esquematização de conexão dos componentes com o Arduíno

Figura 9: Robô com o chassi superior e inferior conectados.



Fonte: Elaborado pelo autor.

Figura 10: Esquematização de conexão dos componentes utilizada no robô.




Fonte: Elaborado pelo autor.

## PROGRAMAÇÕES DAS ATIVIDADES

A linguagem de programação utilizada nesse trabalho foi o C++, que é uma linguagem compilada e multi-paradigma. Para compilar o programa, ou seja, converter essa linguagem para a máquina é preciso utilizar um compilador. O compilador integrado que utilizamos foi o Arduíno IDE, disponível no site oficial do Arduíno (www.arduino.cc). As programações que escrevemos e compilamos nos robôs para a realização das atividades estão descritas a baixo.

**ATIVIDADE_1.1**

```
// Programa: Atividade 1 - 1
// Autor: Ewerton de Barcellos Junior

//                          // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção
```

```
// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {

  int result;  // Resultadi final do calculo
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda
em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno

}
// Função para avançar

void MoveForward(int steps, int mspeed)

{

   counter_A = 0;  // Reseta o contador A para zero

   counter_B = 0;  // Reseta o contador B para zero
// Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
```

```
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
      if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
    }
      // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

  // Defini o reverso do motor A
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o reverso do motor B
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);

  // Avança ate o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
```

```
    if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }

  // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
void SpinLeft(int steps, int mspeed)
{
   counter_A = 0;  //  reset counter A to zero
   counter_B = 0;  //  reset counter B to zero
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
      // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
  }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
```

```
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);  // Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);  // Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 1-1
    MoveForward(CMtoSteps(800), 255);  //  Andar para frente a uma distância de 0,7 mestros com velocidade de 190
  delay(50);  // Espera 50 milisegundos
  }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```

**ATIVIDADE_1.2**

```
// Programa: Atividade 1 - 2
// Autor: Ewerton de Barcellos Junior

//                              // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção

// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {
```

```
  int result;  // Resultadi final do calculo
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno
}
// Função para avançar
void MoveForward(int steps, int mspeed)
{
   counter_A = 0;  // Reseta o contador A para zero
   counter_B = 0;  // Reseta o contador B para zero
// Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
```

```
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
    }
      // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
```

```
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

  // Defini o reverso do motor A
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o reverso do motor B
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);

  // Avança ate o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {

    if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }

  // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
void SpinLeft(int steps, int mspeed)
{
```

```
   counter_A = 0;  //  reset counter A to zero
   counter_B = 0;  //  reset counter B to zero
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
      // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
  }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);       //
Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);       //
Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 1-2
```

```
  MoveForward(CMtoSteps(500), 255);  // Andar para frente a uma distância de 0,5 mestros com velocidade de 255
  delay(50);  // Espera 50 milisegundos
  }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```

**ATIVIDADE_1.3**

```
// Programa: Atividade 1 - 3
// Autor: Ewerton de Barcellos Junior

//                              // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção

// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {
```

```
  int result;  // Resultadi final do calculo
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno
}
// Função para avançar
void MoveForward(int steps, int mspeed)
{
   counter_A = 0;  // Reseta o contador A para zero
   counter_B = 0;  // Reseta o contador B para zero
// Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
```

```
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
    }
      // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
```

```
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero

 // Defini o reverso do motor A
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
// Defini o reverso do motor B
 digitalWrite(in3, HIGH);
 digitalWrite(in4, LOW);
    // Avança ate o valor do passo seja atingido
  while (steps > counter_A && steps > counter_B) {

   if (steps > counter_A) {
   analogWrite(enA, mspeed);
   } else {
   analogWrite(enA, 0);
   }
   if (steps > counter_B) {
   analogWrite(enB, mspeed);
   } else {
   analogWrite(enB, 0);
   }
  }
    // Para quando terminar
 analogWrite(enA, 0);
 analogWrite(enB, 0);
 counter_A = 0;  //  reset counter A to zero
 counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
void SpinLeft(int steps, int mspeed)
{
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
```

```
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);

  digitalWrite(in4, HIGH);
      // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
  }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);      //
Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);      //
Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 1-3
MoveForward(CMtoSteps(700), 255);  // Andar para frente a uma distância de 0,7
```

```
mestros com velocidade de 255
  delay(50);  // Espera 50 milisegundos
   MoveReverse(CMtoSteps(400), 255);
  delay(1000);  // Espera 1000 milisegundos
  }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```

**ATIVIDADE_1.4**

```
// Programa: Atividade 1 - 4
// Autor: Ewerton de Barcellos Junior

//                              // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção

// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {
```

```
  int result;  // Resultadi final do calculo
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno
}
// Função para avançar
void MoveForward(int steps, int mspeed)
{
   counter_A = 0;  // Reseta o contador A para zero
   counter_B = 0;  // Reseta o contador B para zero
// Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
```

```
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
    }
      // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
```

```
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero

 // Defini o reverso do motor A
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
// Defini o reverso do motor B
 digitalWrite(in3, HIGH);
 digitalWrite(in4, LOW);
    // Avança ate o valor do passo seja atingido
  while (steps > counter_A && steps > counter_B) {

   if (steps > counter_A) {
   analogWrite(enA, mspeed);
   } else {
   analogWrite(enA, 0);
   }
   if (steps > counter_B) {
   analogWrite(enB, mspeed);
   } else {
   analogWrite(enB, 0);
   }
  }
    // Para quando terminar
 analogWrite(enA, 0);
 analogWrite(enB, 0);
 counter_A = 0;  //  reset counter A to zero
 counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
void SpinLeft(int steps, int mspeed)
{
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
```

```
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
      // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
  }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);  // Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);  // Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 1-4
MoveForward(CMtoSteps(700), 255);    // Andar para frente a uma distância de 0,7 mestros com velocidade de 255
```

```
  delay(50);  // Espera 50 milisegundos
   MoveReverse(CMtoSteps(200), 255);   // Andar para tras a uma distância de 0,2 mestros com velocidade de 255
  delay(1000);  // Espera 1000 milisegundos
  }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```

**ATIVIDADE_2**

```
// Programa: Atividade 2
// Autor: Ewerton de Barcellos Junior

//                              // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção

// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {
```

```
  int result;  // Resultadi final do calculo
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno
}
// Função para avançar
void MoveForward(int steps, int mspeed)
{
   counter_A = 0;  // Reseta o contador A para zero
   counter_B = 0;  // Reseta o contador B para zero
// Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
```

```
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
    }
      // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
```

```
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero

 // Defini o reverso do motor A
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
// Defini o reverso do motor B
 digitalWrite(in3, HIGH);
 digitalWrite(in4, LOW);
    // Avança ate o valor do passo seja atingido
  while (steps > counter_A && steps > counter_B) {

   if (steps > counter_A) {
   analogWrite(enA, mspeed);
   } else {
   analogWrite(enA, 0);
   }
   if (steps > counter_B) {
   analogWrite(enB, mspeed);
   } else {
   analogWrite(enB, 0);
   }
  }
    // Para quando terminar
 analogWrite(enA, 0);
 analogWrite(enB, 0);
 counter_A = 0;  //  reset counter A to zero
 counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
void SpinLeft(int steps, int mspeed)
{
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
```

```
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);
      // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
  }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);      //
Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);      //
Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 2
MoveForward(CMtoSteps(700), 255);  //  Andar para frente a uma distância de 0,7
mestros com velocidade de 255
```

```
  delay(50);  // Espera 50 milisegundos
    }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```

**ATIVIDADE_3.1**

```
// Programa: Atividade 3.1
// Autor: Ewerton de Barcellos Junior

//                              // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção

// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {
```

```
  int result;  // Resultadi final do calculo
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno
}
// Função para avançar
void MoveForward(int steps, int mspeed)
{
   counter_A = 0;  // Reseta o contador A para zero
   counter_B = 0;  // Reseta o contador B para zero
// Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
```

```
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
    }
      // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
```

```
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero

 // Defini o reverso do motor A
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
// Defini o reverso do motor B
 digitalWrite(in3, HIGH);
 digitalWrite(in4, LOW);
    // Avança ate o valor do passo seja atingido
  while (steps > counter_A && steps > counter_B) {

   if (steps > counter_A) {
   analogWrite(enA, mspeed);
   } else {
   analogWrite(enA, 0);
   }
   if (steps > counter_B) {
   analogWrite(enB, mspeed);
   } else {
   analogWrite(enB, 0);
   }
  }
    // Para quando terminar
 analogWrite(enA, 0);
 analogWrite(enB, 0);
 counter_A = 0;  //  reset counter A to zero
 counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
void SpinLeft(int steps, int mspeed)
{
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
```

```
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
     // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
      if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
  }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);  //
Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);  //
Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 3-1
MoveForward(CMtoSteps(850), 140);  //  Andar para frente a uma distância de 0,85
mestros com velocidade de 140
```

```
  delay(50);  // Espera 50 milisegundos
    }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```

**ATIVIDADE_3.2**

```
// Programa: Atividade 3.2
// Autor: Ewerton de Barcellos Junior

//                              // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção

// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {
```

```
  int result;  // Resultadi final do calculo
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda
em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno
}
// Função para avançar
void MoveForward(int steps, int mspeed)
{
   counter_A = 0;  // Reseta o contador A para zero
   counter_B = 0;  // Reseta o contador B para zero
// Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
   while (steps > counter_A && steps > counter_B) {
      if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
```

```
      if (steps > counter_A) {
   analogWrite(enA, mspeed);
   } else {
   analogWrite(enA, 0);
   }
   if (steps > counter_B) {
   analogWrite(enB, mspeed);
   } else {
   analogWrite(enB, 0);
   }
   }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

  // Defini o reverso do motor A
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o reverso do motor B
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Avança ate o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {

   if (steps > counter_A) {
   analogWrite(enA, mspeed);
   } else {
   analogWrite(enA, 0);
   }
   if (steps > counter_B) {
   analogWrite(enB, mspeed);
   } else {
   analogWrite(enB, 0);
   }
  }
    // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
```

```
void SpinLeft(int steps, int mspeed)
{
   counter_A = 0;  //  reset counter A to zero
   counter_B = 0;  //  reset counter B to zero
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
      // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
  }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);       //
Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);       //
Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 3-2
MoveForward(CMtoSteps(850), 255);  //  Andar para frente a uma distância de 0,85
mestros com velocidade de 255
  delay(50);  // Espera 50 milisegundos
     }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```

```
// Programa: Atividade 4.1
// Autor: Ewerton de Barcellos Junior

//                                  // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção

// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {

  int result;  // Resultadi final do calculo
```

```
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno
}
// Função para avançar
void MoveForward(int steps, int mspeed)
{
   counter_A = 0;  // Reseta o contador A para zero
   counter_B = 0;  // Reseta o contador B para zero
// Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
   while (steps > counter_A && steps > counter_B) {
      if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
      if (steps > counter_A) {
```

```
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
    }
      // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

  // Defini o reverso do motor A
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o reverso do motor B
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
     // Avança ate o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {

    if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
void SpinLeft(int steps, int mspeed)
```

```
{
   counter_A = 0;  //  reset counter A to zero
   counter_B = 0;  //  reset counter B to zero
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
      // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
  }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);       //
Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);       //
Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 4-1
MoveForward(CMtoSteps(100), 140);   //   Andar para frente a uma distância de 0,1
mestros com velocidade de 140
  delay(1);  // Espera 1 milisegundos
  MoveForward(CMtoSteps(100), 160);   //   Andar para frente a uma distância de 0,1
mestros com velocidade de 160
  delay(1);  // Espera 1 milisegundos
MoveForward(CMtoSteps(100), 180);   //   Andar para frente a uma distância de 0,1
mestros com velocidade de 180
  delay(1);  // Espera 1 milisegundos
MoveForward(CMtoSteps(100), 200);   //   Andar para frente a uma distância de 0,1
mestros com velocidade de 200
  delay(1);  // Espera 1 milisegundos
  MoveForward(CMtoSteps(100), 220);   //   Andar para frente a uma distância de 0,1
mestros com velocidade de 220
  delay(1);  // Espera 1 milisegundos
```

```
  MoveForward(CMtoSteps(100), 240);  //  Andar para frente a uma distância de 0,1 mestros com velocidade de 240
  delay(1);  // Espera 1 milisegundos
  MoveForward(CMtoSteps(300), 255);  //  Andar para frente a uma distância de 0,1 mestros com velocidade de 255
  delay(1);  // Espera 1 milisegundos
  }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```

**ATIVIDADE_4.2**

```
// Programa: Atividade 4.2
// Autor: Ewerton de Barcellos Junior

//                              // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção

// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {
```

```
  int result;  // Resultadi final do calculo
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda
em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno
}
// Função para avançar
void MoveForward(int steps, int mspeed)
{
   counter_A = 0;  // Reseta o contador A para zero
   counter_B = 0;  // Reseta o contador B para zero
   // Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
```

```
      if (steps > counter_A) {
   analogWrite(enA, mspeed);
   } else {
   analogWrite(enA, 0);
   }
   if (steps > counter_B) {
   analogWrite(enB, mspeed);
   } else {
   analogWrite(enB, 0);
   }
   }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

  // Defini o reverso do motor A
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o reverso do motor B
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Avança ate o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {

   if (steps > counter_A) {
   analogWrite(enA, mspeed);
   } else {
   analogWrite(enA, 0);
   }
   if (steps > counter_B) {
   analogWrite(enB, mspeed);
   } else {
   analogWrite(enB, 0);
   }
  }
    // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
```

```
void SpinLeft(int steps, int mspeed)
{
   counter_A = 0;  //  reset counter A to zero
   counter_B = 0;  //  reset counter B to zero
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
      // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
  }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);       //
Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);       //
Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 4-2
MoveForward(CMtoSteps(850), 200);  //  Andar para frente a uma distância de 0,85
mestros com velocidade de 200
  delay(50);  // Espera 50 milisegundos
     }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```

**ATIVIDADE_5**

```
// Programa: Atividade 5
// Autor: Ewerton de Barcellos Junior

//                              // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção

// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {
```

```
  int result;  // Resultadi final do calculo
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno
}
// Função para avançar
void MoveForward(int steps, int mspeed)
{
   counter_A = 0;  // Reseta o contador A para zero
   counter_B = 0;  // Reseta o contador B para zero
// Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
```

```
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
    }
      // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
```

```
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero

 // Defini o reverso do motor A
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
// Defini o reverso do motor B
 digitalWrite(in3, HIGH);
 digitalWrite(in4, LOW);
    // Avança ate o valor do passo seja atingido
  while (steps > counter_A && steps > counter_B) {

   if (steps > counter_A) {
   analogWrite(enA, mspeed);
   } else {
   analogWrite(enA, 0);
   }
   if (steps > counter_B) {
   analogWrite(enB, mspeed);
   } else {
   analogWrite(enB, 0);
   }
  }
    // Para quando terminar
 analogWrite(enA, 0);
 analogWrite(enB, 0);
 counter_A = 0;  //  reset counter A to zero
 counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
void SpinLeft(int steps, int mspeed)
{
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
```

```
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);
      // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
  }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);      //
Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);      //
Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 5
MoveForward(CMtoSteps(900), 255);  //  Andar para frente a uma distância de 0,9
mestros com velocidade de 255
```

```
  delay(50);  // Espera 50 milisegundos
     }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```

**ATIVIDADE_6.1**

```
// Programa: Atividade 6.1
// Autor: Ewerton de Barcellos Junior

//                              // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção

// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {
```

```
  int result;  // Resultadi final do calculo
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno
}
// Função para avançar
void MoveForward(int steps, int mspeed)
{
   counter_A = 0;  // Reseta o contador A para zero
   counter_B = 0;  // Reseta o contador B para zero
// Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
```

```
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
    }
      // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
```

```
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero

 // Defini o reverso do motor A
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
// Defini o reverso do motor B
 digitalWrite(in3, HIGH);
 digitalWrite(in4, LOW);
    // Avança ate o valor do passo seja atingido
  while (steps > counter_A && steps > counter_B) {

   if (steps > counter_A) {
   analogWrite(enA, mspeed);
   } else {
   analogWrite(enA, 0);
   }
   if (steps > counter_B) {
   analogWrite(enB, mspeed);
   } else {
   analogWrite(enB, 0);
   }
  }
    // Para quando terminar
 analogWrite(enA, 0);
 analogWrite(enB, 0);
 counter_A = 0;  //  reset counter A to zero
 counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
void SpinLeft(int steps, int mspeed)
{
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
```

```
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
     // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
      if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
  }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);      //
Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);      //
Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 6-1
MoveForward(CMtoSteps(900), 255);  //  Andar para frente a uma distância de 0,9
mestros com velocidade de 255
```

```
  delay(50);  // Espera 50 milisegundos
     }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```

**ATIVIDADE_6.2**

```
// Programa: Atividade 6.2
// Autor: Ewerton de Barcellos Junior

//                              // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção
```

```
// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {

  int result;  // Resultadi final do calculo
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno
}
// Função para avançar
void MoveForward(int steps, int mspeed)
{
   counter_A = 0;  // Reseta o contador A para zero
   counter_B = 0;  // Reseta o contador B para zero
// Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
```

```
   while (steps > counter_A && steps > counter_B) {
      if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
      if (steps > counter_A) {
    analogWrite(enA, mspeed);
```

```
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
    }
      // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

  // Defini o reverso do motor A
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o reverso do motor B
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
     // Avança ate o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {

    if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
```

```
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
void SpinLeft(int steps, int mspeed)
{
   counter_A = 0;  //  reset counter A to zero
   counter_B = 0;  //  reset counter B to zero
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
      // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
```

```
    analogWrite(enB, 0);
    }
  }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);  // Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);  // Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 6-2
MoveForward(CMtoSteps(900), 255);  //  Andar para frente a uma distância de 0,9 mestros com velocidade de 255
  delay(50);  // Espera 50 milisegundos
     }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```

**ATIVIDADE_6.3**

```
// Programa: Atividade 6.3
// Autor: Ewerton de Barcellos Junior

//                                  // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção
```

```
// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {

  int result;  // Resultadi final do calculo
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno
}
// Função para avançar
void MoveForward(int steps, int mspeed)
{
   counter_A = 0;  // Reseta o contador A para zero
   counter_B = 0;  // Reseta o contador B para zero
// Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
   while (steps > counter_A && steps > counter_B) {
```

```
      if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
      if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
```

```
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
    }
      // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

  // Defini o reverso do motor A
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o reverso do motor B
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
     // Avança ate o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {

    if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
```

```
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
void SpinLeft(int steps, int mspeed)
{
   counter_A = 0;  //  reset counter A to zero
   counter_B = 0;  //  reset counter B to zero
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);
digitalWrite(in4, HIGH);
      // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
```

```
   }
  }
    // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);  // Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);  // Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 6-3
MoveForward(CMtoSteps(900), 255);  //  Andar para frente a uma distância de 0,9 mestros com velocidade de 255
  delay(50);  // Espera 50 milisegundos
    }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```

**ATIVIDADE_6.4**

```
// Programa: Atividade 6.4
// Autor: Ewerton de Barcellos Junior

//                              // --- Mapeamento de Hardware ---
const byte MOTOR_A = 3;  // Sensor de velocidade do Motor 2 - INT 1 - Motor da direita
const byte MOTOR_B = 2;  // Sensor de velocidade do Motor 1 - INT 0 - Motor da esquerda

// Constante para as voltas do disco
const float stepcount = 20.00;  // 20 Slots no disco

// Constante para o diametro da roda
const float wheeldiameter = 66.10; // O diametro da roda em milimetros

// Inteiros para contadores de pulsos
volatile int counter_A = 0;
volatile int counter_B = 0;

// Motor A

int enA = 10;
int in1 = 9;
int in2 = 8;

// Motor B

int enB = 5;
int in3 = 7;
int in4 = 6;

// Rotinas de serviço de interrupção
```

```
// Contagem de impulsos do motor A ISR
void ISR_countA()
{
  counter_A++;  // incremento do valor do contador do motor A
}

// Contagem de impulsos do motor B ISR
void ISR_countB()
{
  counter_B++;  // incremento do valor do contador do motor B
}

// Função para converter de cemtimetros para passos
int CMtoSteps(float cm) {

  int result;  // Resultadi final do calculo
  float circumference = (wheeldiameter * 3.14) / 10; // Calculo da circuferencia da roda em cm
  float cm_step = circumference / stepcount;  // CM por passo

  float f_result = cm / cm_step;  // Resultado do calculo como float
  result = (int) f_result; // Converte para um inteiro
    return result;  // Resultado final e retorno
}
// Função para avançar
void MoveForward(int steps, int mspeed)
{
   counter_A = 0;  // Reseta o contador A para zero
   counter_B = 0;  // Reseta o contador B para zero
   // Defini o motor B para frente
   digitalWrite(in3, LOW);
   digitalWrite(in4, HIGH);
      // Avança ate o valor do passo ser atingido
   while (steps > counter_A && steps > counter_B) {
```

```
      if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando termina
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função para se mover no sentido inverso
void MoveReverse(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

   // Defini o motor A para tras
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o motor B para tras
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
    // Retorna ate o valor de passo ser atingido
   while (steps > counter_A && steps > counter_B) {
      if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
```

```
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
    }
      // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
// Função de girar para a direita
void SpinRight(int steps, int mspeed)
{
   counter_A = 0;  //  Reseta o contador A para zero
   counter_B = 0;  //  Reseta o contador B para zero

  // Defini o reverso do motor A
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
// Defini o reverso do motor B
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
     // Avança ate o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {

    if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
```

```
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
    }
   }
     // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  reset counter A to zero
  counter_B = 0;  //  reset counter B to zero
}
// Função de virar para a direita
void SpinLeft(int steps, int mspeed)
{
   counter_A = 0;  //  reset counter A to zero
   counter_B = 0;  //  reset counter B to zero
// Defini o motor A para frente
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Defini o motor B para tras
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
      // Andar até que o valor do passo seja atingido
   while (steps > counter_A && steps > counter_B) {
       if (steps > counter_A) {
    analogWrite(enA, mspeed);
    } else {
    analogWrite(enA, 0);
    }
    if (steps > counter_B) {
    analogWrite(enB, mspeed);
    } else {
    analogWrite(enB, 0);
```

```
   }
  }
    // Para quando terminar
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  counter_A = 0;  //  Reseta o contador A para zero
  counter_B = 0;  //  Reseta o contador B para zero
}
void setup()
{
  attachInterrupt(digitalPinToInterrupt (MOTOR_A), ISR_countA, RISING);  // Aumento do contador A quando o pino do sensor de velocidade estiver alto
  attachInterrupt(digitalPinToInterrupt (MOTOR_B), ISR_countB, RISING);  // Aumento do contador B quando o pino do sensor de velocidade estiver alto
    // Realizando o atividade 6-4
MoveForward(CMtoSteps(900), 190);  //  Andar para frente a uma distância de 0,9 mestros com velocidade de 190
  delay(50);  // Espera 50 milisegundos
    }
void loop()
{
  // Espaço destinado a fazer a repetição do que quiser
}
```