Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Campus Hortolândia
Eletroeletrônica-Projeto Integrador. Profº.Ricardo Leite.
Nomes: Gabriel Alonso de Castro, Leonardo Galvão de Freitas, Matheus Alves Ramos e Pedro Augusto Melo.

ENTREGA FINAL DO RELATÓRIO

São Paulo

2025

Introdução Teórica

De acordo com a rede de notícias CNN Brasil, no ano de 2024, o Brasil registrou mais de 2 mil acidentes elétricos. E infelizmente, mais de 800 mortes por causas que variam desde incêndios de origem elétrica e descargas atmosféricas, até acidentes elétricos no campo de trabalho - com pessoas que possuem experiência na área. Tendo em vista esta última citação, é notório que com o passar do tempo, essa área tende a ser mais automatizada, com o objetivo de evitar mortes de trabalhadores que interagem com altas tensões.

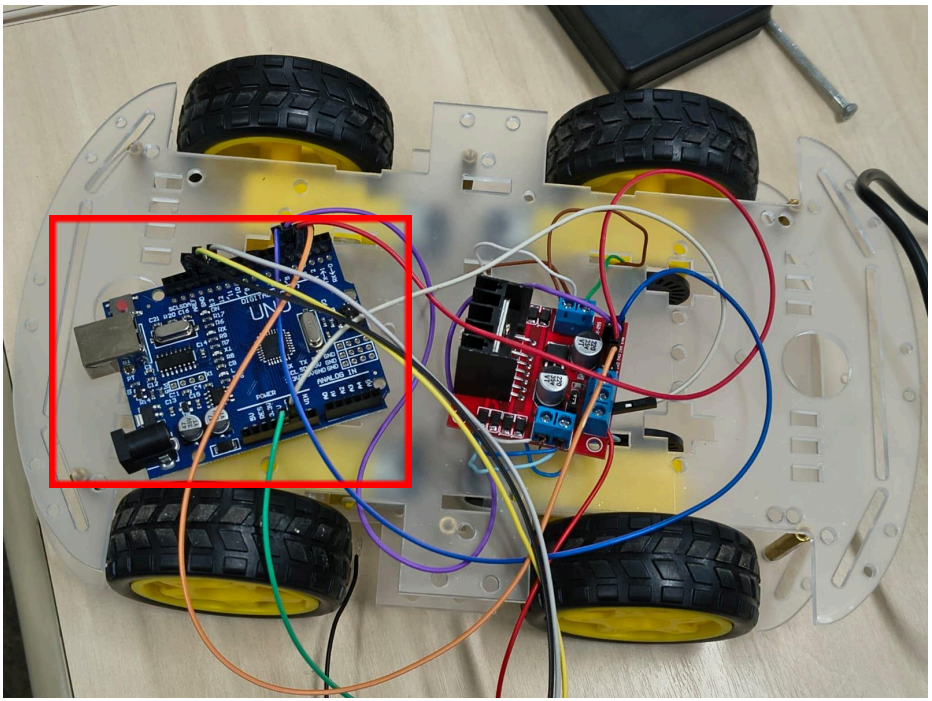
Concomitantemente à segurança, a modernização do setor elétrico deve caminhar junto com a inclusão e a acessibilidade. As exigências físicas da profissão, como o carregamento de malas de ferramentas pesadas, muitas vezes excluem profissionais que têm mobilidade reduzida ou limitações físicas.

Visando acabar com essas barreiras, este projeto propõe um veículo autônomo assistivo. Utilizando sensores ultrassônicos para locomoção, o protótipo assume funções desde o transporte de cargas até a limpeza do ambiente de trabalho, garantindo que a limitação não seja um impedimento para a atuação técnica, além de contribuir para um ambiente de trabalho mais organizado e seguro.

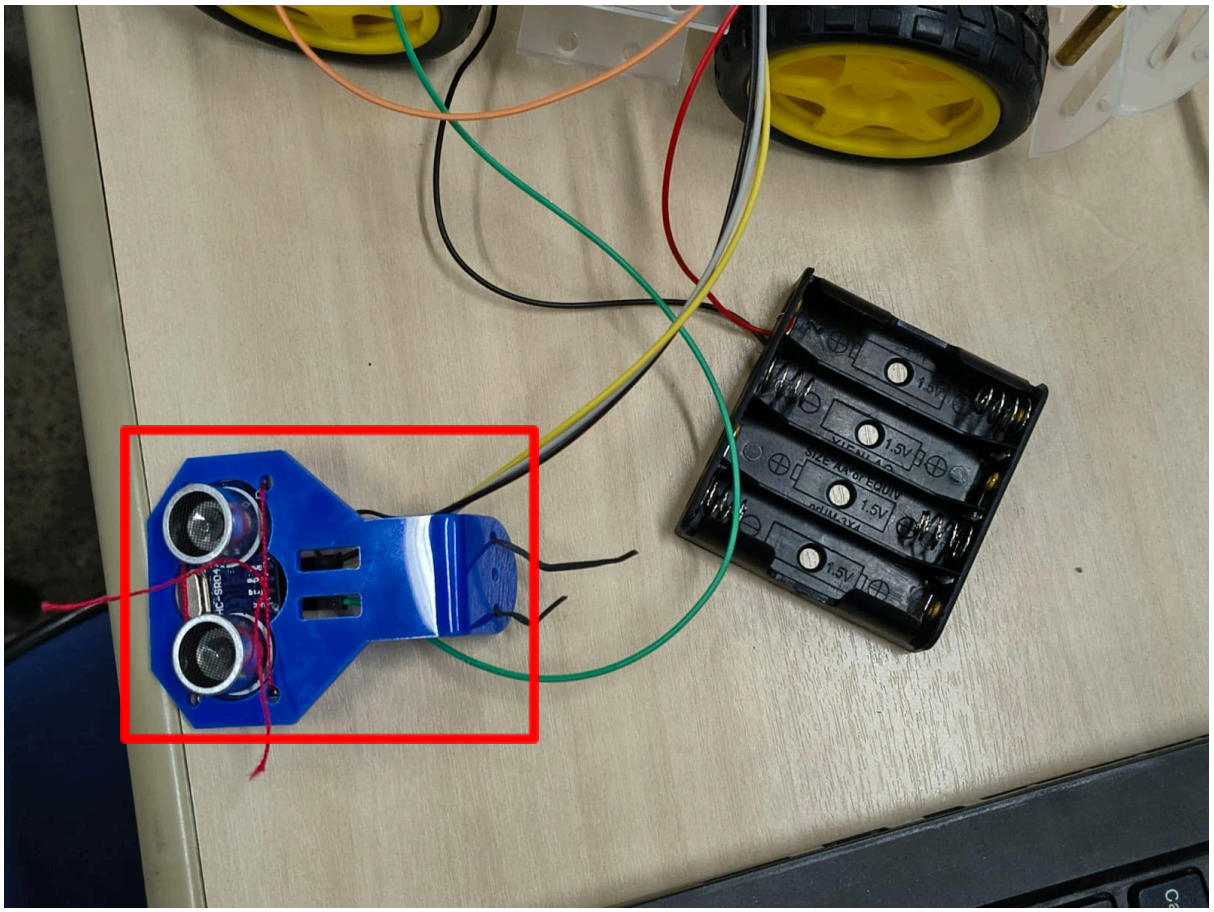
Funcionamento do Protótipo/Resultados

O sistema do carrinho assistivo foi projetado de maneira que o microcontrolador central recebe a tensão de alimentação e os dados, atuando sobre os drivers e servomecanismos/motores. Abaixo, será detalhada a função específica de cada componente na arquitetura do projeto:

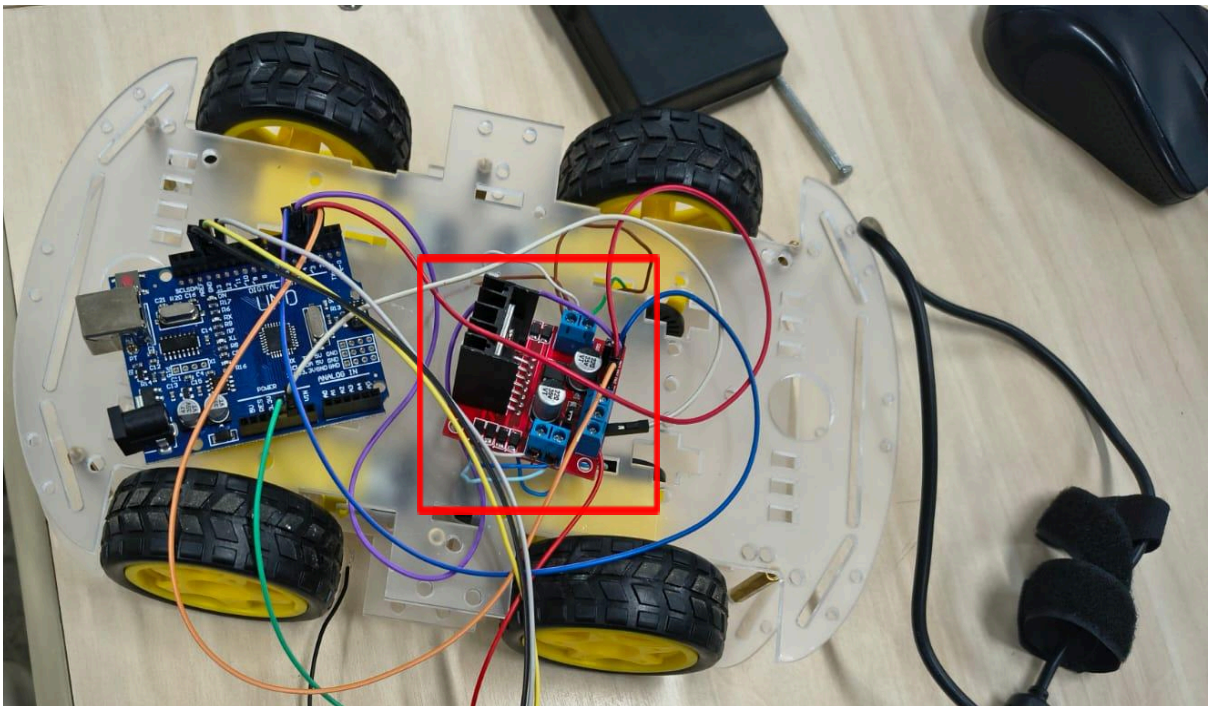
- Microcontrolador (Arduino Uno): Atua como a unidade central de processamento dos dados do nosso projeto. Foi escolhido devido a sua versatilidade, intuitividade para programar e compatibilidade com outros componentes. Sendo responsável por processar os dados emitidos pelos sensores, o Arduino Uno é o “cérebro” do carrinho. Ele calcula a distância com base no tempo de resposta do sensor e programação inserida nele e envia para os motores.



- Sensor Ultrassônico (HC-SR04): Esse sensor funciona emitindo pulsos ultrassônicos (40 kHz, de acordo com o seu datasheet). O sistema calcula a distância do obstáculo baseando-se no tempo do eco rebater no mesmo. Com isso, é pensado que o carrinho pare a uma distância segura, evitando colisões com obstáculos e pessoas no campo de trabalho.



- Driver Ponte H (L298N): Como o Arduino trabalha com correntes baixas (na casa dos mA, pelo datasheet), insuficientes para girar os motores do carrinho. O L298N recebe os sinais do Arduino e a corrente da bateria para os motores, permitindo o controle de velocidade e uma provável inversão do sentido de rotação.



- Motores DC com Caixa de Redução (3-6V): Esses motores com redução são estratégicos para este projeto, pois como o objetivo é carregar peso (de ferramentas ou outros utensílios), a caixa de redução age, convertendo a elevada RPM do motor em torque, dando a possibilidade de acelerar com peso mais elevado.

- Bateria Li-Ion 7V: Garante o correto funcionamento de todo o sistema do carrinho. A tensão nominal de 7V é a necessária para alimentar o Arduino (pelo pino de alimentação do Arduino) e os motores (pelo pino Vin).

- Micro Servo Motor SG90: Usamos para fazer o sistema de levante da nossa vassoura (que foi improvisada de um pincel).

Descrito o funcionamento individual de cada componente, abaixo o circuito baseado para a montagem do nosso carrinho:

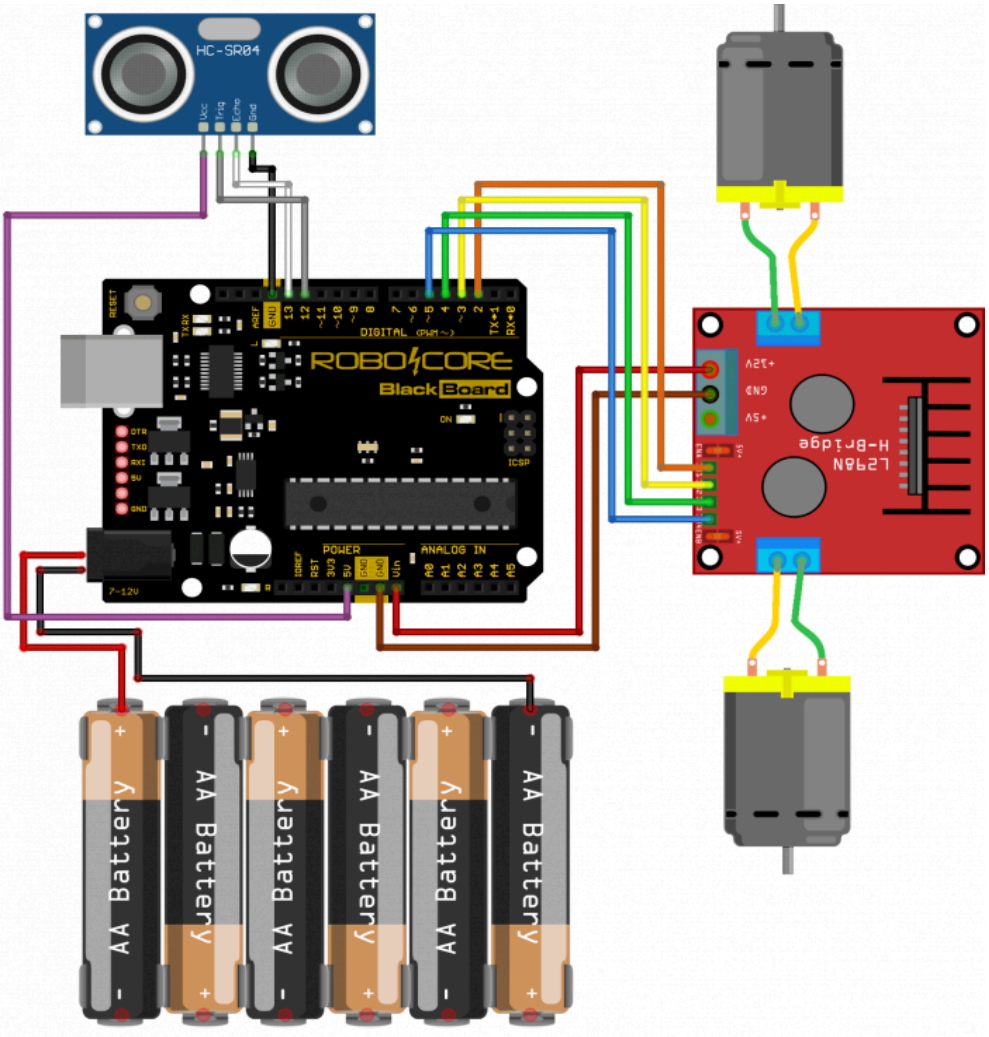
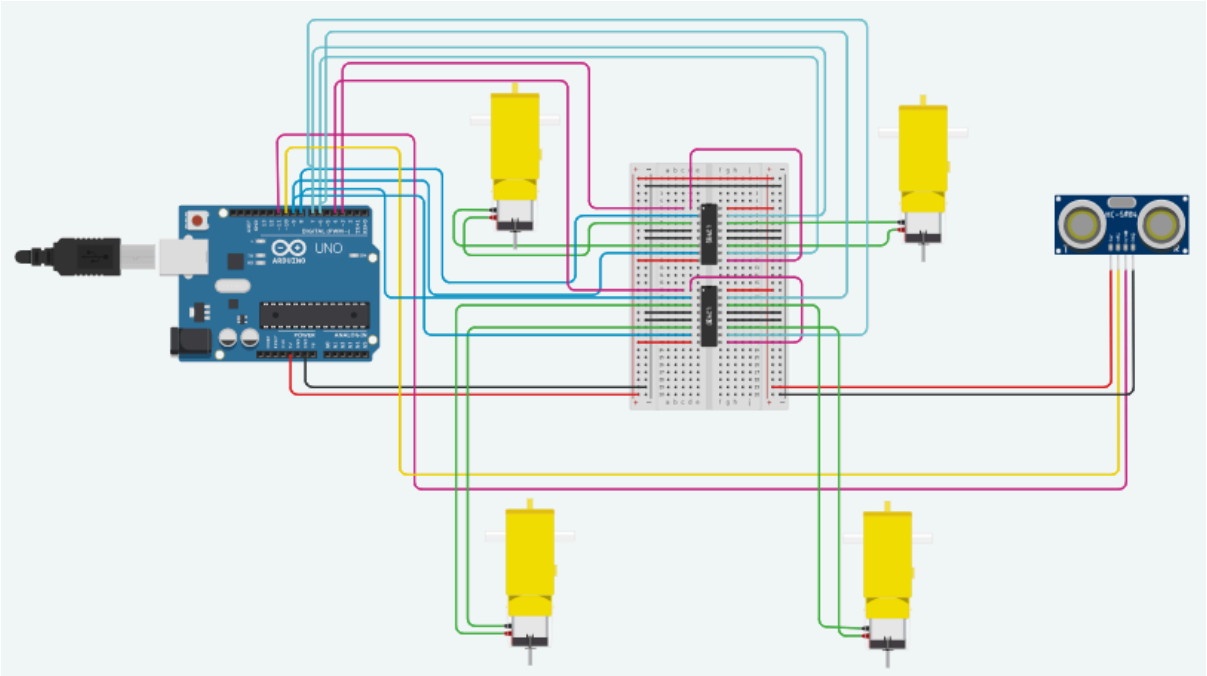
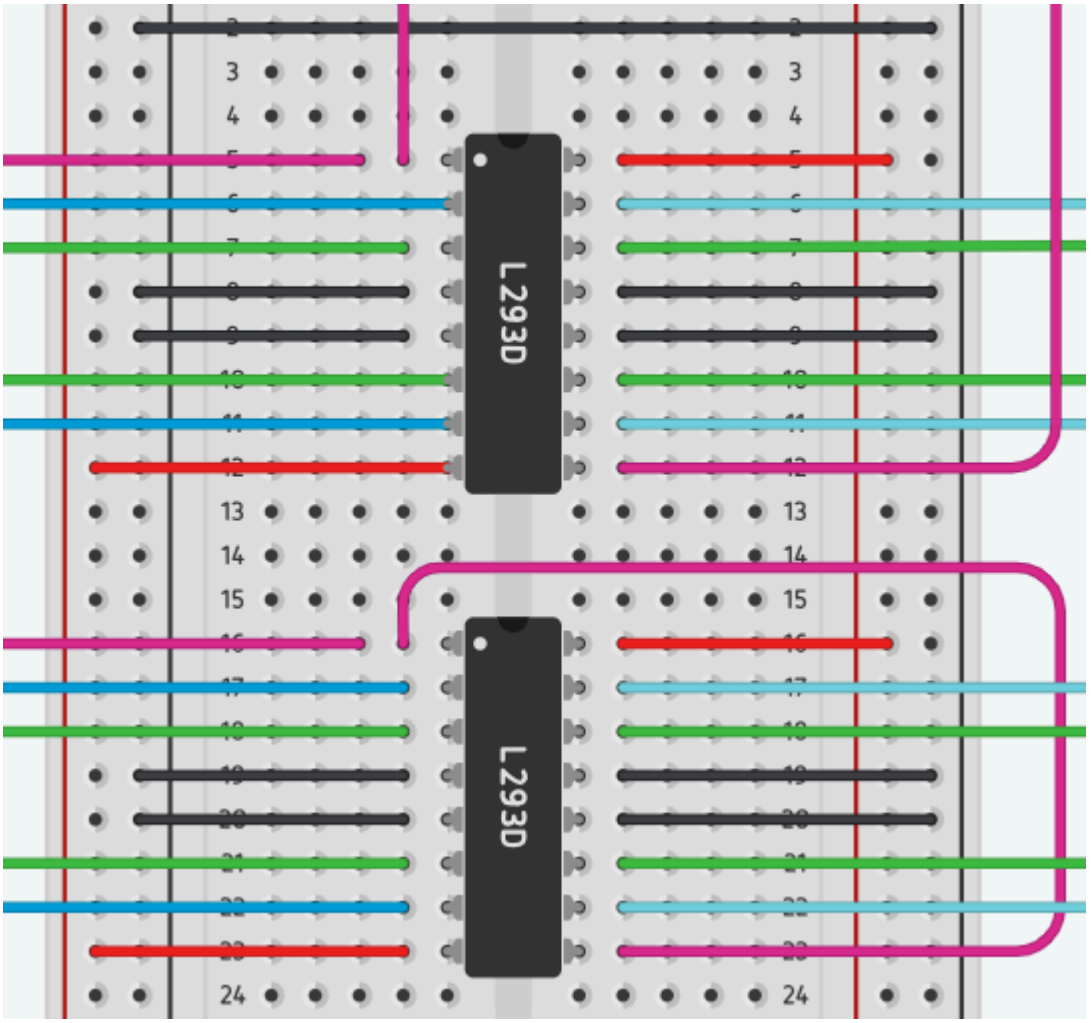


Imagem retirada do site "RoboCore", pelo qual nos inspiramos para montar nosso carrinho.

A principal diferença é que adaptamos mais dois motores para maior potência e torque no carrinho, além de utilizarmos baterias ao invés das pilhas. Abaixo, segue o esquemático feito pelo *Tinkercad*:



Nosso circuito foi montado exatamente assim na prática. Circuito via Tinkercad.



A principal diferença está nesse circuito integrado, pois no Tinkercad não possui o driver de motor que usamos no nosso carrinho. Para simular corretamente, usamos esse CI.

Abaixo, está o código inserido no Arduino, que permite o correto funcionamento do carrinho:

CodigoVassoura.ino

```
#include <Servo.h> // Inclui a biblioteca para controlar servo motores

// CONFIGURAÇÃO DO SERVO:
Servo servoVassoura; // Cria uma espécie de variável, chamada "servoVassoura" para controlar o motor
const int PINO_SERVO = 11; // Define que o fio de sinal do servo está ligado no pino digital 11
const int ANGULO_BAIXO = 55;    // Define o ângulo (em graus) para a vassoura ficar no chão
const int ANGULO_ALTO = 155;    // Define o ângulo (em graus) para a vassoura levantar

// SENSOR ULTRASSÔNICO:
const int PINO_SENSOR_ECHO = 13; // Define o pino Echo (receptor) do sensor no pino 13
const int PINO_SENSOR_TRIGGER = 12; // Define o pino Trigger (emissor) do sensor no pino 12

// DRIVER DE MOTOR(Ponte H L298N):
const int PIN_MOTOR_IN1 = 2; // Define o pino IN1 do motor A no pino 2
const int PIN_MOTOR_IN2 = 3; // Define o pino IN2 do motor A no pino 3
const int PIN_MOTOR_IN3 = 4; // Define o pino IN3 do motor B no pino 4
const int PIN_MOTOR_IN4 = 5; // Define o pino IN4 do motor B no pino 5

// CONSTANTES:
const int DISTANCIA_SEGURA = 25; // Define a distância mínima (25cm) para detectar um obstáculo
const int PAUSA = 100; // Define o tempo de espera (100ms) entre as leituras do sensor

// DECLARAÇÃO DAS FUNÇÕES:
int ler_distancia(void);
void mover_frente(void);
void mover_tras(void);
void parar(void);
void levantar_vassoura(void);
```

CodigoVassoura.ino

```
void abaixar_vassoura(void);

void setup() {
  // CONFIGURAÇÃO DO SERVO MOTOR:
  servoVassoura.attach(PINO_SERVO); // Conecta o servo ao pino físico 11(PINO_SERVO)
  levantar_vassoura(); // Levanta a vassoura ao ligar (por segurança)

  // CONFIGURAÇÃO DO SENSOR ULTRASSÔNICO:
  pinMode(PINO_SENSOR_ECHO, INPUT); // Configura o Echo como ENTRADA (recebe os dados)
  pinMode(PINO_SENSOR_TRIGGER, OUTPUT); // Configura o Trigger como SAÍDA (envia os dados)
  digitalWrite(PINO_SENSOR_TRIGGER, LOW); // Garante que o Trigger comece desligado

  // CONFIGURAÇÃO DO DRIVER DE MOTOR:
  pinMode(PIN_MOTOR_IN1, OUTPUT); // Configura IN1 como saída
  pinMode(PIN_MOTOR_IN2, OUTPUT); // Configura IN2 como saída
  pinMode(PIN_MOTOR_IN3, OUTPUT); // Configura IN3 como saída
  pinMode(PIN_MOTOR_IN4, OUTPUT); // Configura IN4 como saída

  parar(); // Faz com que os motores comecem desligados
}

void loop() {  // Tudo entre essa função void fica em Loop
  int distancia = ler_distancia();// Chama a função que mede a distância e armazena na variável 'distancia'

  // Estrutura de decisão: Verifica se a distância lida é menor que a de 25cm
  if(distancia < DISTANCIA_SEGURA){ // Se houver um obstáculo

    parar(); // Para os motores imediatamente e já levanta a vassoura
```

```
    delay(500); // Aguarda 500ms para transicionar de ação

    mover_tras(); // Move o carrinho para trás

    delay(1000); // Fica dando ré por 1 segundo
    parar(); // Para os motores

    // Lógica para decidir para qual lado girar: millis() retorna o tempo que o Arduino está ligado, e "% 2" verifica se esse número é par ou ímpar:
    bool par = (millis() % 2 == 0) ? true : false;

    if(par){ // Se o tempo for par
      // Gira o motor 1 para frente, e o motor 2 fica desligado (curva para a esquerda)
      digitalWrite(PIN_MOTOR_IN1, HIGH);
      digitalWrite(PIN_MOTOR_IN2, LOW);
    } else { // Se o tempo for ímpar
      // Gira o motor 3 para frente, e o motor 4 fica desligado (curva para a direita)
      digitalWrite(PIN_MOTOR_IN3, HIGH);
      digitalWrite(PIN_MOTOR_IN4, LOW);
    }

    delay(500);// Mantém o carrinho girando por meio segundo
    parar(); // Para

  } else { // Se houve caminho livre, sem obstáculos

    mover_frente(); // Retorna a função de entrar para frente, e deixa a vassoura abaixada.
  }
```

CodigoVassoura.ino

```
  delay(PAUSA); // Aguarda o tempo de 100ms(nomeado em "PAUSA") de leitura entre os dados do sensor ultrassônico
}

// FUNÇÕES AUXILIARES:

int ler_distancia(void){ // Lê a distância com o sensor e envia um pulso ultrassônico
  digitalWrite(PINO_SENSOR_TRIGGER,HIGH); // Liga o trigger do sensor
  delayMicroseconds(10); // Aciona-o por 10 microssegundos
  digitalWrite(PINO_SENSOR_TRIGGER,LOW); // Desliga-o

  return pulseIn(PINO_SENSOR_ECHO, HIGH) / 58; // // Mede quanto tempo o som demorou para ir e voltar(ms), e divide por 58(conversão para cm)
}

void mover_frente(void){// Função que move o carrinho para a frente
  abaixar_vassoura(); // Abaixa a vassoura

  digitalWrite(PIN_MOTOR_IN1, HIGH); // Motor A (frente) ligado
  digitalWrite(PIN_MOTOR_IN2, LOW); // Motor A (trás) desligado
  digitalWrite(PIN_MOTOR_IN3, HIGH); // Motor B (frente) ligado
  digitalWrite(PIN_MOTOR_IN4, LOW); // Motor B (trás) desligado
}

void mover_tras(void){ // Função que move o carrinho para a frente
  levantar_vassoura(); // Levanta a vassoura

  digitalWrite(PIN_MOTOR_IN1, LOW); // Motor A (frente) desligado
  digitalWrite(PIN_MOTOR_IN2, HIGH); // Motor A (trás) ligado
```

```
    digitalWrite(PIN_MOTOR_IN3, LOW); // Motor B (frente) desligado
    digitalWrite(PIN_MOTOR_IN4, HIGH); // Motor B (trás) ligado
}

// Função para parar todos os motores
void parar(void){ // Função que para todos os motores
    levantar_vassoura(); // Levanta a vassoura

    // Desliga todos os pinos dos motores
    digitalWrite(PIN_MOTOR_IN1, LOW);
    digitalWrite(PIN_MOTOR_IN2, LOW);
    digitalWrite(PIN_MOTOR_IN3, LOW);
    digitalWrite(PIN_MOTOR_IN4, LOW);
}

// FUNÇÕES PARA A VASSOURA:
void levantar_vassoura(){
    servoVassoura.write(ANGULO_ALTO); // Envia o comando do ângulo de 155°
}

void abaixar_vassoura(){
    servoVassoura.write(ANGULO_BAIXO); // Envia o comando de ângulo de 55 graus)
}
```

Justificativas

- O que faltou fazer?
  - ➔ Módulo de Bluetooth HC-SR04: Inicialmente, avaliamos o uso deste módulo, para possível controle direto das direções do carrinho pelo próprio celular. Se tornou inviável, devido ao funcionamento do sensor ultrassônico (HC-SR04). Isso aconteceu pois não teria como usar os dois modos de comandos ao mesmo tempo, como planejado. Por isso, priorizamos a autonomia assistida e o torque para maiores cargas.
  - ➔ Sensor Fotoresistor LDR LM393: O uso desse sensor seguidor de faixa exige uma adaptação prévia do ambiente (pintar ou colar fitas no chão). Como o objetivo é a acessibilidade e a versatilidade em diferentes locais no trabalho, concluiu-se que o carrinho não deveria depender do local previamente preparado. Por isso o sensor ultrassônico mostrou-se superior para essa aplicação, permitindo a interação com o ambiente real e desviar de obstáculos sem exigir a modificação do local de trabalho.
  - ➔ Bombinha de água DC 12v: Para essa implementação, ela se tornou inviável devido ao seu funcionamento. Ela demonstraria eficiência total por volta dos 12v (após testes), o que estouraria a capacidade de fornecimento de energia da nossa bateria (7v); sendo assim, precisaria de um sistema de alimentação separado para a própria bomba de água (o que não tornaria o sistema tão prático, como o esperado desde o começo). Outro ponto é que o foco é o carrinho assistido ajudar em áreas de instalações elétricas e nesses ambientes correlacionados de trabalho: e a introdução de água no sistema aumentaria a complexidade (tanque, mangueira, rodo etc) e traria riscos desnecessários de ao ambiente de trabalho.
- O que foi alterado?
  - ➔ Problemas com a pilha e a troca para a bateria de Li-Ion: Durante a fase inicial de testes, o sistema de alimentação era dividido entre: uma bateria de 8V usada para alimentar o Arduino e um conjunto de 4 pilhas recarregáveis tipo AA (totalizando cerca de 5V) para a alimentação do Driver Ponte H e dos motores rotatórios do carrinho.
    No entanto,observou-se que ao acionar os motores com carga (peso da estrutura + arrancada inicial), o carrinho andava muito devagar, além de mais pra frente, parar de funcionar totalmente. Após medições com o multímetro e análises teóricas, percebemos que a corrente fornecida pelas pilhas era insuficiente para suprir a demanda dos motores, especialmente para o driver L298N, que era de 3000 mAh.
    Para corrigir a falha, o sistema de alimentação foi substituído por uma bateria de Li-Ion de aproximadamente 7V e que fornecia a corrente necessária. Com a nova fonte de alimentação, nosso projeto passou a operar como o esperado. Notamos um aumento significativo no torque e velocidade dos motores, permitindo ao veículo uma maior aceleração inicial e transportar cargas sem problemas.

Website / App/ Portfolio

Criamos um repositório via GitHub do nosso projeto. Pode ser acessado pelo link: https://github.com/Gabxzn/Carrinho-Assistido-com-Sensor-Ultrass-nico-Projeto-Integrador-/tree/main?tab=readme-ov-file

Referências Bibliográficas:

- CNN BRASIL. **CNN Energia: Acidentes elétricos mataram 840 brasileiros em 2024**. 8 de maio de 2024. Disponível em: https://www.youtube.com/watch?v=mfhceEwuDiY. Acesso em: 21 de novembro de 2025.
- DIY ARDUINO ROBOT. **Arduino obstacle avoiding + voice control + Bluetooth control Robot**. 18 de maio de 2021. Disponível em: https://www.youtube.com/watch?v=aE_J7B-O4VQ. Acesso em: 21 de novembro de 2025.
- HESHAN PASINDU SANKALPA. **How to make a Bluetooth controlled car using Arduino**. 2 de setembro de 2023. Disponível em: https://www.youtube.com/watch?v=gU-CZP2nIwQ. Acesso em: 21 de novembro de 2025.
- MANUAL DO MUNDO. **Como fazer um robô seguidor de linha (Robô Baratinha)**. 17 de julho de 2018. Disponível em: https://www.youtube.com/watch?v=5KwH-bQYOEc. Acesso em: 21 de novembro de 2025.
- MANUAL DO MUNDO. **Conheça os sensores do Arduino #ManualMaker Aula 6, Vídeo 1**. 28 de março de 2019. Disponível em: https://www.youtube.com/watch?v=vEdYjAbzrAE. Acesso em: 21 de novembro de 2025.
- MAGERUSESTHIS. **Bluetooth Controlled Car and Obstacle Avoiding Robot**. Arduino Forum, 10 de fevereiro de 2024. Disponível em: https://forum.arduino.cc/t/bluetooth-controlled-car-and-obstacle-avoiding-robot/1222158. Acesso em: 21 de novembro de 2025.
- ROBOCORE. **Robô Pali - Desviando de Obstáculos**. Disponível em: https://www.robocore.net/tutoriais/robo-pali-desviando-de-obstaculos?srsltid=AfmBOoqmeSNZmBO5fd2bO5JGuWs8_JFCjYLFZ5M3mS5e771K3PJk36YE. Acesso em: 21 de novembro de 2025.
- TALUKDAR, Alok. **Bluetooth Controlled Obstacle Avoidance Robot**. Arduino Project Hub, 9 de maio de 2020. Disponível em: https://projecthub.arduino.cc/alokmech007/bluetooth-controlled-obstacle-avoidance-robot-9ecb75. Acesso em: 21 de novembro de 2025.
- TINKERCAD. **Neat Blad**. Disponível em: https://www.tinkercad.com/things/3YHBs0PSy3q-neat-blad/editel?returnTo=https%3A%2F%2Fwww.tinkercad.com%2Fdashboard. Acesso em: 21 de novembro de 2025.