Robótica Educacional

Módulo 3

Aula 26

PalmBot - I

Diretoria de Tecnologia e Inovação

PARANÁ
GOVERNO DO ESTADO
SECRETARIA DA EDUCAÇÃO

GOVERNADOR DO ESTADO DO PARANÁ
Carlos Massa Ratinho Júnior

SECRETÁRIO DE ESTADO DA EDUCAÇÃO
Roni Miranda Vieira

DIRETOR DE TECNOLOGIA E INOVAÇÃO
Claudio Aparecido de Oliveira

COORDENADOR DE TECNOLOGIAS EDUCACIONAIS
Marcelo Gasparin

Produção de Conteúdo
Cleiton Rosa
Darice Alessandra Deckmann Zanardini

Validação de Conteúdo
Cleiton Rosa

Revisão Textual
Kellen Pricila dos Santos Cochinski

Projeto Gráfico e Diagramação
Edna do Rocio Becker

2024

Sumário

Aula 26 PalmaBot - I

Introdução

A Robótica é um campo vasto e nos estimula a pensar em projetos novos ou mesmo criar outros com base no que já aprendemos, ressignificando componentes e ações!

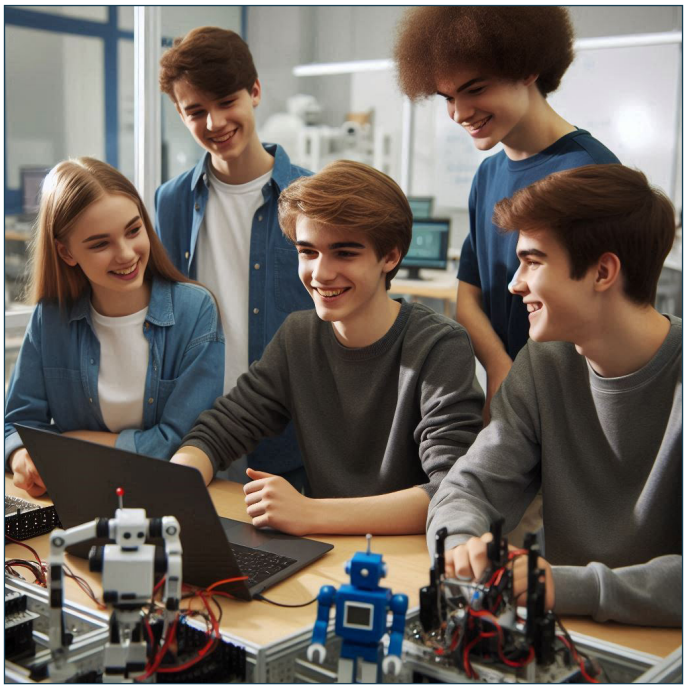
Que tal você se aventurar conosco em mais um desafio e criar um robô com interação por palmas?

Objetivos desta aula

- Iniciar o desenvolvimento do robô com interação por palmas;
- Programar o robô com interação por palmas;
- Explorar utilizações da função millis();
- Programar movimentos de avanço, esquerda, ré e direita conforme detecção de palmas.

Lista de materiais

- Notebook ou computador;
- 1 Arduino Uno.



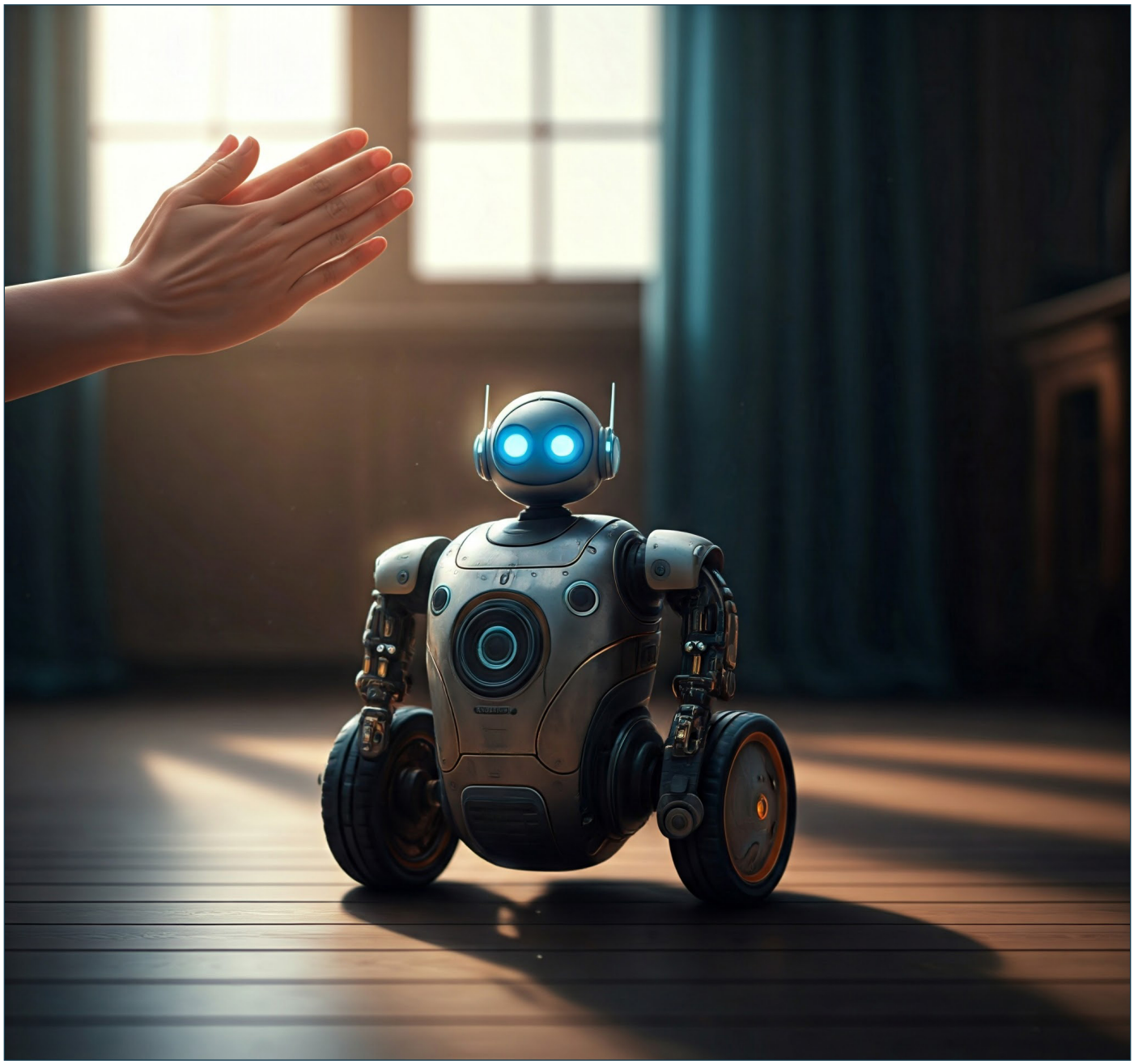
Aula 26 PalmaBot - I

Roteiro da aula

1. Contextualização

A Robótica, além de criar soluções para desafios do mundo real, pode ser também uma ferramenta para a inovação e aprimoramento de projetos, permitindo novas formas de aplicação e controles.

Figura 1 - Representação de um robô com interação por palmas



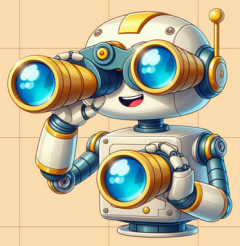
Fonte: Imagem gerada por IA.

Aula 26 PalmaBot - I

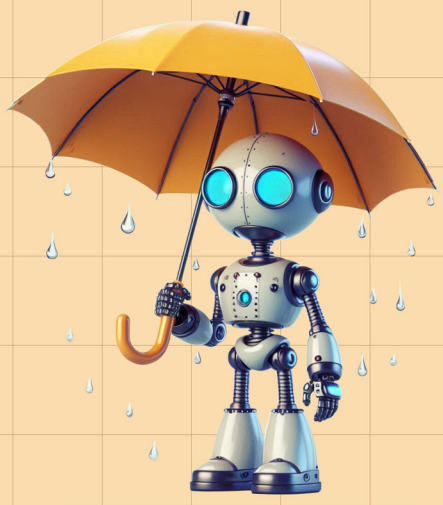
Que tal construir um robô capaz de tomar decisões simples pela interação com o ambiente?

Robôs podem realizar tarefas e tomar decisões de forma independente, sem necessidade de controle humano constante, via periféricos, para as ações em tempo real. Eles conseguem essa autonomia através da integração de sensores, atuadores e processamento de dados e a interação com o ambiente é fundamental para suas operações. É pela interação que o robô percebe seu redor e toma decisões sobre como se mover e agir.

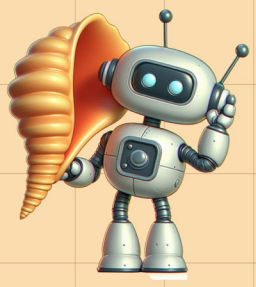
Nos Módulos 1 e 2 de Robótica Educacional, desenvolvemos os robôs controlados por rádio e wireless, também desenvolvemos os robôs sumô e seguidor de linha, compreendendo a importância dos sensores ultrassônico e infravermelho para a autonomia desses robôs e interação com o ambiente. Os sensores, portanto, funcionam como os "órgãos dos sentidos" do robô na coleta de informação sobre o ambiente e há outros tipos que podem ser usados em projetos de Robótica e desenvolver os "sentidos" do robô:

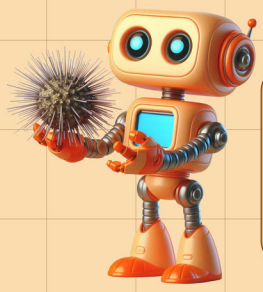


Visão: câmeras e sensores LiDAR (Light Detection and Ranging) permitem que o robô "veja" o mundo ao seu redor, identificando objetos, cores e distâncias.



Audição: microfones captam sons, permitindo que o robô ouça comandos de voz ou detecte eventos sonoros.





Tato: sensores táteis permitem que o robô perceba o contato com objetos, como ao pegar um objeto ou evitar colisões.

Outras sensações: sensores de temperatura, umidade, pressão e outros podem ser utilizados para coletar informações adicionais sobre o ambiente.

Aula 26 PalmaBot - I

Iniciaremos nesta aula, o projeto de um robô focado no sentido da audição, capaz de responder a comandos simples através de palmas. Programaremos o sensor de som para detectar palmas e, conforme detecção, controlar os motores do robô, o que possibilitará seus movimentos de avanço, giro, retorno e parada.

Além de ser um projeto divertido, poderemos refletir sobre as etapas de programação, montagem e possíveis aplicações de um robô que interage por palmas, pensando também sobre o que define os conceitos de robô controlado e robô autônomo e como a interação de robôs com o ambiente é um campo dinâmico!

2. Programação

A programação do robô que interage por palmas contará com funções essenciais para seu funcionamento:

- Funções de controle;
- Funções de movimento.

Vamos começar? No sketch da programação do robô que interage por palmas, crie uma aba com a extensão **.h** para o preâmbulo.

Figura 2 - Inserção da aba **preambulo.h**

Fonte: Arduino IDE.

Iniciaremos a programação do nosso robô definindo na aba **preambulo.h** os pinos da ponte H e do sensor do som e criando as variáveis para controle de palmas.

Aula 26 PalmaBot - I

```
/* Definindo os pinos da ponte H L298N. */
#define pinIN1 5   /* Pino IN1 da ponte H para o motor esquerdo (anti-horário). */
#define pinIN2 6   /* Pino IN2 da ponte H para o motor esquerdo (horário). */
#define pinIN3 9   /* Pino IN3 da ponte H para o motor direito (horário). */
#define pinIN4 10  /* Pino IN4 da ponte H para o motor direito (anti-horário). */

/* Definindo o pino do sensor de som. */
#define pinSensorSom 2  /* Pino digital ao qual o sensor de som está conectado. */

/* Variáveis para controle de palmas. */
int contadorPalmas = 0;  /* Contador para o número de palmas detectadas. */
unsigned long tempoInicioEscuta = 0;  /* Armazena o tempo em que o modo de escuta é iniciado. */
unsigned long ultimaDeteccaoSom = 0;  /* Armazena o tempo da última detecção de som. */
const unsigned long tempoEscuta = 1000;    /* Duração total de escuta após a primeira palma (1 segundo). */
const unsigned long debounceTempo = 250;   /* Tempo mínimo entre detecções para evitar contagem duplicada. */
bool escutandoPalmas = false;  /* Indica se o robô está no modo de escuta. */
bool estadoAnteriorSom = LOW;  /* Armazena o estado anterior do sensor de som. */
```

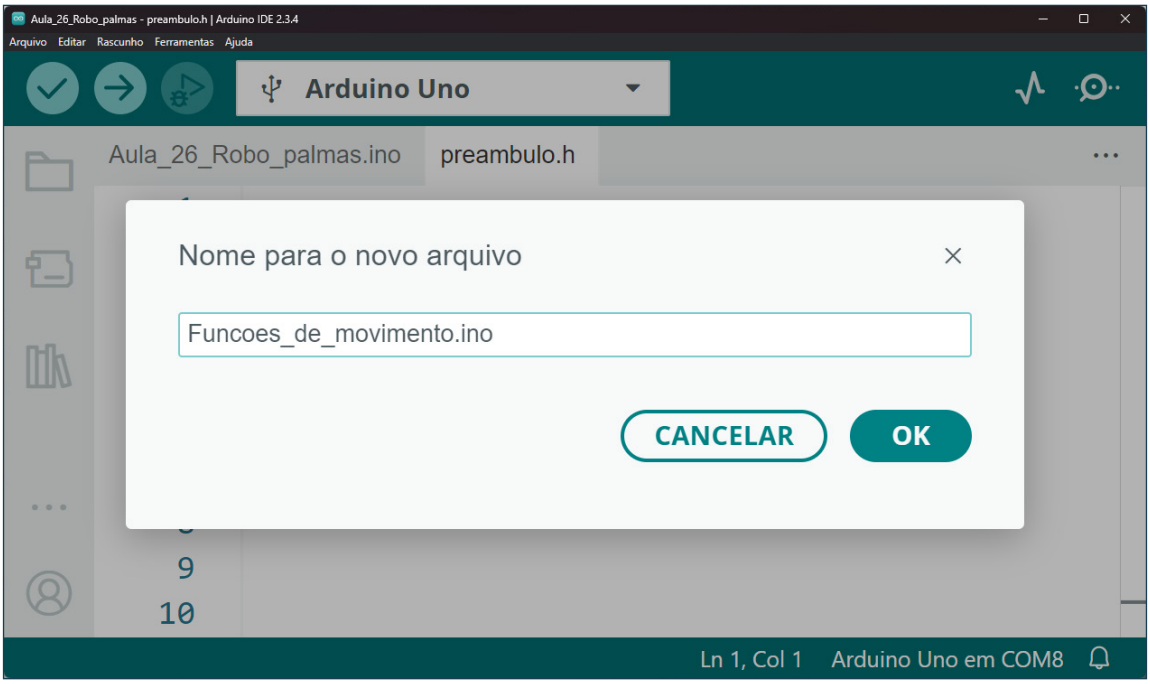
Aula 26 PalmaBot - I

Com essa estrutura inicial, o código monitorará continuamente as entradas do sensor de som. Ao detectar uma palma, iniciará o modo de escuta para contar o número de palmas no intervalo de tempo definido. Dependendo do número de palmas, diferentes comandos poderão ser enviados aos motores, permitindo o controle do robô. Observe que nossa programação traz tipos de variáveis para o controle de palmas:

- `int`, para armazenar os números inteiros, como a contagem de palmas detectadas.
- `unsigned long`, para armazenar números inteiros não negativos, como os tempos armazenados em milissegundos.
- `const unsigned long`, similar ao **unsigned long**, mas o valor é constante e não muda após ser inicializado, definindo tempos fixos para o período de escuta e debounce.
- `bool`, para armazenar valores booleanos **true** ou **false**, indicando se o robô está no modo de escuta e o estado do sensor de som.

Criaremos a sequência das funções de controle e das funções de movimento em duas abas respectivas – vamos à primeira, **Funcoes_de_movimento.ino**, para definir toda a movimentação dos motores que conectaremos ao nosso projeto e porque quando criarmos as funções de controle por palmas, os movimentos do robô serão requisitados.

Figura 3 - Inserção da aba **Funcoes_de_movimento.ino**



Fonte: Arduino IDE.

Aula 26 PalmaBot - I

A ideia é que nosso robô execute os movimentos padrões - frente, ré, direita e esquerda – e pare. Para isso, criaremos nessa aba cinco funções nas quais atribuiremos potência **0** (mínima) ou **255** (máxima) aos motores conforme as conexões com a ponte H.

```
/* Funções de movimento do robô: */

void moverFrente() {
  analogWrite(pinIN1, 255);   /* Motor esquerdo sentido anti-horário. */
  analogWrite(pinIN2, 0);     /* Motor esquerdo sentido horário (parado). */
  analogWrite(pinIN3, 255);   /* Motor direito sentido horário. */
  analogWrite(pinIN4, 0);     /* Motor direito sentido anti-horário (parado). */
  Serial.println("Movimento: Frente");
}

void moverRe() {
  analogWrite(pinIN1, 0);     /* Motor esquerdo sentido anti-horário (parado). */
  analogWrite(pinIN2, 255);   /* Motor esquerdo sentido horário. */
  analogWrite(pinIN3, 0);     /* Motor direito sentido horário (parado). */
  analogWrite(pinIN4, 255);   /* Motor direito sentido anti-horário. */
  Serial.println("Movimento: Ré");
}
```

```
void moverDireita() {
  analogWrite(pinIN1, 255);   /* Motor esquerdo sentido anti-ho-
rário. */
  analogWrite(pinIN2, 0);     /* Motor esquerdo sentido horário
(parado). */
  analogWrite(pinIN3, 0);     /* Motor direito sentido horário
(parado). */
  analogWrite(pinIN4, 255);   /* Motor direito sentido anti-horá-
rio. */
  Serial.println("Movimento: Direita");
}

void moverEsquerda() {
  analogWrite(pinIN1, 0);     /* Motor esquerdo sentido anti-ho-
rário. (parado) */
  analogWrite(pinIN2, 255);   /* Motor esquerdo sentido horário.
*/
  analogWrite(pinIN3, 255);   /* Motor direito sentido horário.
*/
  analogWrite(pinIN4, 0);     /* Motor direito sentido anti-horá-
rio (parado). */
  Serial.println("Movimento: Esquerda");
}

void pararRobo() {
  analogWrite(pinIN1, 0);     /* Para o motor esquerdo. */
  analogWrite(pinIN2, 0);     /* Para o motor esquerdo. */
  analogWrite(pinIN3, 0);     /* Para o motor direito. */
  analogWrite(pinIN4, 0);     /* Para o motor direito. */
  Serial.println("Movimento: Parado");
}
```

Aula 26 PalmaBot - I

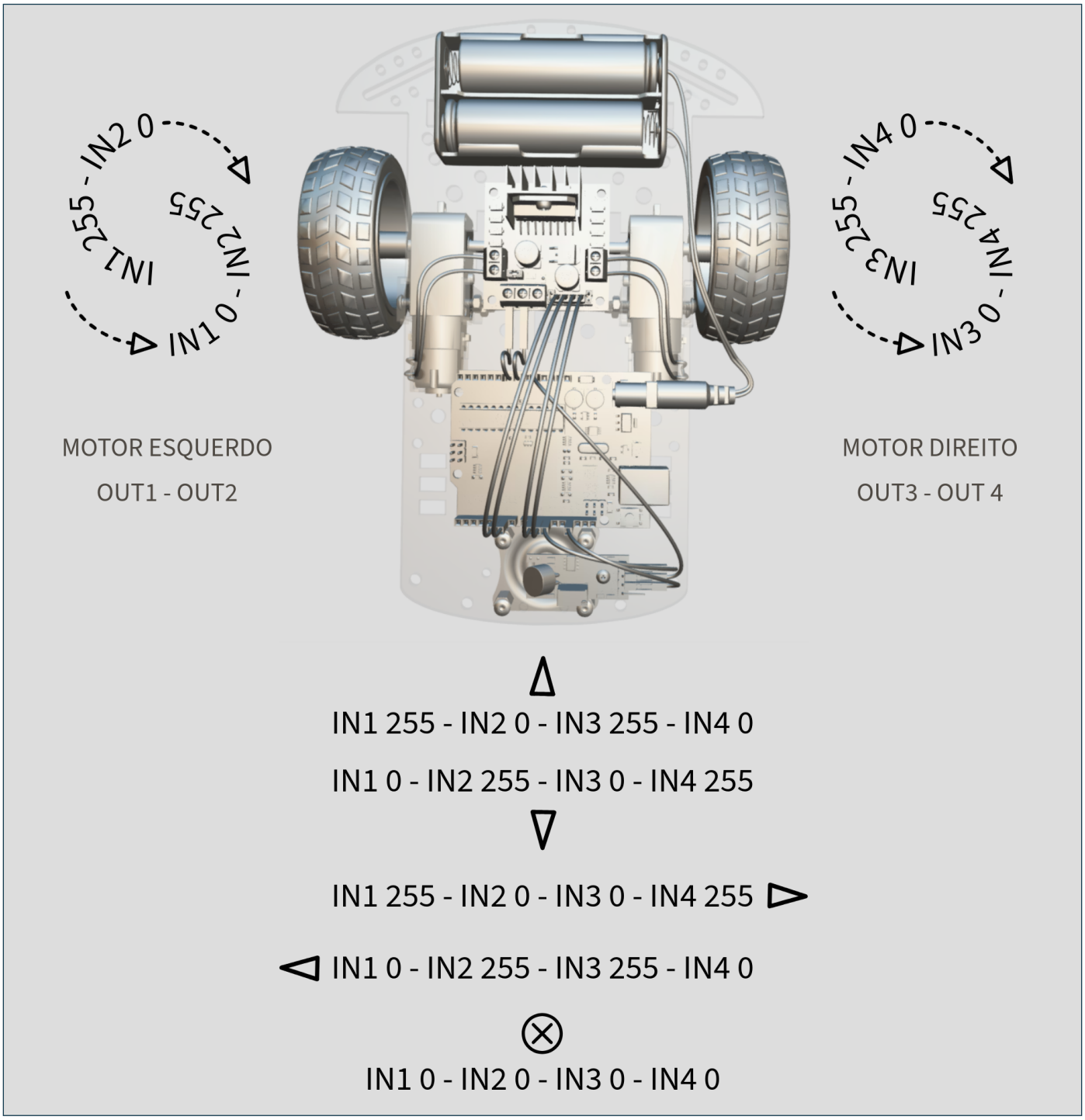
As cinco funções presentes na aba controlam a direção dos motores, alterando a potência e polaridade aplicada aos pinos da ponte H, permitindo o movimento em várias direções, além da parada do robô, quando todos os pinos estão com potência 0.

- `moverFrente()`
  - Pelas funções **analogWrite(pinIN1, 255)** e **analogWrite(pinIN3, 255)**, os pinos **pinIN1** e **pinIN3** são alimentados ao máximo, fazendo com que ambos os motores que estarão conectados girem para frente. Pelas funções **analogWrite(pinIN2, 0)** e **analogWrite(pinIN4, 0)**, os demais pinos ficam com potência zerada, garantindo que os motores não girem no sentido contrário.
- `moverRe()`
  - Aplicando a potência **255** aos pinos **IN2** e **IN4**, enquanto os pinos **IN1** e **IN3** estão com potência **0**, há inversão do sentido dos motores, fazendo-os girar para trás.
- `moverDireita()`
  - A potência **255** aplicada ao pino **IN1** faz o motor esquerdo girar no sentido anti-horário e aplicada ao pino **IN4** faz o motor direito girar no sentido anti-horário, criando efeito de rotação para a direita.
- `moverEsquerda()`
  - A potência **255** aplicada ao pino **IN2** faz o motor esquerdo girar no sentido horário e aplicada ao pino **IN3** faz o motor direito girar no sentido horário, criando efeito de rotação para a esquerda.
- `pararRobo()`
  - A potência **0** aplicada a **IN1**, **IN2**, **IN3** e **IN4** desliga todos os pinos, parando os motores.

Como você pode observar, a aba **Funcoes_de_movimento.ino** traz a sequência de funções do tipo "**void**" para controlar a direção dos motores, alterando a polaridade aplicada aos pinos da **ponte H**, o que permite o movimento em várias direções e a parada do robô. Os pinos **IN1**, **IN2**, **IN3** e **IN4** são utilizados para controlar a direção e o movimento dos motores DC, pois a **ponte H** permite a inversão da polaridade aplicada ao motor, o que possibilita que ele gire em diferentes sentidos (horário e anti-horário).

Aula 26 PalmaBot - I

Infográfico rotação e movimentos

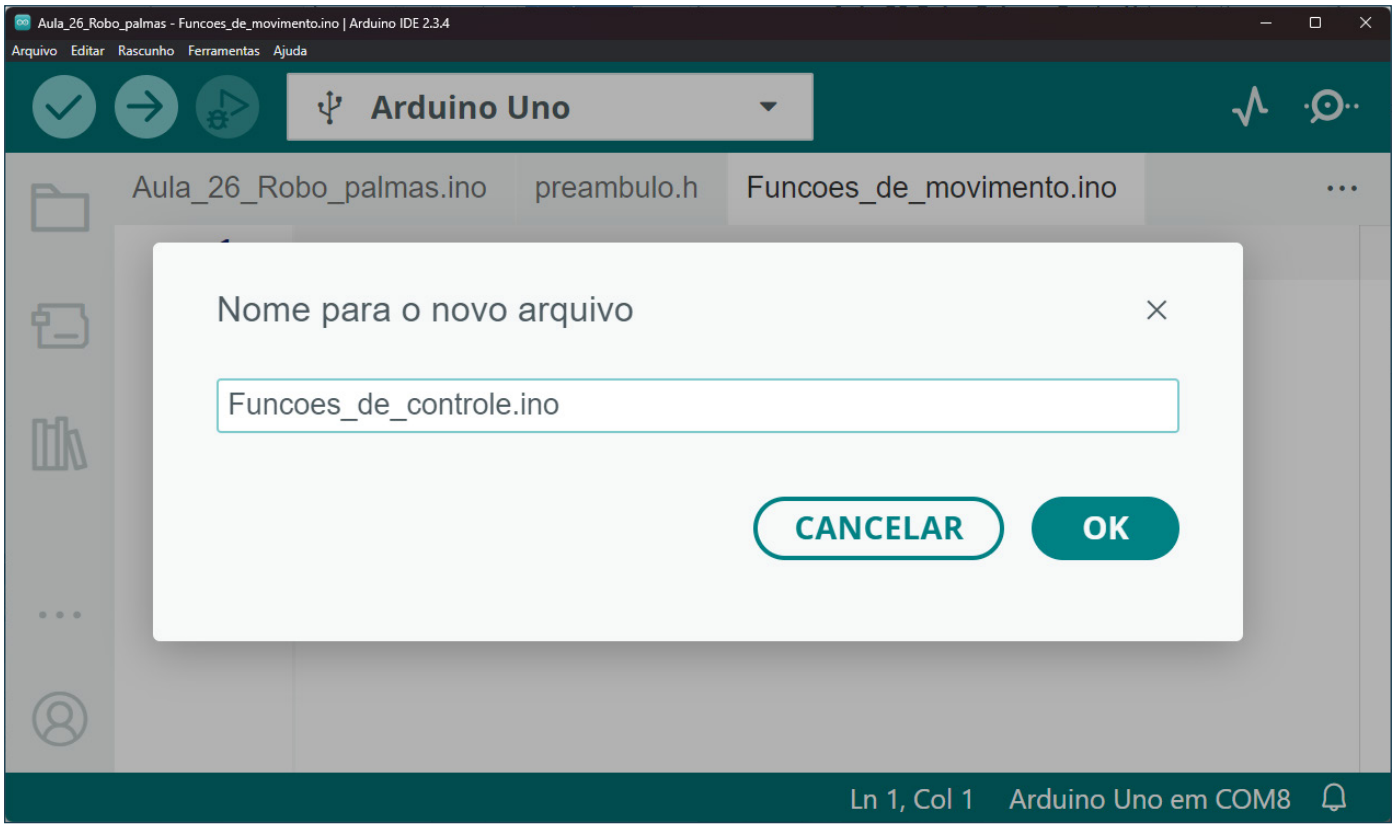


Outro recurso aplicado às funções de movimento é o de imprimir, via monitor serial do Arduino IDE, qual movimento está sendo executado ou se o robô está parado. Esse recurso da impressão será útil quando fizermos os primeiros testes de escuta do robô que interage pelas palmas!

Aula 26 PalmaBot - I

Vamos agora para a próxima aba da programação do robô, destinada às funções de controle do robô que interage por palmas.

Figura 4 - Inserção da aba **Funcoes_de_controle.ino**



Fonte: Arduino IDE.

As funções que adicionaremos à aba trabalham juntas para permitir que o robô responda à interação pelas palmas de maneira eficiente, contando palmas, temporizando as detecções e executando movimentos com base na contagem de palmas.

Aula 26 PalmaBot - I

```
/* Função para iniciar o modo de escuta ao detectar a primeira palma. */
void iniciarEscuta() {
  contadorPalmas = 1; /* Define o contador de palmas para 1 após detectar a primeira palma. */
  escutandoPalmas = true; /* Ativa o modo de escuta. */
  tempoInicioEscuta = millis(); /* Armazena o tempo atual de início do modo de escuta. */
  ultimaDeteccaoSom = millis(); /* Atualiza o tempo da última detecção de som. */
  Serial.println("Primeira palma detectada, iniciando contagem...");
}

/* Função para verificar e contar palmas enquanto escuta, com controle de temporização para debounce. */
void contarPalmas() {
  int leituraSom = digitalRead(pinSensorSom); /* Lê o estado do sensor de som. */
  /* Detecta transição de LOW para HIGH com controle de debounce. */
  if (leituraSom == HIGH && estadoAnteriorSom == LOW) {
    unsigned long tempoAtual = millis();                    /* Armazena o tempo atual. */
    if (tempoAtual - ultimaDeteccaoSom >= debounceTempo) { /* Verifica se passou o tempo mínimo entre detecções. */
      contadorPalmas++;                                     /* Incrementa o contador de palmas. */
```

```
      Serial.print("Contador de Palmas: ");
      Serial.println(contadorPalmas);
      ultimaDeteccaoSom = tempoAtual; /* Atualiza o tempo da última detecção de som. */
    }
  }
  estadoAnteriorSom = leituraSom; /* Atualiza o estado anterior para o próximo loop. */
}

/* Função para executar ações com base na contagem de palmas após o tempo de escuta. */
void executarMovimento() {
  if (contadorPalmas == 1) {
    moverFrente();
    delay(500);
  } else if (contadorPalmas == 2) {
    moverRe();
    delay(500);
  } else if (contadorPalmas == 3) {
    moverDireita();
    delay(200);
  } else if (contadorPalmas == 4) {
    moverEsquerda();
    delay(200);
  }

  pararRobo(); /* Chama a função para parar o robô após o movimento. */
  contadorPalmas = 0; /* Reseta o contador de palmas. */
  escutandoPalmas = false; /* Sai do modo de escuta. */
}
```

Aula 26 PalmaBot - I

Vamos entender a proposta geral de cada função?

- `iniciarEscuta()`
  - Essa função é chamada quando a primeira palma é detectada. Ela inicia o modo de escuta e prepara as variáveis necessárias para contar as palmas subsequentes e enviar uma mensagem para o monitor serial, indicando que a contagem de palmas começou.
- `contarPalmas()`
  - Verifica continuamente o estado do sensor de som e conta o número de palmas detectadas, utilizando uma temporização para evitar detecções duplicadas (debounce).
- `executarMovimento()`
  - Executa ações específicas com base no número de palmas detectadas após o período de escuta, chamando as funções que criamos na aba **Funcoes_de_movimento.ino**.

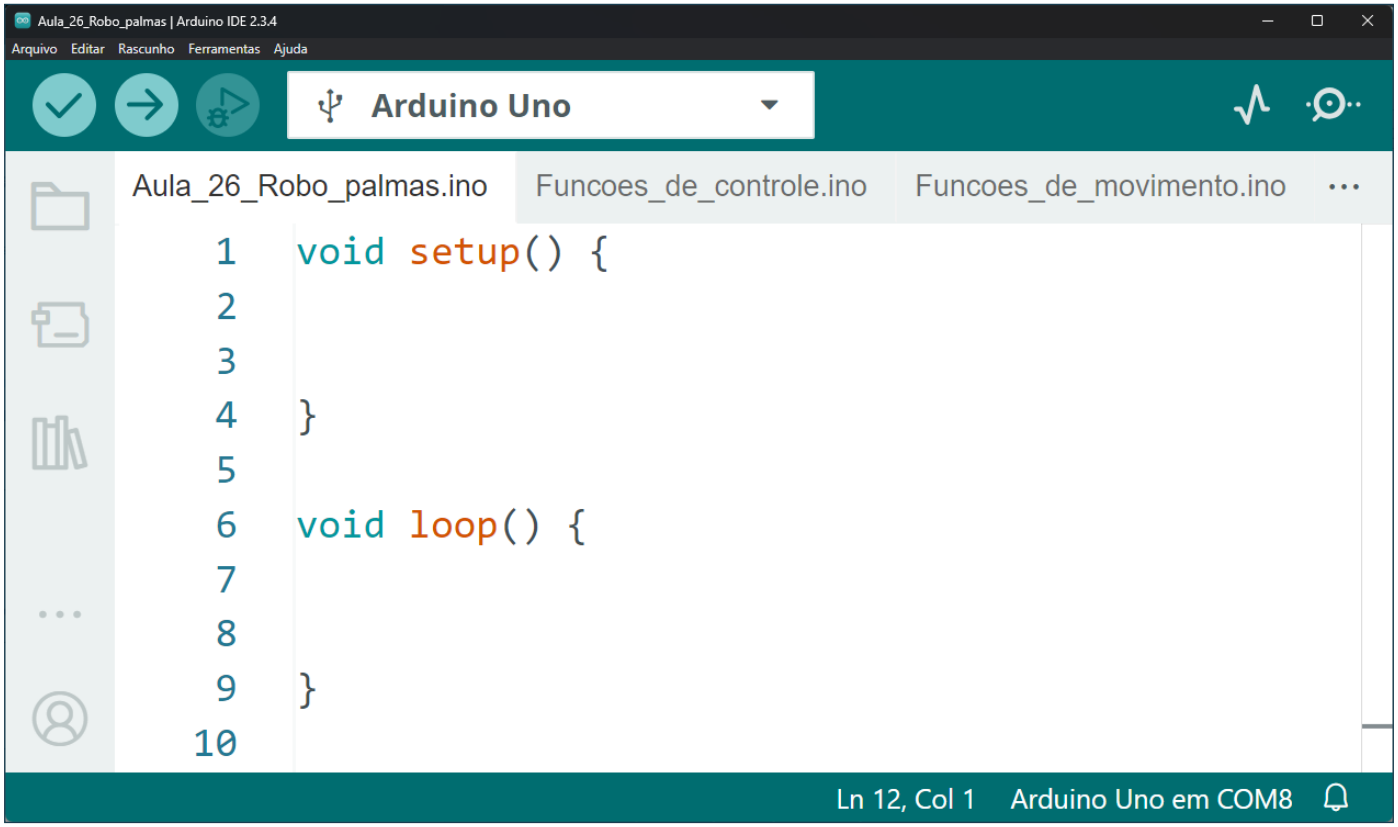
Observando as funções criadas, podemos perceber novamente, como em projetos anteriores, a presença da função **millis()** para controlar o tempo sem bloquear a execução do programa – o que aconteceria se usássemos o **delay()**, permitindo que outras tarefas continuem sendo executadas. Vamos revisar como **millis()**, que retorna o número de milissegundos que se passaram desde que o Arduino começou a executar a programação atual, é utilizada no nosso robô que interage por palmas?

- **Iniciar o modo de escuta:**
  - `tempoInicioEscuta = millis()` e `ultimaDeteccaoSom = millis()`: armazenam o tempo atual em que a função foi chamada. Isso é essencial para saber quando o modo de escuta começou e para controlar o debounce.
- **Contar palmas:**
  - `unsigned long tempoAtual = millis()`: armazena o tempo atual a cada iteração do loop.
  - `(tempoAtual - ultimaDeteccaoSom >= debounceTempo)`: verifica se o tempo passado desde a última detecção de som é maior que o tempo de debounce (250 ms). Isso evita que uma mesma palma seja contada mais de uma vez devido a ruídos ou vibrações.

- **Executar movimento:**
  - Na `executarMovimento()`, **millis()** não é usada diretamente, mas é essencial para garantir que as funções que dependem do tempo funcionem corretamente sem bloquear a execução do programa.

Vamos finalizar a programação do robô que interage por palmas com o sketch principal e verificar mais uma utilização de **millis()**.

Figura 5 - Sketch principal



Fonte: Arduino IDE.

O sketch principal integra todos os componentes e funções do robô. Vamos completá-lo com a configuração dos pinos e inicialização do monitor serial e do robô parado no **void setup()** e, pelo **void loop()**, gerenciar a detecção de palmas, o modo de escuta e a execução de movimentos com base no número de palmas detectadas, o que permitirá o robô responder ao controle por palmas de maneira contínua e eficiente.

```
#include "preambulo.h"  /* Inclui o arquivo de cabeçalho "preambulo.h". */

/* Função de configuração inicial. */
void setup() {
  Serial.begin(9600);  /* Inicializa o monitor serial a 9600 bps. */
  Serial.println("Iniciando o sistema...");  /* Imprime mensagem inicial. */

  pinMode(pinSensorSom, INPUT);  /* Configura o pino do sensor de som como entrada. */

  /* Configura os pinos da ponte H como saída. */
  pinMode(pinIN1, OUTPUT);
  pinMode(pinIN2, OUTPUT);
  pinMode(pinIN3, OUTPUT);
  pinMode(pinIN4, OUTPUT);

  pararRobo();  /* Inicializa o robô parado. */
}

/* Função principal que roda continuamente. */
void loop() {
  if (escutandoPalmas) {  /* Verifica se o robô está no modo de escuta. */
    contarPalmas();  /* Chama a função para contar palmas. */
    if (millis() - tempoInicioEscuta >= tempoEscuta) {  /* Verifica se o tempo de escuta terminou. */
      Serial.print("Tempo de escuta terminado. Palmas contadas: ");
```

```
      Serial.println(contadorPalmas);  /* Imprime o número de pal-
mas contadas. */
      executarMovimento();  /* Executa o movimento correspondente
ao número de palmas contadas. */
    }
  } else { /* Se não estiver no modo de escuta, verifica o estado do sensor de som.
*/
    int leituraSom = digitalRead(pinSensorSom);  /* Lê o estado
do sensor de som. */
    if (leituraSom == HIGH && estadoAnteriorSom == LOW) {  /* De-
tecta transição de LOW para HIGH no sensor de som. */
      iniciarEscuta();  /* Inicia o modo de escuta ao detectar a
primeira palma. */
    }
    estadoAnteriorSom = leituraSom;  /* Atualiza o estado ante-
rior para próxima iteração do loop. */
  }
}
```

Como podemos também observar pelas linhas do sketch principal e seus comentários, no **void loop()** a função **millis()** compara o tempo atual com o tempo de início da escuta para determinar se o período de escuta terminou, permitindo a execução dos movimentos do robô com base no número de palmas detectadas – e esse recurso será experimentado na Parte II.

Confira o código completo, pelo Arduino IDE Online, do robô que interage por palmas!

Aula 26 PalmaBot - I

Nesta aula realizamos a programação do robô que interage por palmas, compreendendo todas as suas funções e modos de controle, ou melhor, modos de operação para interpretar sons de palmas como comandos, movendo-se para frente, para trás ou mudando de direção. Na próxima aula, nos dedicaremos à montagem do nosso robô com base no **modelo 3D**. E aí, sim, poderemos testar suas funções e fazer os ajustes necessários no sensor de som e ver a capacidade do nosso robô perceber o mundo. Até lá!

Desafio:

Pela proposta inicial do robô e recursos aplicados às funções, quais outros componentes você adicionaria ao projeto?

E se...

O código de programação não compilar corretamente, verifique se todas as funções foram devidamente criadas e com a mesma grafia.

Verifique também, no sketch principal, a inclusão do preâmbulo da programação pela diretiva **#include "preambulo.h"**.

3. Feedback e finalização

O aprimoramento de projetos é passo essencial para, cada vez mais, ampliarmos as potencialidades da Robótica e refletirmos cada vez mais sobre os conceitos de interação e controle!

O uso de **millis()** na programação permite controlar o tempo de forma eficiente e precisa, sem interromper a execução do restante do código. Isso é particularmente útil para debouncing, contagem de eventos e execução de ações baseadas em intervalos de tempo, como poderemos verificar ao experimentarmos o robô e sua interação com o ambiente.

REFERÊNCIAS

ARDUINO. **Documentação de Referência da Linguagem Arduino**. Disponível em: https://www.arduino.cc/reference/pt/. Acesso em: 27 mai. 2024.

DIRETORIA DE TECNOLOGIAS E INOVAÇÃO (DTI)

COORDENAÇÃO DE TECNOLOGIAS EDUCACIONAIS (CTE)

EQUIPE ROBÓTICA PARANÁ

- Adilson Carlos Batista
- Ailton Lopes
- Andrea da Silva Castagini Padilha
- Cleiton Rosa
- Darice Alessandra Deckmann Zanardini
- Edna do Rocio Becker
- Kellen Pricila dos Santos Cochinski
- Marcelo Gasparin
- Michele Serpe Fernandes
- Michelle dos Santos
- Roberto Carlos Rodrigues
- Sandra Aguera Alcova Silva
- Viviane Dziubate Pittner

Os materiais, aulas e projetos da "Robótica Paraná", foram produzidos pela Coordenação de Tecnologias Educacionais (CTE), da Diretoria de Tecnologia e Inovação (DTI), da Secretaria de Estado da Educação do Paraná (SEED), com o objetivo de subsidiar as práticas docentes com os estudantes por meio da Robótica.
Este material foi produzido para uso didático-pedagógico exclusivo em sala de aula.