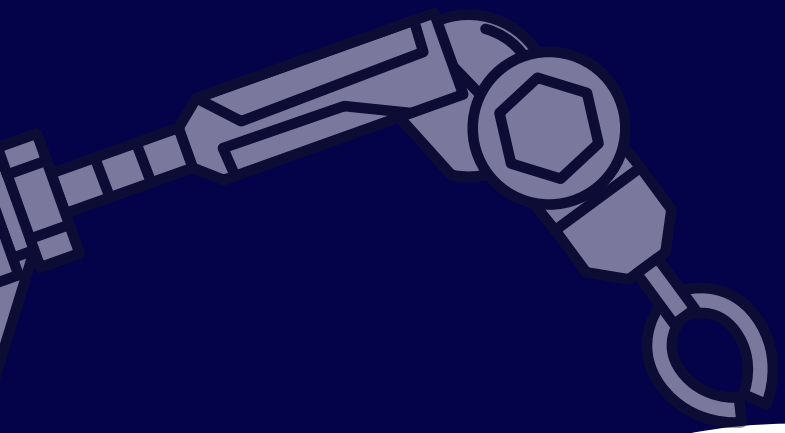
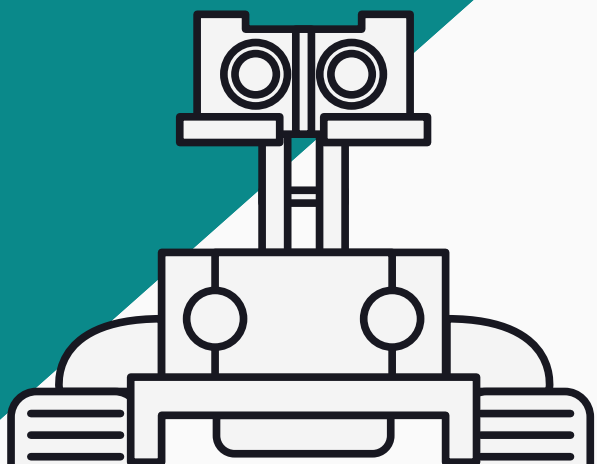


ELETRÔNICA **ÔMEGA**



E-BOOK ROBÓTICA PARA INICIANTES

Sumário

1 INTRODUÇÃO 2
2 CONCEITOS IMPORTANTES 3
3 CONTROLANDO MOTORES 17
4 SENSOR DE ESTACIONAMENTO 32
5 ROBÔ AUTÔNOMO 46
6 ROBÔ COM BLUETOOTH 62
7 BRAÇO ROBÓTICO 75
8 FINALIZAÇÃO 94

1 Introdução

Essa apostila foi desenvolvida pela **Eletrônica Ômega** e destina-se aos makers iniciantes que querem aprender robótica de forma prática e divertida! Aqui você vai aprender como construir projetos de robótica utilizando os materiais disponíveis no **Kit Arduino Robôs**. Será apresentado qual a função dos principais componentes, conceitos de eletrônica e elétrica, possíveis projetos e como montar um **Robô Autônomo**, um **Robô Controlado Por App via Bluetooth**, um **Braço Robótico Controlado por Joysticks** e um **Braço Robótico Controlada via Bluetooth**.

Kit Arduino Robôs

O **Kit Arduino Robôs** é ideal para os makers que estão pensando em desenvolver algum veículo ou robô com Arduino, pois inclui um chassi que acompanha **rodas e motores**, um **braço robótico**, **ponte H** para controlar motores, **sensor de distância** e vários outros itens comuns nos projetos de robótica, a sua imaginação será o limite para os projetos que este Kit é capaz de oferecer !!!

Você pode adquirir esse Kit **Clicando Aqui** !!

2 Conceitos Importantes

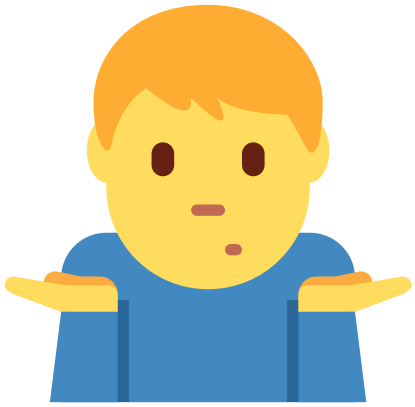
Antes de iniciarmos nossos projetos é necessário conhecermos alguns **conceitos** que serão utilizados ao decorrer desta apostila.

O que é Arduino ??

O Arduino é uma plataforma eletrônica de código aberto desenvolvido na Itália que une hardware e software simples para uso. As placas Arduino conseguem ler **Entradas** e transformar em **Saídas** graças a um **Microcontrolador** embutido. Dessa forma, através de códigos é possível dizer a sua placa o que ela deve fazer. Por ser uma plataforma de código aberto, qualquer pessoa pode modificar, melhorar e personalizar, utilizando o mesmo hardware do Arduino, porém não é permitido utilizar o nome Arduino e nem a logo.

Tá.... mas o que é uma Entrada e uma Saída ?

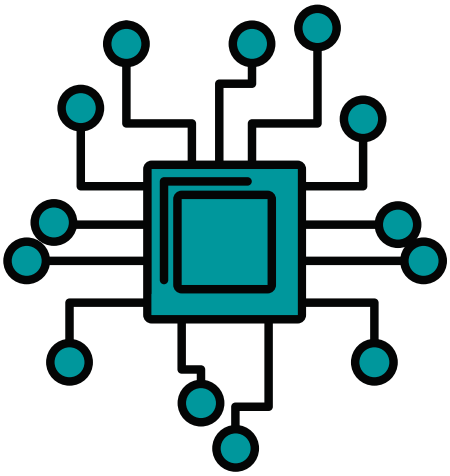
De forma resumida as entradas são por exemplo apertar um **botão**, quando ele é acionado é enviado um sinal para o **microcontroldor** do Arduino que pode ativar algo, como por exemplo um **LED** ou um **motor**.



Já as saídas são exatamente o que é ativado pelo Arduino, como um **LED**, **motor**, **buzzer**, entre outros.

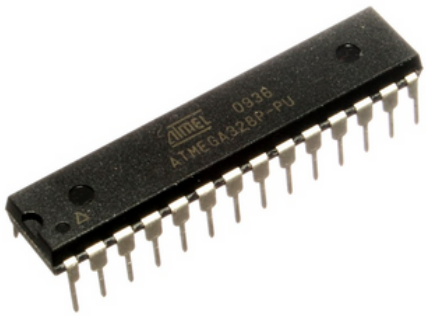
Microcontroladores

Os microcontroladores são uma junção de Software e Hardware que através da programação conseguimos controlá-los para desempenhar tarefas. Ele é um tipo especial de circuito integrado, visto que conseguimos programar os microcontroladores para realizar tarefas específicas. Em breve vamos descobrir diversas coisas que podemos fazer utilizando esses microcontroladores. O **microcontrolador** utilizado no **Arduino Uno R3**, o qual utilizaremos é o **ATmega 328.**



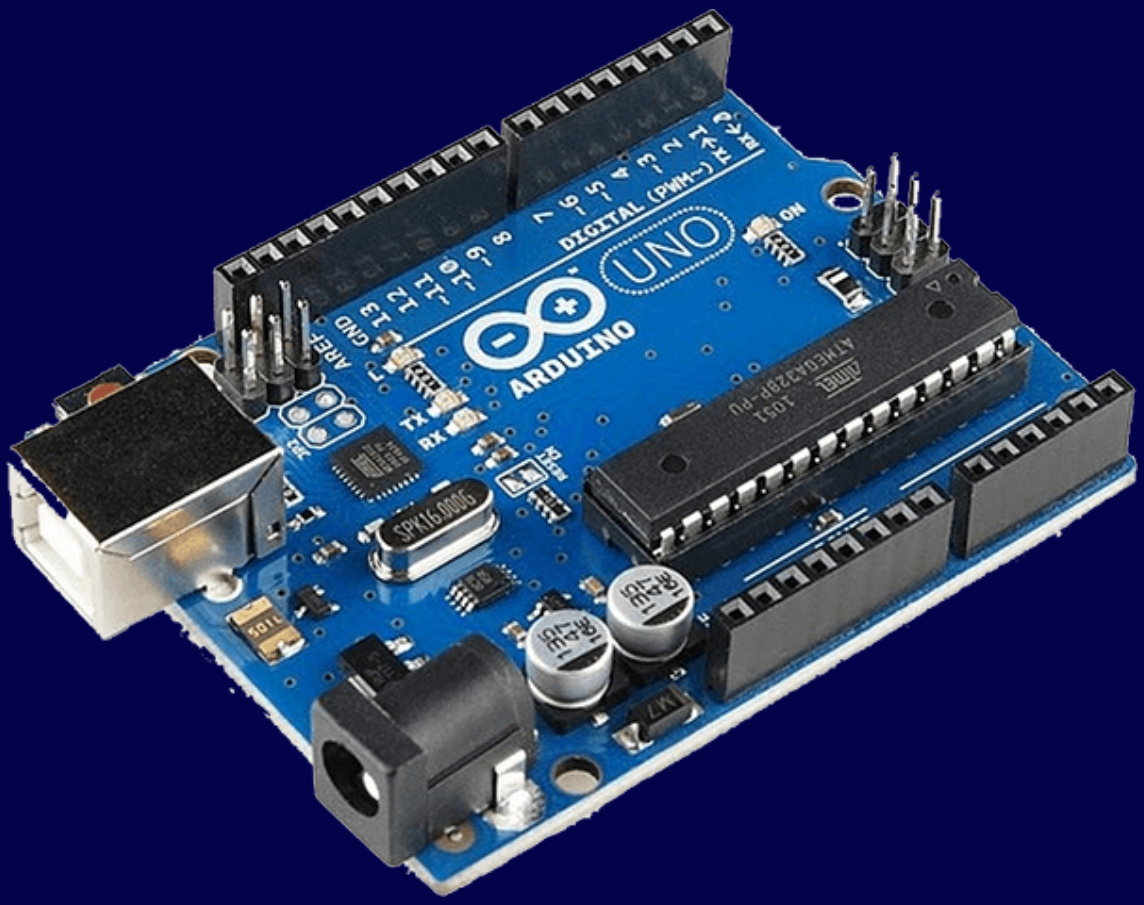
ATmega

Os ATmegas são microcontroladores desenvolvidos pela empresa Atmel.



Qual Arduino Utilizaremos ??

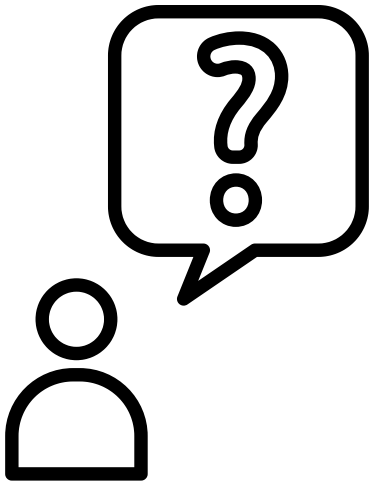
Em nossos projetos vamos utilizar a placa Arduino Uno, esse modelo é o mais conhecido e mais utilizado por pessoas que querem ingressar no mundo da **Robótica** e **Automação**. Ele possui 14 **portas digitais** e 6 delas podem ser utilizadas como saída **PWM** (controle de dispositivos variando a intensidade). Além disso, possui um microcontrolador ATmega328 que possui 32 KB de memória, que é suficiente para desenvolver diversos projetos.



OBS: Se o Arduino Uno for alimentado com uma tensão abaixo de **7V**, tensão de funcionamento da placa, pode ficar instável e se for alimentada com tensão acima de **12V**, o regulador de tensão da placa pode superaquecer e acabar danificando o Arduino.

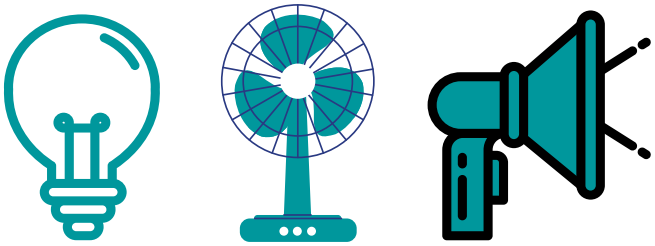
Sinais Digitais

Acabamos de conhecer mais sobre o **Arduino Uno**, como visto anteriormente esse modelo possui 14 portas digitais, mas afinal, você sabe o que é um **Sinal Digital ?**



O **Sinal Digital** possui uma quantidade limitada, geralmente é representado por dois níveis (Alto ou Baixo). Assim é definido para instantes de tempo e o conjunto de valores que podem assumir é limitada. Podemos usar como **exemplos** de **Saídas Digitais:**

- Lâmpadas;
- Buzinas;
- Ventiladores.



Observe que todos esses equipamentos estão ou ligados ou desligados, não há uma variação contínua.

E também existem as Entradas Digitais, por exemplo:

- Chaves seletoras;
- Fins de Curso;
- Botões.

Sinais Analógicos

Além de portas digitais, o Arduino também possui **Portas Analógicas**. O **Sinal Analógico** é um sinal que pode assumir infinitos valores em um intervalo de tempo. **Exemplos** de **Sinais Analógicos** são:

- **Sensores de Temperatura;**
- **Sensores de Umidade;**
- **Sensores de Pressão;**
- **Potenciômetros.**

Simplificando...

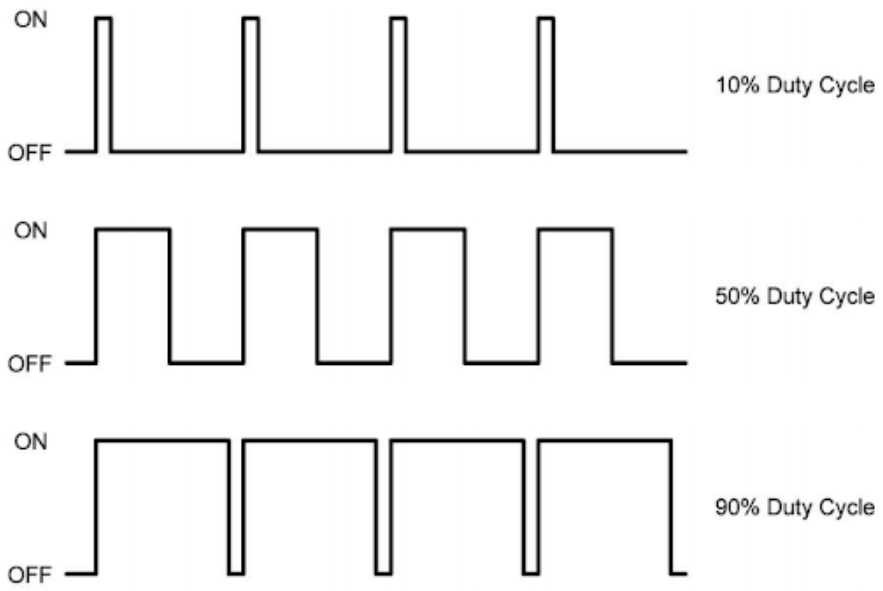
Para simplificar, pense no interruptor e um dimmer de uma lâmpada…

Ou o interruptor está **acionado** e a lâmpada está **acesa** ou está **desacionado** e a lâmpada **apagada**, não existe outro estado. Dessa forma o interruptor é considerado um Sinal Digital.

Já o dimmer controla a intensidade da lâmpada, girando o potenciômetro conseguimos regular a intensidade de brilho. Esse é considerado o sinal analógico, pois existem infinitos valores.

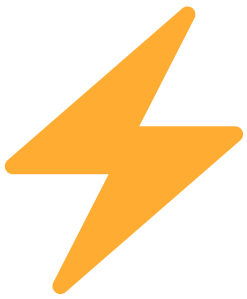
PWM

Outro conceito que aparecerá muito em nossos projetos é o **PWM**, que significa **Pulse Width Modulation**, traduzindo para o português fica Modulação por Largura de Pulso. Ele nada mais é que uma técnica desenvolvida para obter resultados analógicos por meios digitais. Essa técnica consiste na geração de uma onda quadrada em uma frequência muita alta em que pode ser controlada a percentagem do tempo em que a onda permanece em nível lógico alto. Esse tempo é chamado de **Duty Cycle** e sua alteração provoca mudança no valor médio da onda, indo desde 0V (0% de Duty Cycle) a 5V (100% de Duty Cycle) no caso do Arduino. Vamos utilizar o **PWM** para controlar a velocidade dos motores do nosso robô.



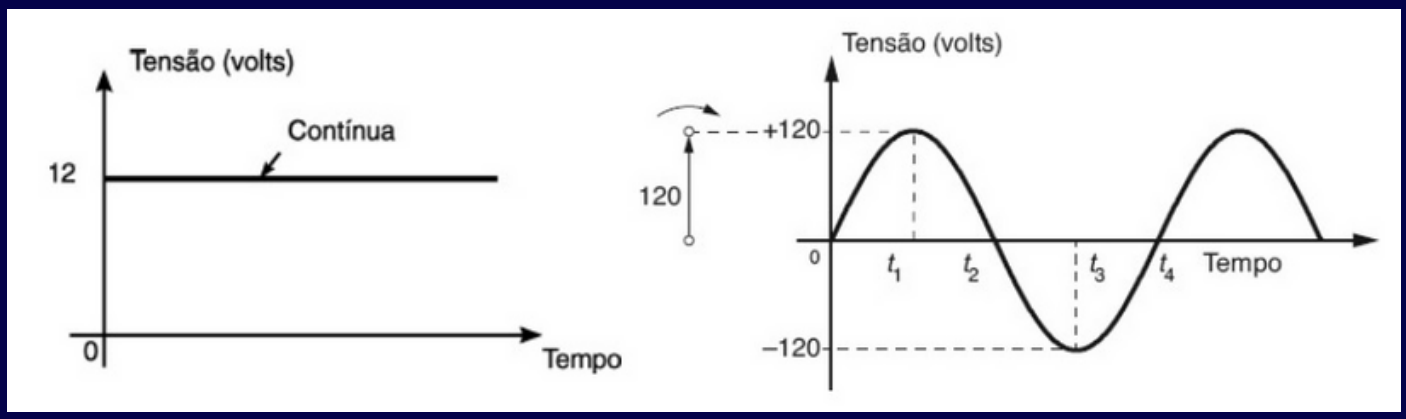
Tensão Elétrica

Também conhecida como **Diferença de Potencial**, a **tensão** é a grandeza que mede a diferença de potencial elétrico entre dois pontos. É a força necessária para mover os elétrons e criar assim uma **corrente elétrica**. O equipamento utilizado para medir a tensão é conhecido como **voltímetro**, no **Sistema Internacional** a unidade de medida é **Volts**, onde o símbolo é **V**.



Tipos de Tensão

A tensão pode ser **Contínua** ou **Alternada**. Na contínua ela não muda de polaridade com o passar do tempo, já na alternada é ao contrário, ou seja, muda de polaridade com o passar do tempo.



Exemplos

A **Contínua** não muda de polaridade com o passar do tempo. A tensão das pilhas são contínuas.

Já a tensão **Alternada** a polaridade será alternada de acordo com a frequência, no caso de uma tomada no Brasil, a frequência normal é de 60Hz, o que quer dizer que a polaridade desta tensão vai alternar 60 vezes por segundo.

Fórmulas

Podemos calcular a **Tensão Elétrica** utilizando a **Lei de Ohm**, onde a tensão é igual a resistência vezes a corrente elétrica.

Assim, temos a seguinte fórmula:

Tensão = Resistência x Corrente

↓

Resumindo: $\boxed{U = R \times I}$

OBS: **U** = Tensão
R = Resistência
I = Corrente

Também é possível calcular a tensão se caso soubermos o valor da **Corrente** e da **Potência** elétrica, onde tensão elétrica é igual a potência dividida pela corrente.

Assim, temos a seguinte fórmula:

$$\textbf{Tensão} = \frac{\text{Potência}}{\text{Corrente}}$$

↓

Resumindo: $\boxed{U = \frac{P}{I}}$

OBS: **U** = Tensão
P = Potência
I = Corrente

OBS: O termo **Voltagem** apesar de ser muito falado não existe, é apenas um termo popular.

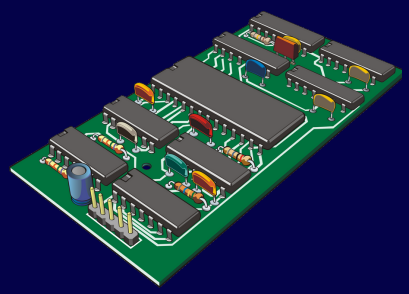
Corrente Elétrica

A **Corrente Elétrica** é um fenômeno físico onde ocorre o movimento de cargas elétricas, como os elétrons, que acontece no interior de diferentes materiais em razão da aplicação de uma diferença de potencial elétrico. A corrente elétrica é uma grandeza escalar, sua unidade de medida de acordo com o Sistema Internacional de Unidades, é o **Ampère**, cujo símbolo é **A**. Essa unidade mede o módulo da carga elétrica que atravessa a secção transversal de um condutor a cada segundo e por isso também pode ser escrita como **Coulombs Por Segundo**, onde o símbolo é **C/s**.

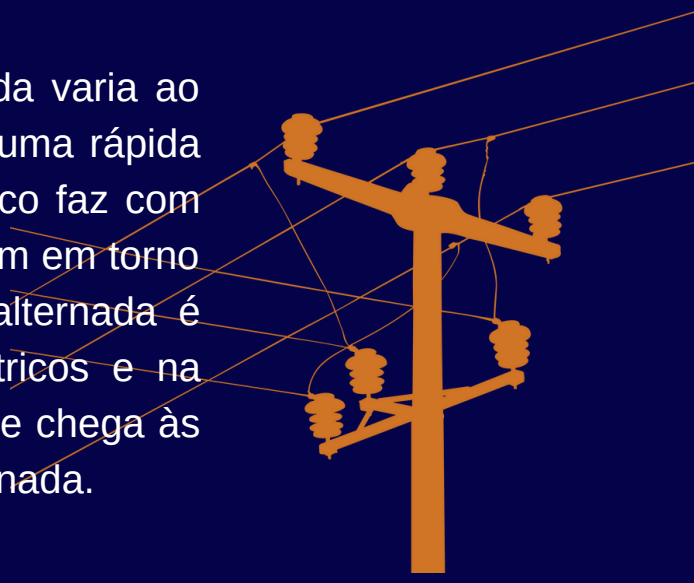
Tipos de corrente

Assim como a **Tensão Elétrica**, existem dois tipos de corrente, a **Corrente Alternada** **(CA)** e a **Corrente Contínua** **(CC)**

- **Corrente Contínua:** Não tem variação ao longo do tempo e se mantém praticamente constante, os elétrons são forçados a deslocar-se em sentido único. Esse tipo de corrente é comum em dispositivos que utilizam baixas tensões, como eletrônicos em geral.



- **Corrente Alternada:** A corrente alternada varia ao longo do tempo. Nesse tipo de corrente, uma rápida inversão de polaridade do potencial elétrico faz com que os elétrons se desloquem em vai e vem em torno de uma posição fixa. Corrente elétrica alternada é utilizada principalmente em motores elétricos e na transmissão de eletricidade, a corrente que chega às nossas casas é uma corrente elétrica alternada.



Fórmulas

Para se saber qual a intensidade da **Corrente Elétrica** em um condutor, é possível utilizar a equação de carga elétrica pelo tempo.

Assim, temos a seguinte fórmula:

$$\textbf{Intensidade de Corrente} = \frac{\text{Carga elétrica}}{\text{Intervalo de tempo}}$$

Resumindo: $\boxed{I = \frac{Q}{\Delta t}}$

OBS: **Q=** Carga elétrica
Δt = Intervalo de tempo
I = Corrente

Utilizando a **Tensão Elétrica** e a **Resistência** também é possível calcularmos a corrente, onde corrente é igual tensão elétrica dividido pela resistência.

Assim, temos a seguinte fórmula:

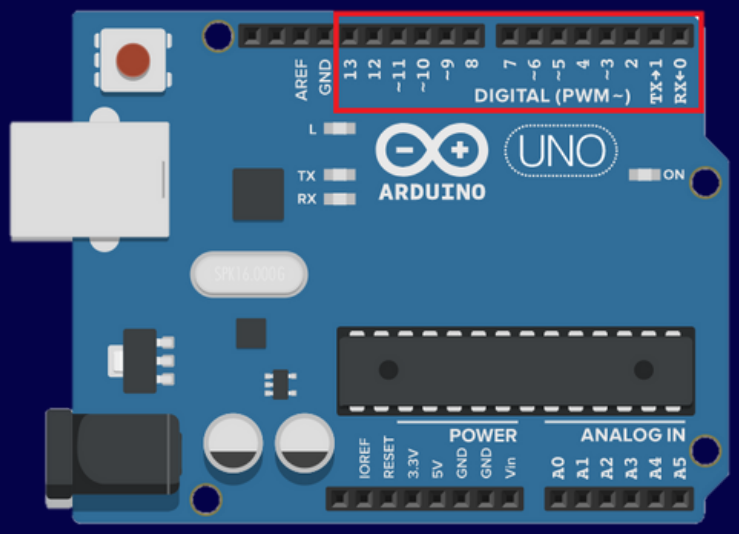
$$\textbf{Corrente} = \frac{\text{Tensão}}{\text{Resistência}}$$

Resumindo: $\boxed{I = \frac{U}{R}}$

OBS: **U =** Tensão
R = Resistência
I = Corrente

Portas do Arduino Uno

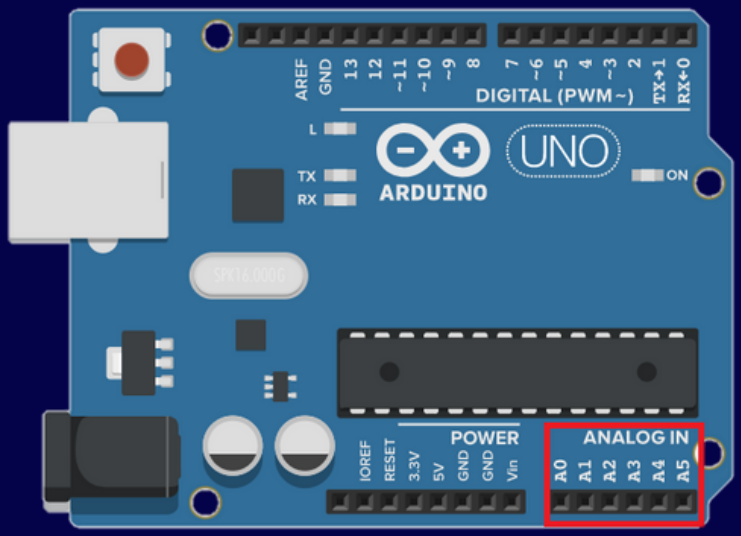
Bem, agora que já conhecemos os principais conceitos que vão ser utilizados, vamos conhecer um pouco mais sobre o modelo de Arduino que utilizaremos.



Como já sabemos, o modelo Uno possui 14 **Portas Digitais** que podem ser utilizadas como **Entradas** ou **Saídas**, sendo assim, podemos comandar 14 dispositivos! As Portas Digitais vão do número **0** ao **13**. Os números que possuem o símbolo **~** são as que podem ser utilizadas como **Portas PWM**. São essas: **3**, **5**, **6**, **9**, **10** e **11**. É válido lembrar que as portas **PWM** também funcionam como portas digitais.

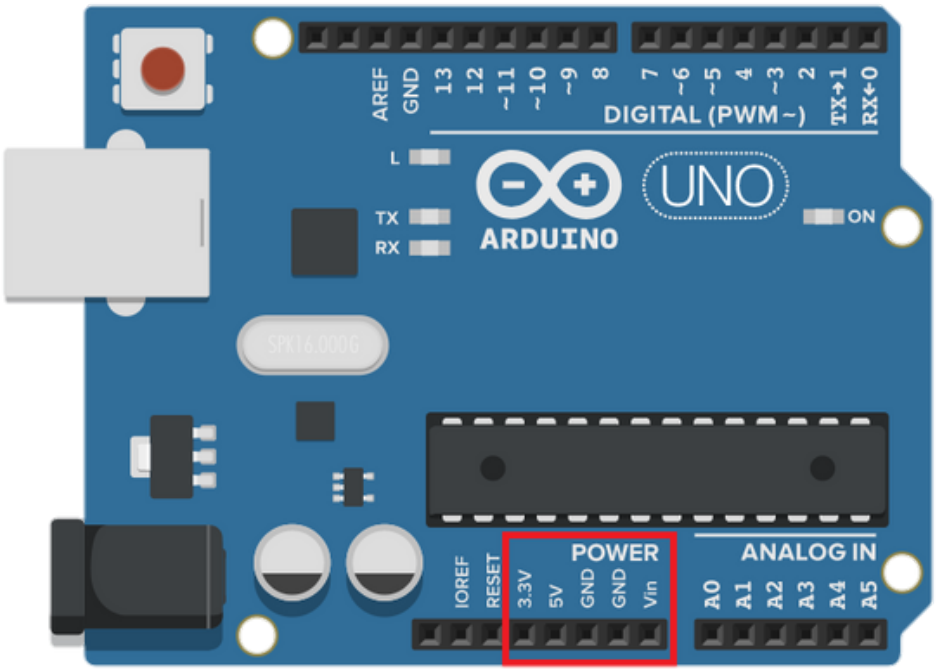
Portas Analógicas

Os pinos **A0** a **A5** são as **Portas Analógicas** e servem exclusivamente para a entrada de dados e são na maioria das vezes utilizadas para a comunicação com sensores.



Portas para alimentação

Essas são as portas destinadas para fornecer **energia** a circuitos e componentes externos.



Vin: A porta Vin fornece a mesma tensão que é recebida pelo pino de alimentação externo.

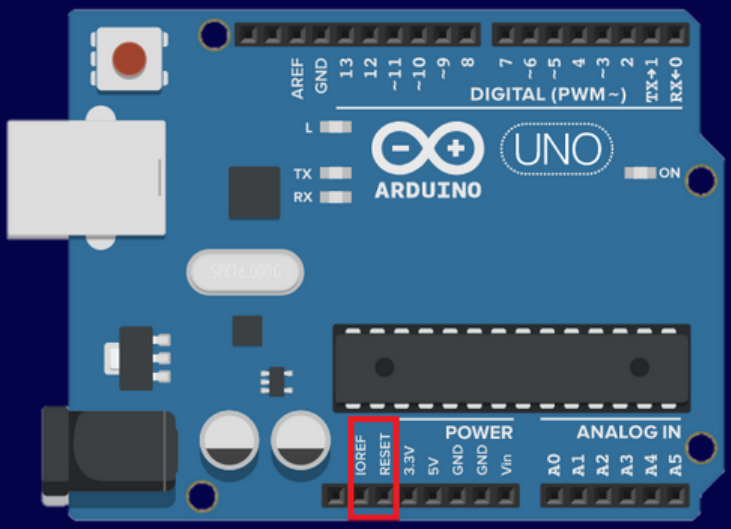
GND: É o terra que funciona como o negativo para todos os circuitos e dispositivos externos. O GND é comum a todas as portas, ou seja, ele é compartilhado, diferente das portas digitais.

5V: Porta que fornece 5 volts para a alimentação de dispositivos externos.

3,3V: Porta que fornece 3,3 volts para a alimentação de dispositivos externos.

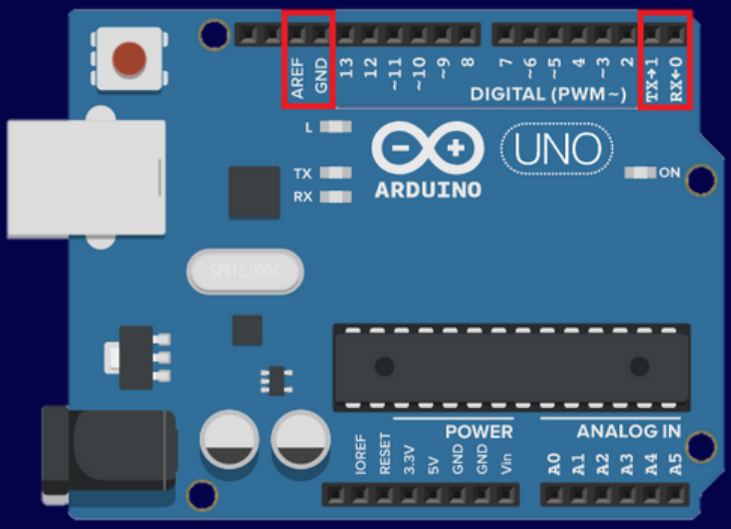
OBS: Os pinos que não possuem identificação não estão conectados a nada, então não devem ser utilizados.

Outras portas



RESET: Podemos utilizar esse pino para resetar o Arduino.

IOREF: Fornece a referência de tensão com que o Arduino está operando.



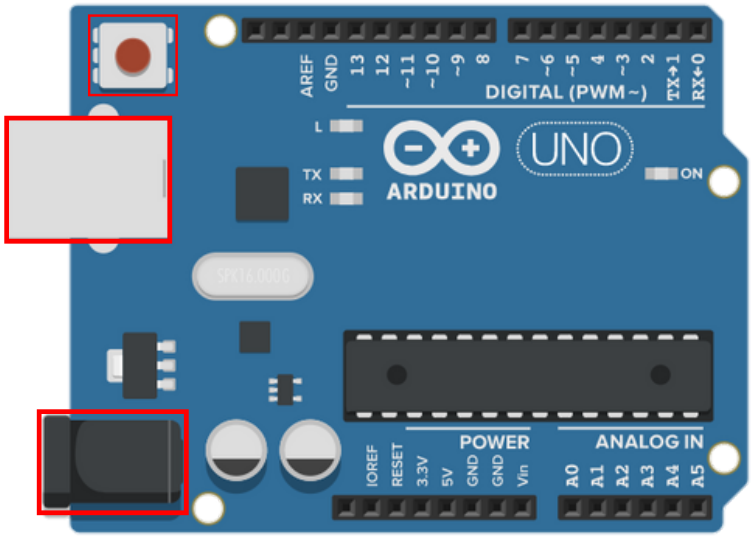
AREF: Este é o conversor analógico-digital (ADC) de referência de tensão. Pode ser usado em vez da referência padrão de 5V para a extremidade superior do espectro analógico.

RX: Receiver
TX: Transmitter

São as portas utilizadas pelo Arduino para fazer a **Comunicação Serial**. Você pode utilizar essas portas como portas digitais normais, porém é melhor evitar !!

- Botão para **Resetar** o Arduino.

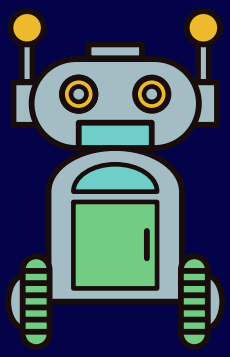
- **Porta USB**, é usada para conectar o Arduino no computador e enviar as programações para o processador, permite a conexão com a serial e também serve para a alimentação da placa.



Este é o **pino de Alimentação** Externa da placa, que é utilizado quando o Arduino não está usando a porta USB conectada ao computador. A tensão para alimentação deve ser entre **7** a **12V** para evitarmos problemas.

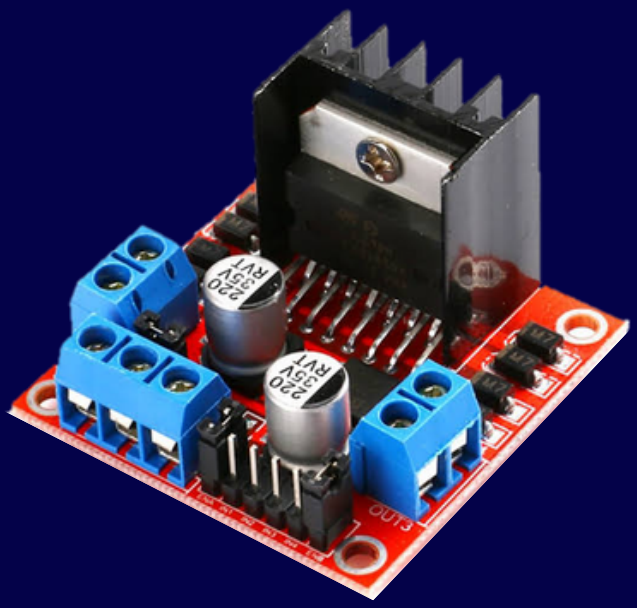
3 Controlando Motores

Finalmente chegamos em nosso primeiro projeto, nele, nós vamos controlar dois Motores DC utilizando uma Ponte H para entendermos o funcionamento do movimento do nosso robô, mas antes é preciso conhecer um pouco mais sobre os componentes que serão utilizados.



Ponte H L298N

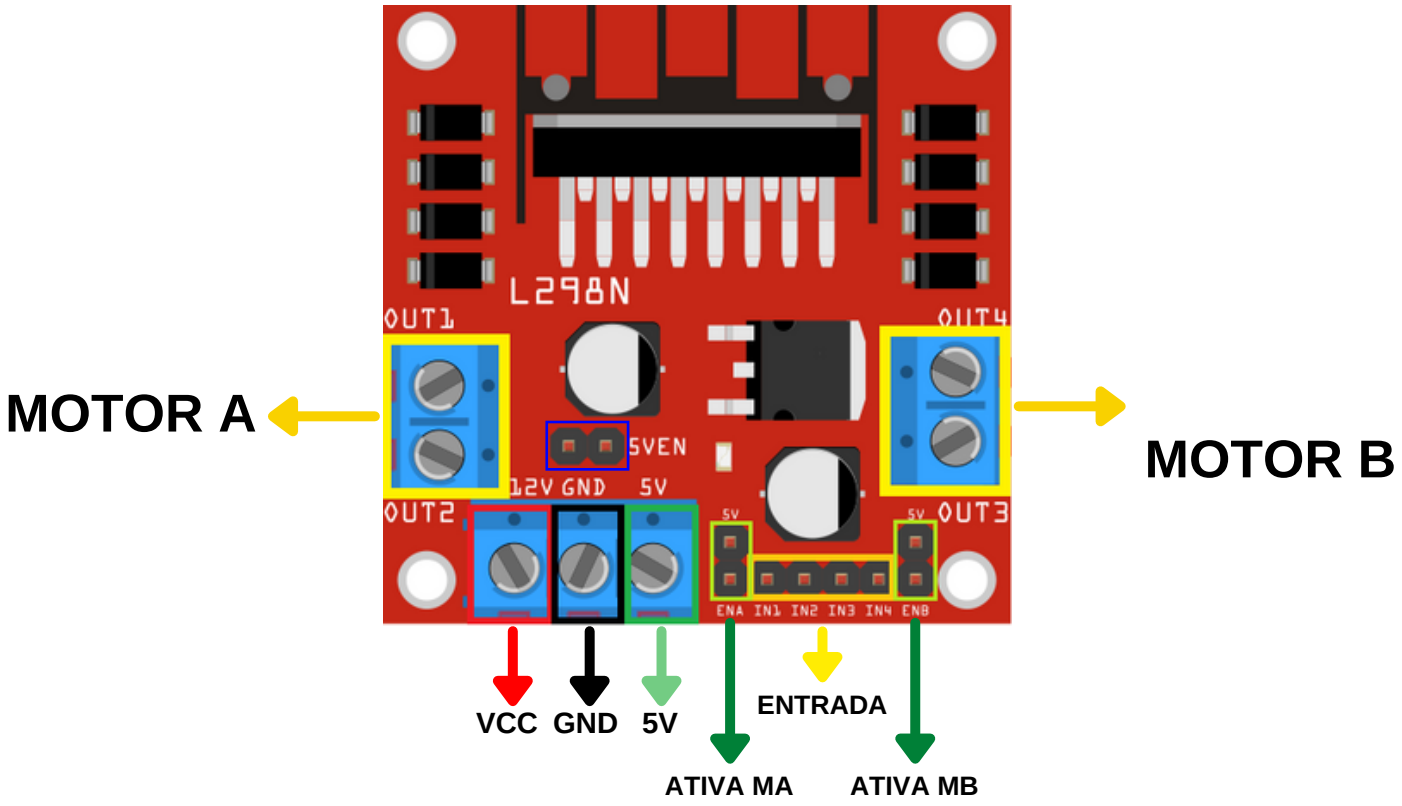
Na maioria dos projetos de **robótica** é necessário o uso de **motores DC** (corrente contínua) para realizar algum tipo de movimento. Grande parte desses **motores** são cargas indutivas que necessitam de uma quantidade de corrente superior à que as portas do Arduino conseguem fornecer, desse modo, não se deve ligar um motor que precisa de uma **corrente acima de 40 mA** nas portas digitais do Arduino, pois ele pode queimar a porta e danificar a placa. Para solucionar a questão da alta corrente é necessário o uso dos **transistores**. A Ponte H nada mais é que **4 transistores**. Este circuito é uma ótima solução por ser capaz de acionar simultaneamente dois motores controlando não apenas seus sentidos, como também as suas velocidades. Além do seu uso ser simples no Arduino.



Vantagens de usar ‼

O circuito integrado **L298N** é muito utilizado para controlar motores. Uma das vantagens do uso deste circuito é o menor espaço ocupado, a baixa complexidade do circuito e o fato de ele já possuir dois circuitos H, podendo assim, controlar dois motores. Outra vantagem do **L298N** é a resposta a sinais de **PWM**. Se forem utilizados sinais de **PWM**, é possível regular a tensão de saída, e dessa forma regular a velocidade dos motores.

Conhecendo o funcionamento...



Motor A e Motor B: Borne para conexão dos motores DC

VCC: Bornes responsáveis pela alimentação dos motores

GND: É o GND da placa, que deverá ser o mesmo GND do Arduino. Por isso, é importante interligar os GNDs sempre com fios.

5V: Em casos de não haver fonte de alimentação com mais de 6V podemos alimentar a placa com 5V por esta porta.

(Ativa MA) e (Ativa MB): são os pinos responsáveis pelo controle PWM dos motores A e B. Se estiver com jumper, não haverá controle de velocidade.

(IN1 e IN2) e (IN3 e IN4): São utilizados para controlar o sentido dos motores.

Ativa 5V: É o Jumper responsável por regular a alimentação da **Ponte H**, caso o jumper esteja conectado, a alimentação deve ser de **6** a **12V**. Se o jumper está desconectado, é preciso utilizar a porta que fornece **5V** do Arduino, porém é necessário se atentar a **tensão** do motor utilizado e a **corrente**, **caso contrário o Arduino pode queimar**.

Controle dos motores

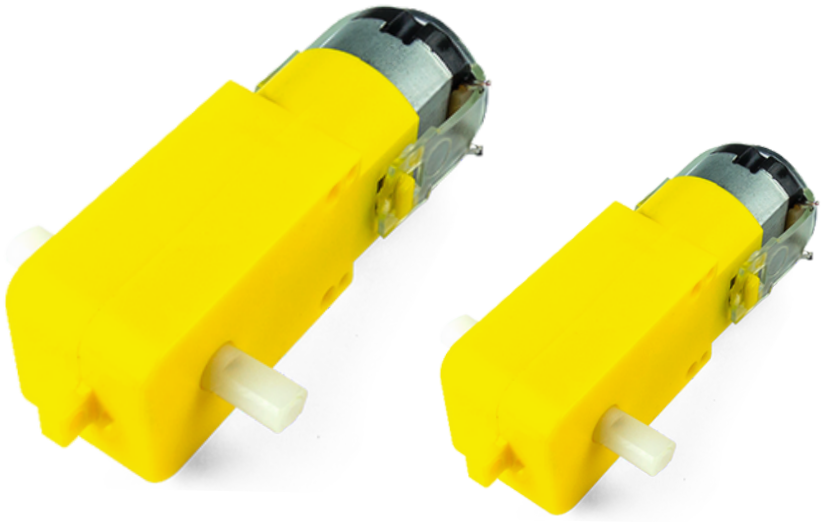
Para o **controle dos motores**, é necessário acionar os pinos da seguinte maneira:

Motor A	IN1	IN2
Horário	High (5V)	Low (GND)
Anti Horário	Low (GND)	High (5V)
Freio	Low (GND)	Low (GND)
Freio	High (5V)	High (5V)

A mesma forma se aplica para o motor **B**, **IN3** e **IN4**.

Motores DC

Um **Motor DC** é um motor que converte a energia elétrica de **Corrente Contínua** em energia mecânica. Eles são motores que possuem imãs permanentes ou então têm campo e armadura, neste caso não possuem ímãs permanentes. Os motores de corrente contínua são muito usados e possuem diversas aplicações como por exemplo, brinquedos, eletrodomésticos, máquinas industriais, veículos elétricos, entre outros.

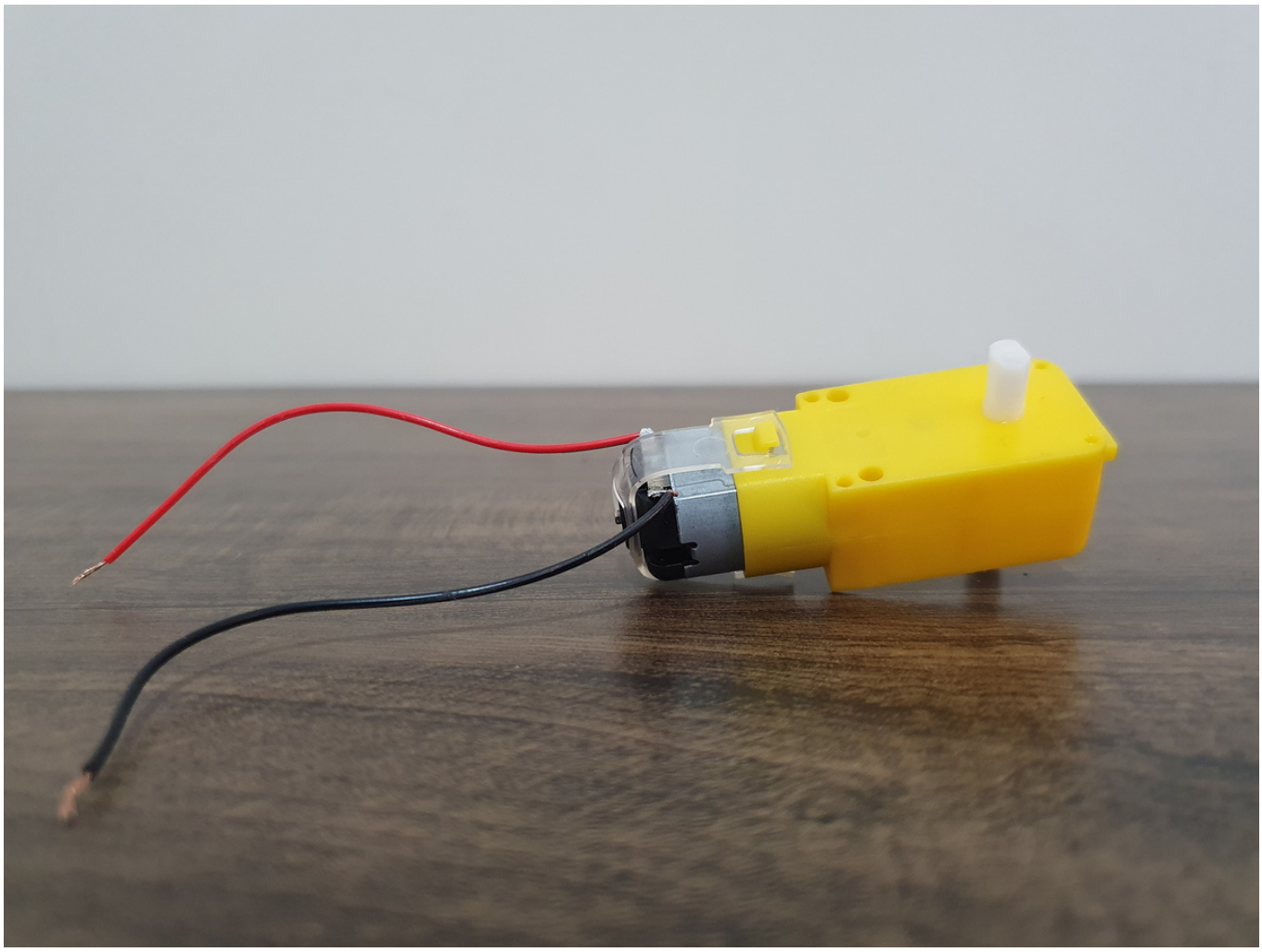


Jumper

Os **Jumpers** nada mais são do que condutores para ligar um ponto a outro.

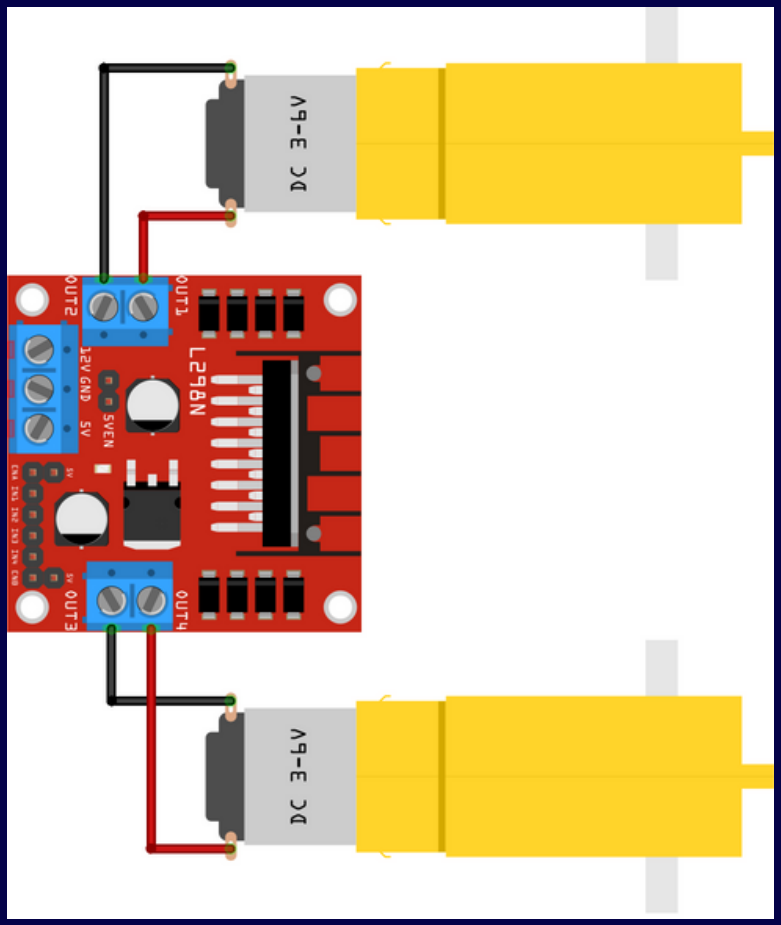
Soldando Motores

É importante saber que os **Motores DC** não vem com os fios soldados. Desse modo, é necessário soldar os fios flexíveis nos motores e para isso é preciso utilizar um **ferro de solda** e **estanho**.



Caso você seja **menor de idade**, peça ajuda a um **adulto** para a realização da solda dos fios nos motores !!

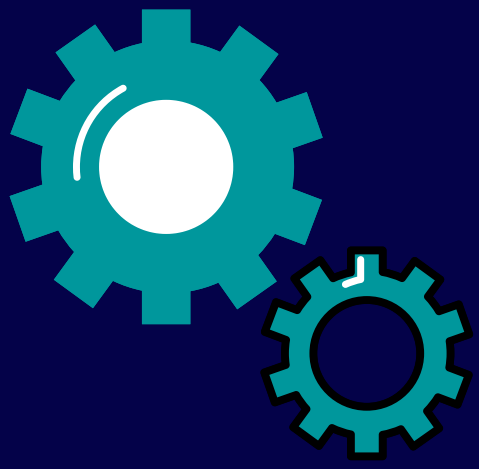
Ligação dos motores



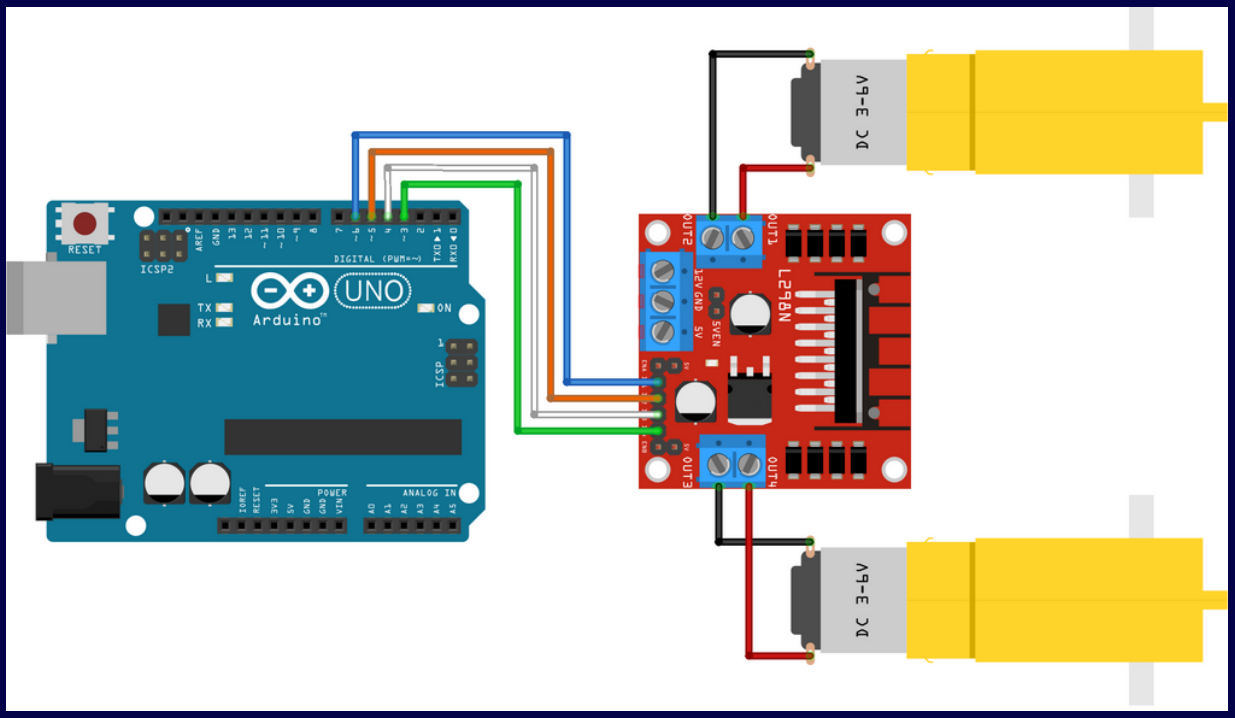
Agora que vimos todos os componentes que utilizaremos, vamos a montagem. Para que os motores girem no **sentido correto**, é necessário ligar ambos da mesma maneira, dessa forma temos:

Positivo Motor 1: **OUT1**
Negativo Motor 1: **OUT2**

Positivo Motor 2: **OUT4**
Negativo Motor 3: **OUT3**



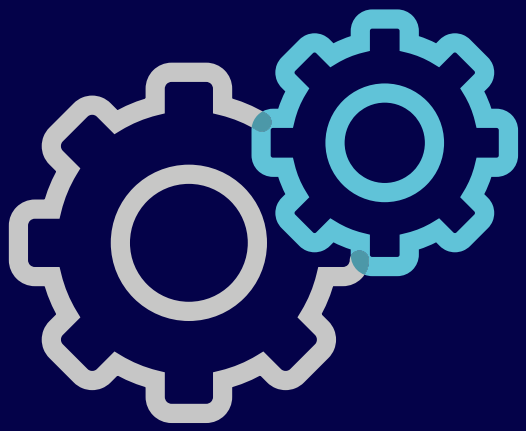
Ligação das entradas no Arduino



Para conseguirmos controlar nossos motores, é necessário ligar os pinos que os controlam nas portas digitais do Arduino, através da **Programação**, nossa placa enviará as instruções para a **Ponte H**.

Desse modo, temos:

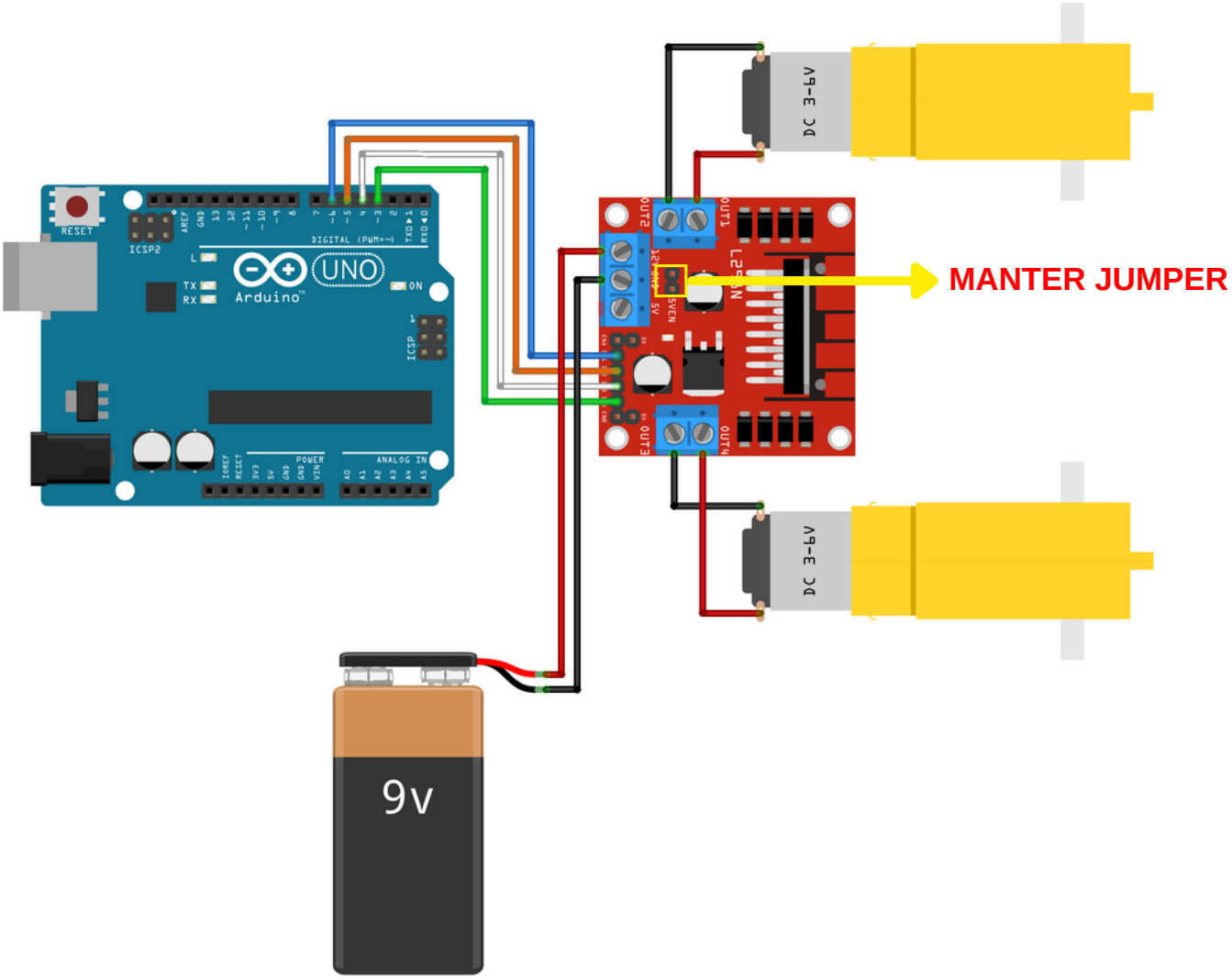
IN1: Porta 6
IN2: Porta 5
IN3: Porta 4
IN4: Porta 3



OBS: Os pinos **ENA** e **ENB** não vão ser utilizados agora, não vamos controlar a velocidade do motor nesse momento, somente o sentido de rotação !!

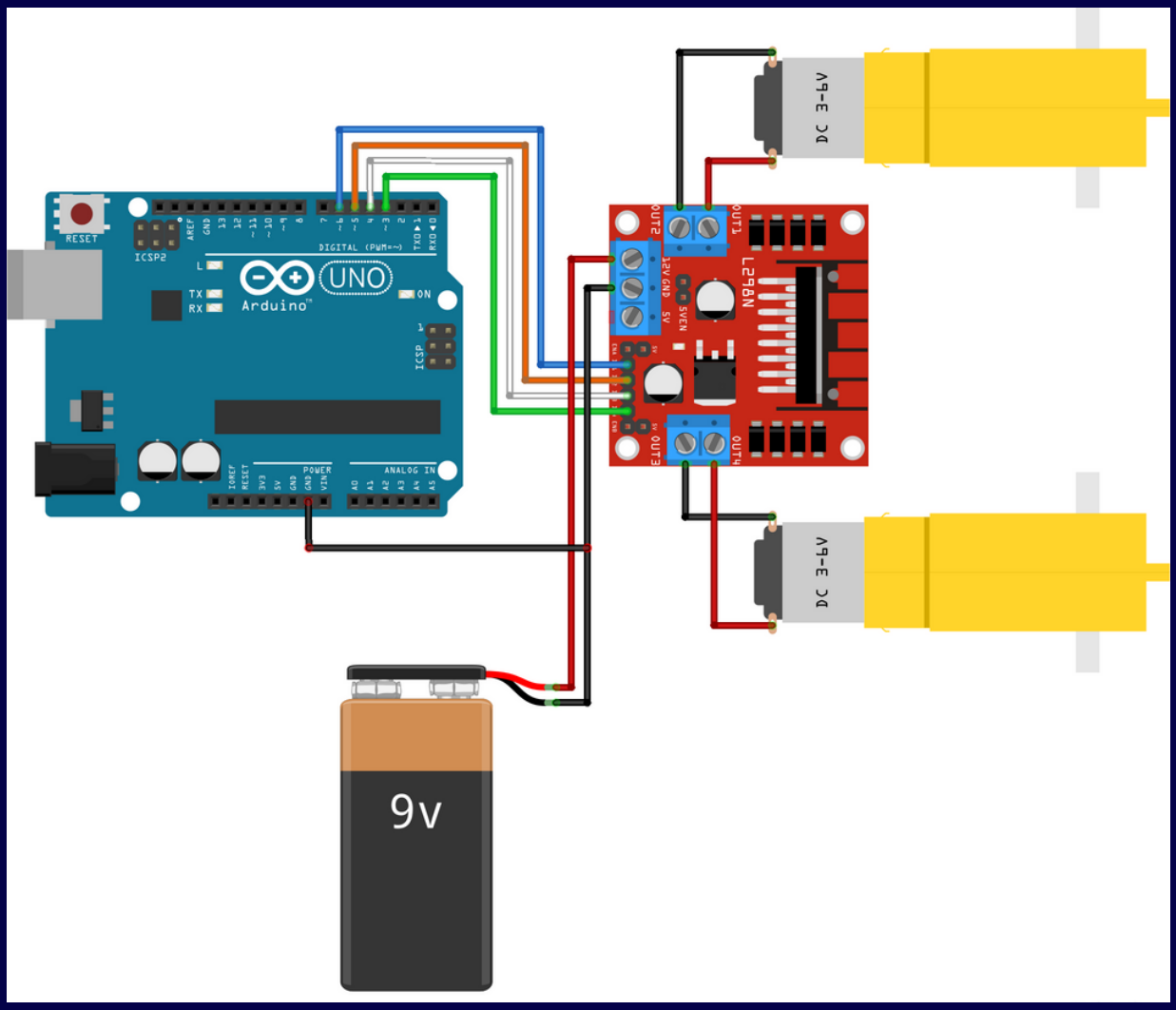
Alimentação do circuito

Para alimentarmos nossa **Ponte H** corretamente, vamos utilizar uma bateria de **9V**, ligar o negativo no **GND** e o positivo no **VCC**.



OBS: É necessário conectar o jumper, já que não vamos utilizar o 5V do Arduino.

Atenção: Não se esqueça de conectar o negativo do Arduino na **Ponte H** como pode ser visto na imagem abaixo.



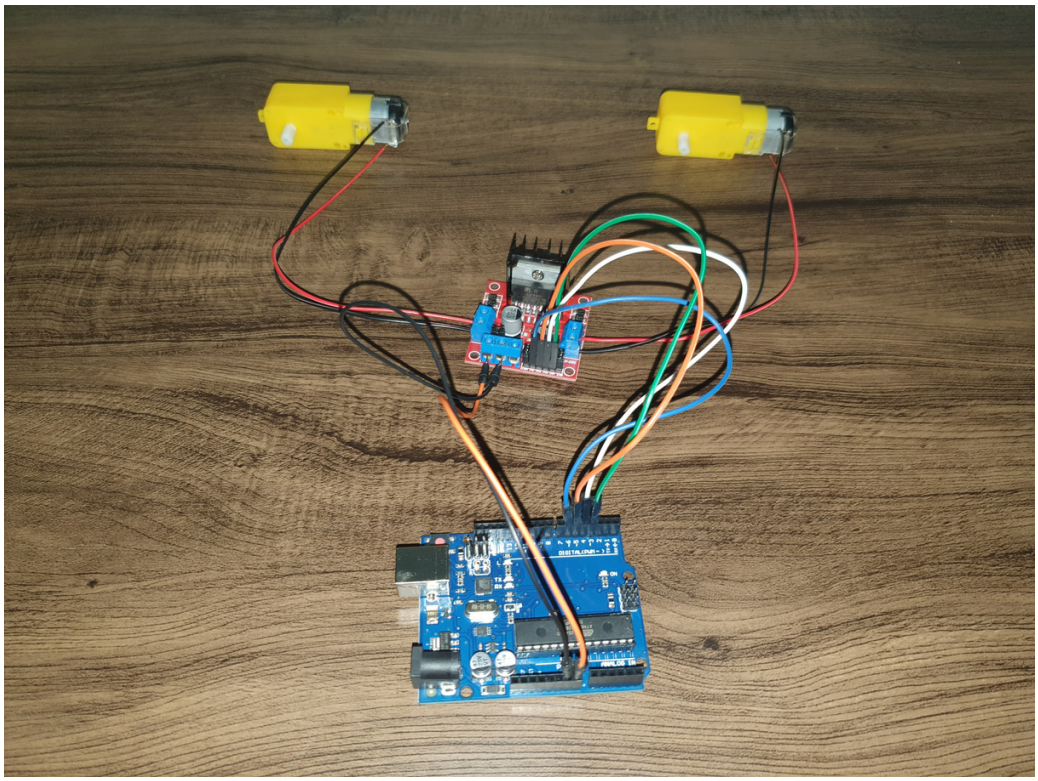
Alimentação do circuito

Também existem outras formas para alimentarmos nossa **Ponte H**, caso você possua uma fonte de alimentação de no mínimo **7** e no máximo **12V** é possível alimentar o Arduino e utilizar a porta **Vin** para alimentar a **Ponte H.**

Lembrando: A porta **Vin** fornece a mesma tensão que é recebida pelo pino de alimentação externo, ou seja, se a fonte for de **9V**, a saída no pino **Vin** será **9V**, por isso é necessário manter o jumper **Ativa 5V** conectado.

Nossos **motores DC** suportam de **3** a **6** **volts**, porém eles **não funcionam** corretamente se usarmos a porta que fornece **5V do Arduino**, dessa forma, não é recomendado o uso da porta para esses motores.

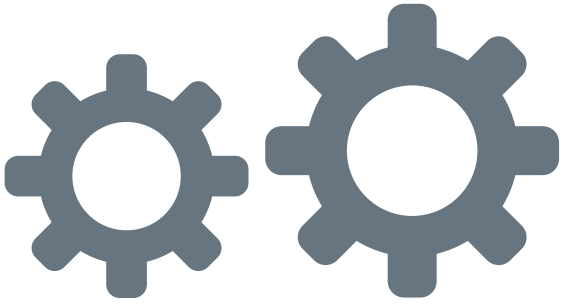
Além das fontes de alimentação, podemos utilizar a própria bateria para alimentação do **Arduino** e a **Ponte H**, basta utilizar o **adaptador**, plugá-lo no pino de alimentação externa do Arduino e utilizar o pino **Vin** para alimentação da Ponte H da mesma maneira que a fonte. Confira a a montagem na prática abaixo:



OBS: Em todos os exemplos de alimentação, o Jumper **Ativa 5V** deve estar **conectado !!**

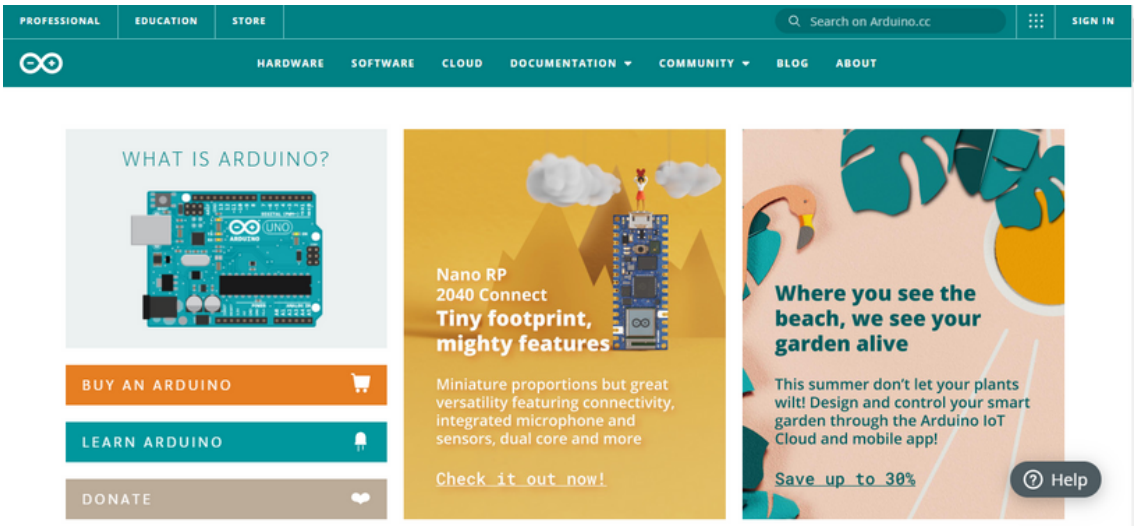
Programação

Agora chegou a hora de programarmos nosso Arduino, para isso é necessários instalarmos a **IDE do Arduino**, **Clique Aqui** para baixar. Caso tenha dúvidas, siga o passo a passo abaixo.



Instalando a IDE do Arduino

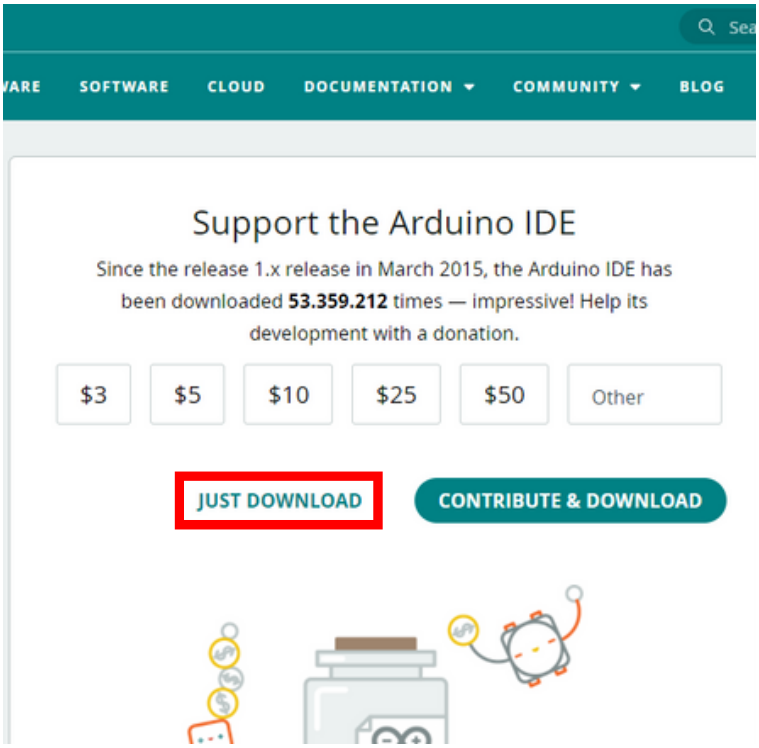
1 Acesse o site oficial do Arduino Clicando aqui.



2 Clique em **Software**.

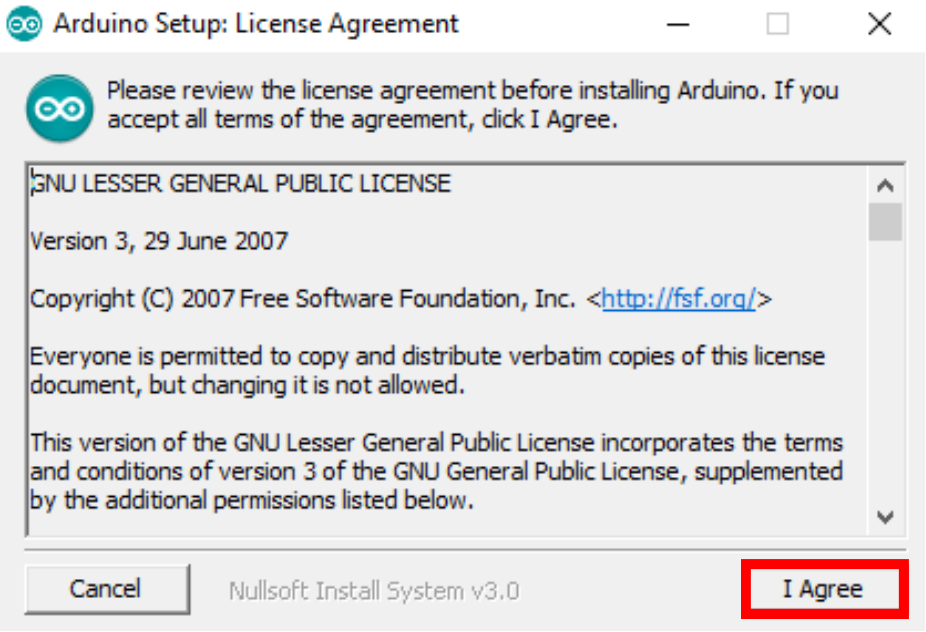
Aqui é onde vamos fazer o Download da **IDE do Arduino**. Selecione qual sistema operacional você utiliza.

Caso possua **Windows 7** ou uma **versão superior**, selecione a primeira opção: **Windows Win 7 and newer**.

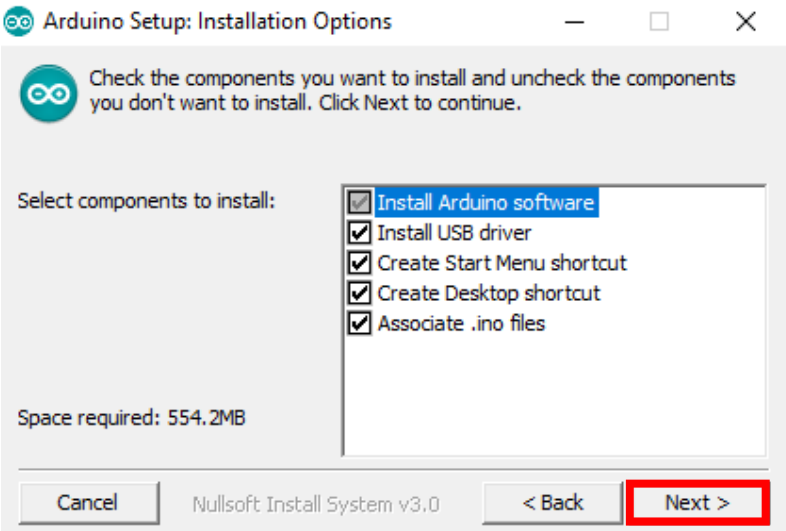


Se quiser, você pode fazer uma doação para ajudar o desenvolvimento da **Arduino IDE**. Caso contrário, basta clicar em **Just Download** e o programa começará a baixar.

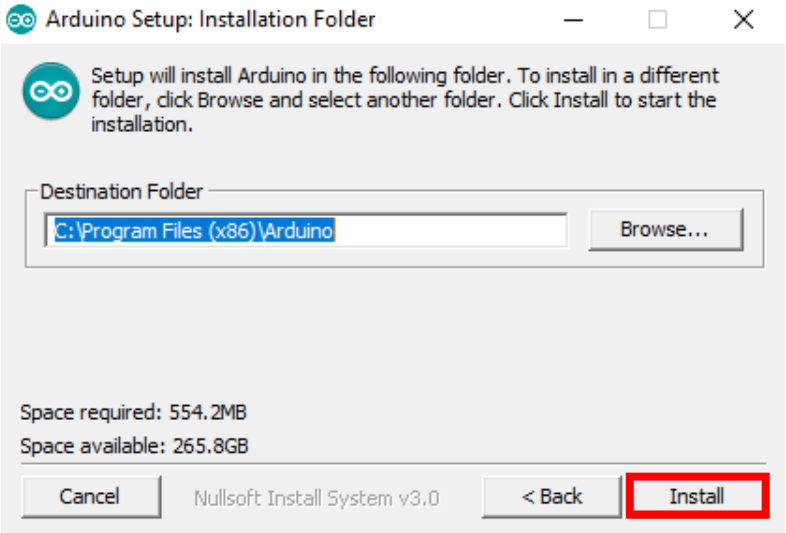
3 Após o Download do instalador da IDE terminar, **clique duas vezes sobre o ícone** gerado para abri-lo.



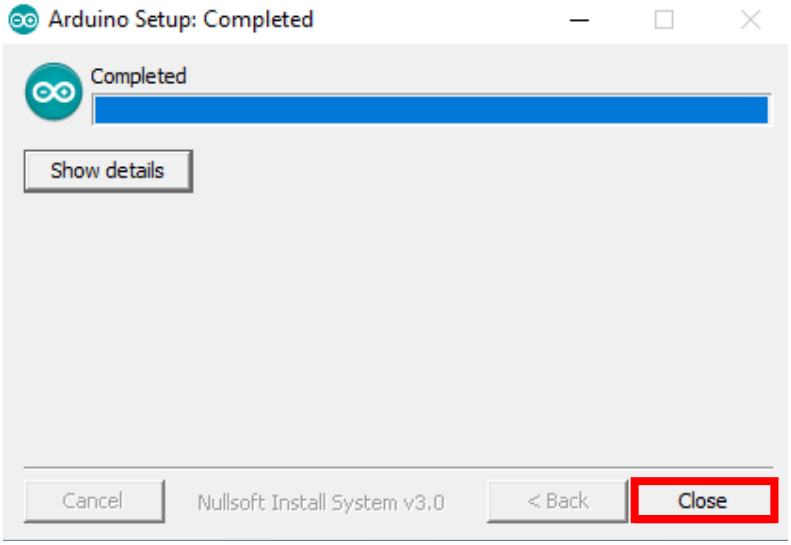
Clique em **I Agree**.



Selecione todas as caixas e clique em **Next**.



Escolha o local onde a IDE vai ser instalada em seu computador e clique em **Install**. A instalação começará.



Quando a instalação for concluída clique em **Close**.

Pronto, agora é só programar! Abra a IDE do Arduino e digite o código abaixo ou baixe o código **clicando aqui!**

Programação

```
// declarando as portas onde os motores estão conectado

int IN1 = 6;
int IN2 = 5;
int IN3 = 4;
int IN4 = 3;

void setup()
{
  //Definindo moto
 pinMode(IN1, OUTPUT);
 pinMode(IN2, OUTPUT);
 pinMode(IN3, OUTPUT);
 pinMode(IN4, OUTPUT);
}
void loop()
{

//comando para girar o Motor A no sentido horário
//OBS: Lembre da tabela da Ponte H !!

 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, LOW);
 delay(2000); // espera 2 segundos

 // Comando para freiar o motor A
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, HIGH);
 delay(1000); // espera 1 segundo

 //Comando para girar o motor B no sentido horário
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, LOW);
 delay(2000); // delay de 2 segundos
```

```
// Comanda para freiar o Motor B

digitalWrite(IN3, HIGH);
digitalWrite(IN4, HIGH);
delay(1000); // delay de 1 segundo

//Comando para girar o Motor A no sentido anti-horário

digitalWrite(IN1, LOW);
digitalWrite(IN2, HIGH);
delay(2000); // delay de 2 segundos

//Comando para freiar o motor A

digitalWrite(IN1, HIGH);
digitalWrite(IN2, HIGH);
delay(1000); // delay de 1 segundo

//Comando para girar o motor B no sentido anti-horário

digitalWrite(IN3, LOW);
digitalWrite(IN4, HIGH);
delay(2000); // delay de 2 segundos

//Freia motor B

digitalWrite(IN3, HIGH);
digitalWrite(IN4, HIGH);
delay(1000); //delay de 1 segundo
}
```

Se você quer baixar essa a programação, **Clique Aqui**!

Quer ver o funcionamento na prática ?? **Clique aqui** e acesse !!

4 Sensor de Estacionamento

Nesse próximo projeto, vamos conhecer outros componentes importantes para construirmos nosso **robô**, vamos conhecer a fundo o **Sensor Ultrassônico** e desenvolver um projeto semelhante aos sensores de estacionamento utilizados nos carros.

Vamos conhecer os componentes:

Sensor Ultrassônico HC-SR04

A função desse sensor é **medir distâncias** com grande precisão. Seu funcionamento é semelhante ao sonar dos morcegos, tem como princípio de funcionamento a emissão de uma onda sonora de alta frequência. O objeto a ser detectado resulta em um eco, que é convertido em sinais elétricos. A detecção do eco depende da intensidade e da distância entre o objeto e o sensor. É a partir disso que sabemos se o objeto está dentro dos parâmetros estabelecidos, ou se está no local para o sensor identificá-lo. Esse sensor tem diversas aplicações, como por exemplo:

- **Registrar a posição dos objetos;**
- **Realizar a medição de distâncias;**
- **Medir níveis;**
- **Contar objetos.**

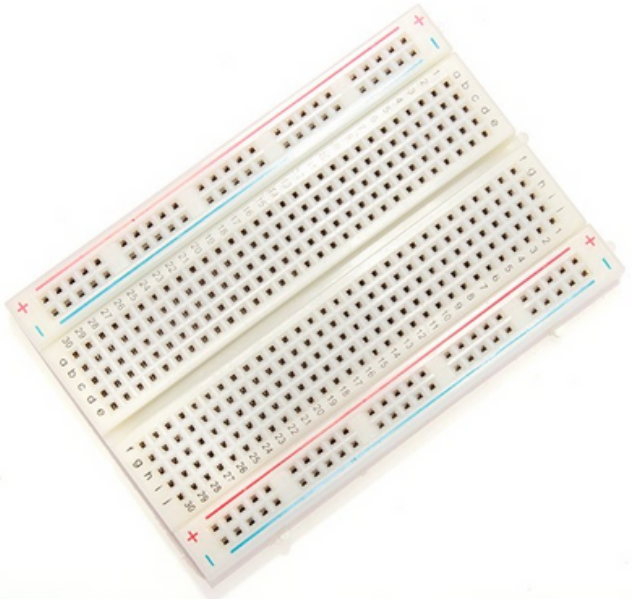
O **Sensor de Distância Ultrassônico HC-SR04** é capaz de medir distâncias de **2cm** a **4m** com ótima precisão. Este módulo possui um circuito pronto com emissor e receptor acoplados e 4 pinos (**VCC**, **Trigger**, **ECHO**, **GND**) para realizar a medição.

Para começar a medição é necessário alimentar o módulo e colocar o pino **Trigger** em nível alto por mais de 10s. Assim o sensor emitirá uma onda sonora que ao encontrar um obstáculo rebaterá de volta em direção ao módulo, sendo que o neste tempo de emissão e recebimento do sinal o pino **ECHO** ficará em nível alto. Logo o cálculo da distância pode ser feito de acordo com o tempo em que o pino **ECHO** permaneceu em nível alto após o pino **Trigger** ter sido colocado em nível alto.

Cálculo: Distância = $\frac{[\text{Tempo ECHO em nível alto} * \text{Velocidade do Som}]}{2}$

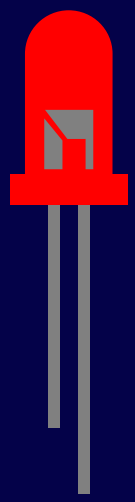
Protoboard

Nada mais é do que uma placa com furos e conexões condutoras utilizada para a montagem de protótipos e projetos que estão na fase inicial. A vantagem de se utilizar uma **Protoboard** é que podemos montar os componentes direto nela, assim eliminamos a necessidade de utilizar a solda.



LED

O LED é um diodo emissor de luz, ele é um componente capaz de emitir luz visível transformando a energia elétrica em energia luminosa.



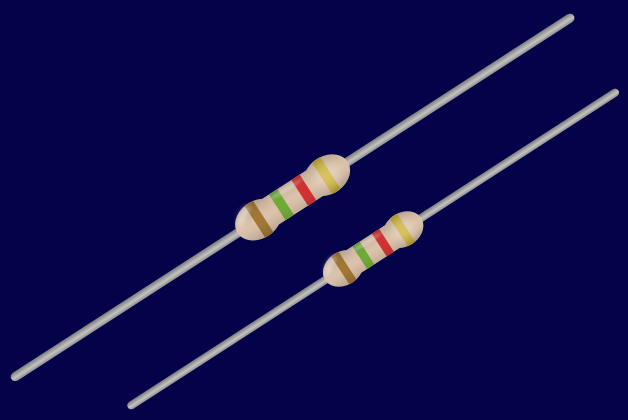
A maior perna do LED é a parte **Positiva** e a menor **Negativa** !!

- **Ânodo**: Positivo
- **Cátodo**: Negativo

Resistores

Resistores são dispositivos muito utilizados para controlar a passagem de corrente elétrica em circuitos elétricos por meio do **efeito Joule**, que converte **energia elétrica** em **energia térmica**.

Vamos utilizar ele pois se ligarmos um LED direto no 5 **volts** fornecido pelo Arduino, ele irá **queimar** !!



Como saber qual a resistência correta ??

Existe uma **fórmula** para saber qual a **resistência** deve ser utilizada em um **LED**.

Mas antes é preciso saber qual é a **Tensão** e a **Corrente** de cada **LED**.

LEDs		
Cor do LED	**Tensão em Volts (V)**	**Corrente em Miliamperes (mA)**
Vermelho	1,8 - 2,0V	20 mA
Amarelo	1,8 - 2,0V	20 mA
Laranja	1,8 - 2,0V	20 mA
Verde	2,0 - 2,5V	20 mA
Azul	2,5 - 3,0V	20 mA
Branco	2,5 - 3,0V	20 mA

Cálculos

Para se calcular o **resistor** adequado para o **LED** utilizaremos a seguinte fórmula:

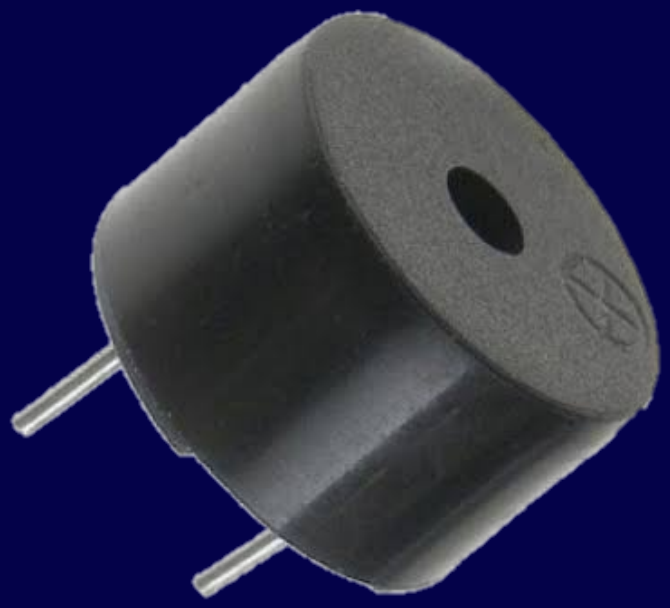
$$\textbf{Resistência do LED} = \boxed{\frac{\text{Tensão de Alimentação - Tensão do LED}}{\text{Corrente do LED}}}$$

OBS: O valor do resistor pode ser acima do resultado, porém nunca abaixo, por exemplo, se o valor for igual a **150 Ω**, podemos usar um resistor de valor mais alto, como **200 Ω**, ou até mesmo **1KΩ**, que são 1000 Ω.

OBS: O símbolo **Ω** significa **Ohms**, que é a nossa unidade de resistência elétrica.

Buzzer

O Buzzer 5V é um componente para adicionar efeitos sonoros em projetos eletrônicos como por exemplo, alarmes, sistemas de sinalização, jogos, brinquedos, etc. O Buzzer do tipo Ativo contém um circuito oscilador embutido, dessa forma basta você energizar o componente para que o mesmo comece a emitir som.



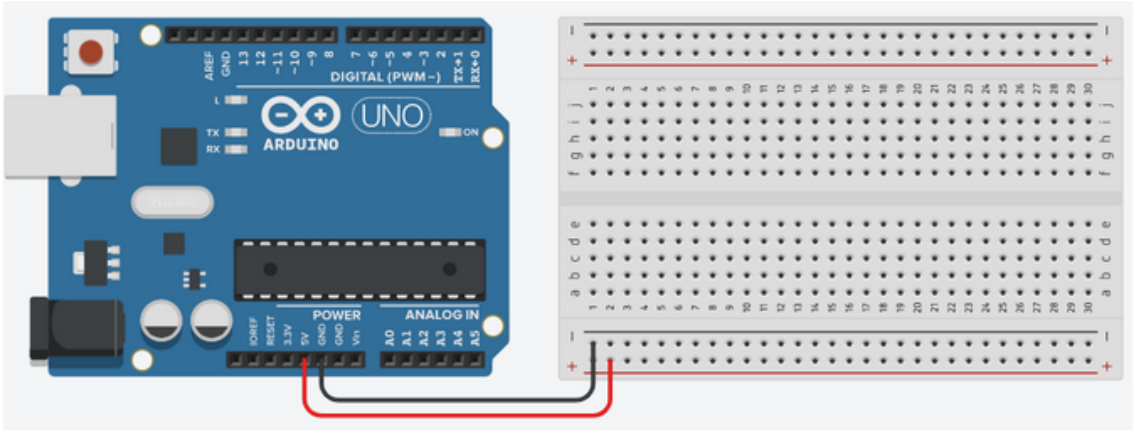
OBS: A maior perna do Buzzer é positiva.

Montagem do Circuito

Agora que conhecemos os componentes, vamos iniciar a montagem do nosso circuito, lembrando que esse projeto funcionará semelhante a um sensor de estacionamento.

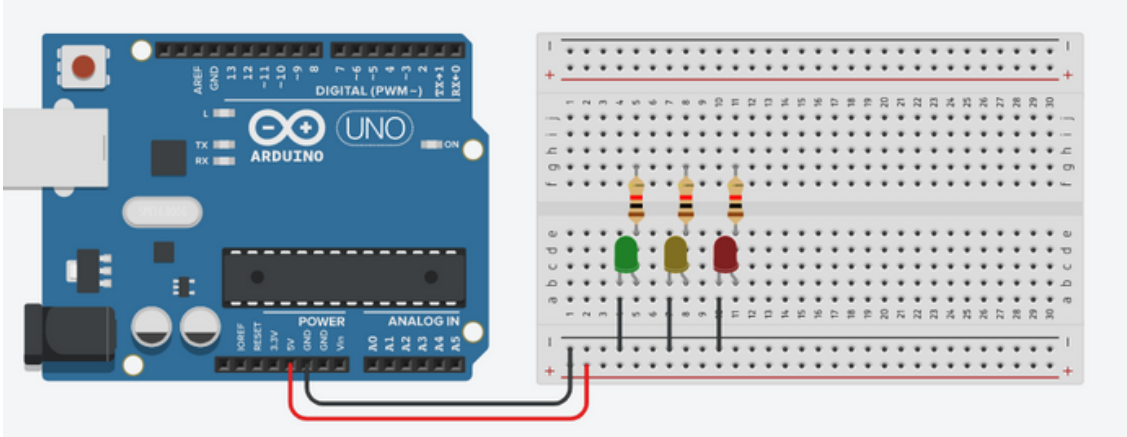
Alimentando o Circuito

O primeiro passo para montarmos nosso circuito é alimentar a nossa **Protoboard** com os pinos **5V** e **GND** do Arduino.



LEDs e Resistores

Agora, vamos colocar nossos **LEDs** e os **Resistores** na **Protoboard**.

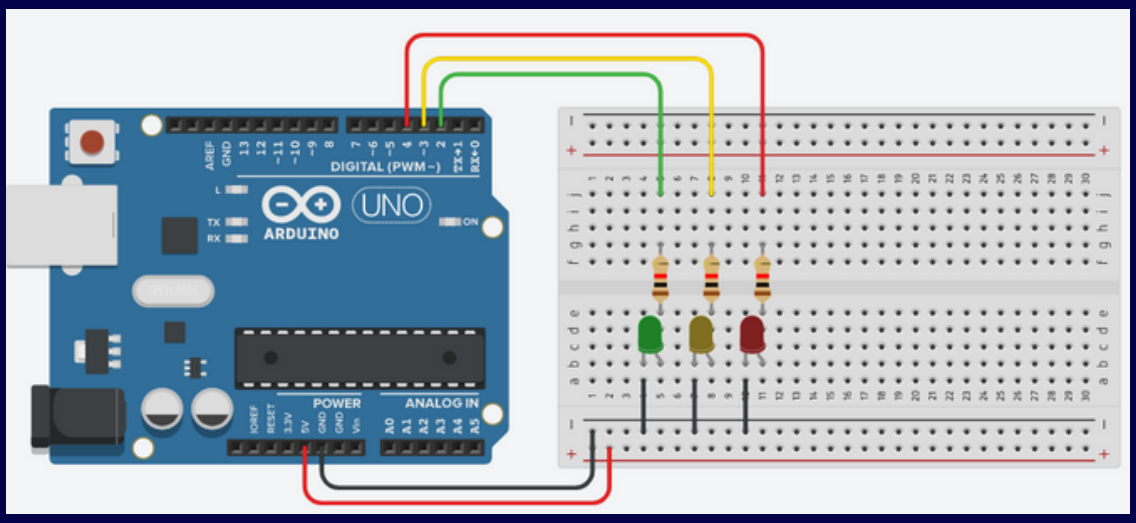


Lembrando: A maior perna do **LED** é o **positivo** e a **menor** o **negativo** !!

OBS: Não faz diferença se conectarmos os resistores no positivo ou negativo dos LEDs, caso conectado no negativo, basta ligar o positivo do **LED** no **5V**, o resistor no negativo e na porta digital.

Conectando os LEDs

Vamos conectar os **Resistores** que estão no positivo dos **LEDs** nas **portas digitais** do Arduino.

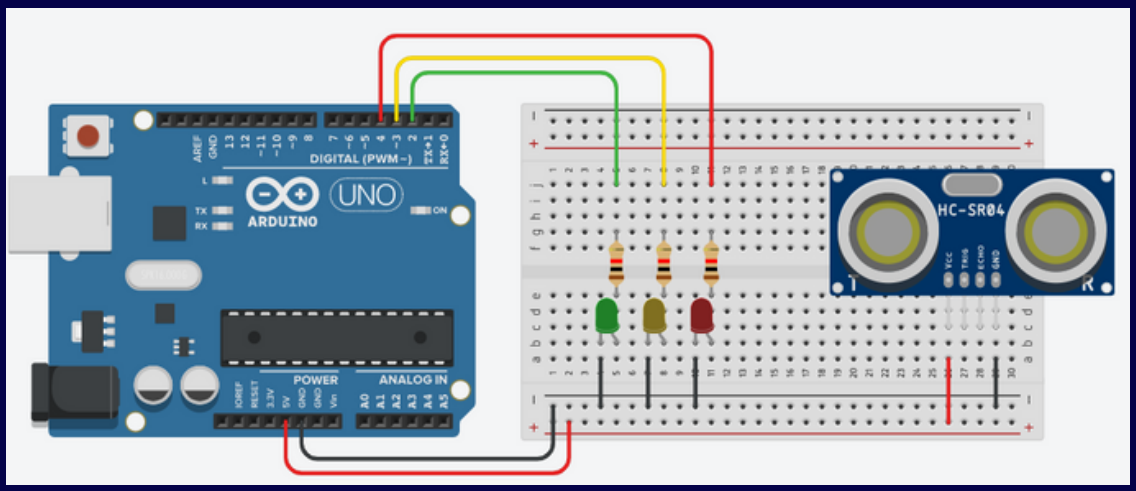


Assim temos:

LED Verde: Porta 2
LED Amarelo: Porta 3
LED Vermelho: Porta 4

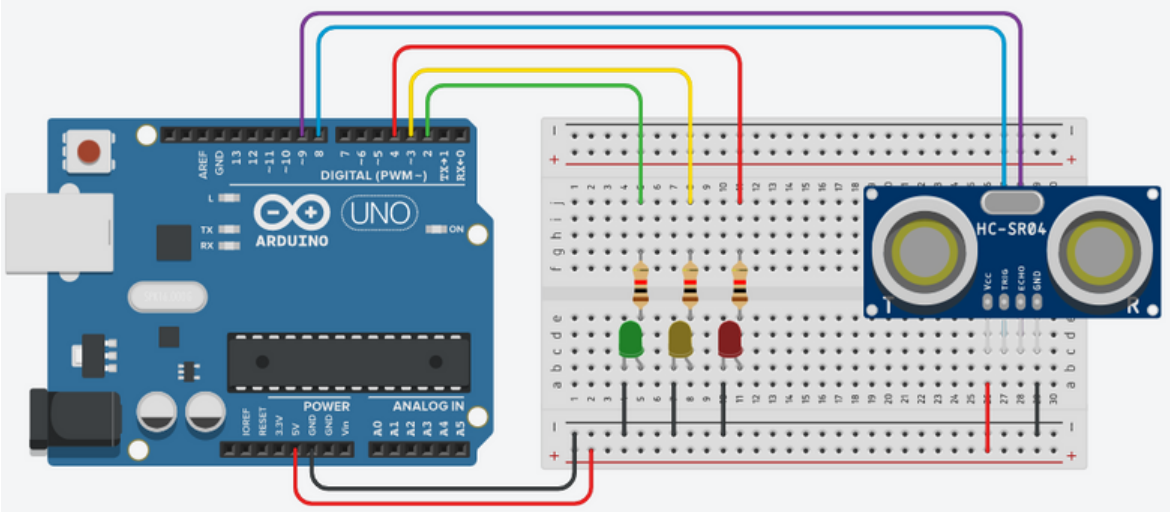
Alimentação Ultrassônico

Conecte o Sensor Ultrassônico na **Protoboard** e ligue o pino **VCC** nos **5V** e o pino **GND** no **negativo**.



Conectando no Arduino

Conecte o Sensor Ultrassônico na **Protoboard** e ligue o pino **VCC** nos **5V** e o pino **GND** no **negativo**.

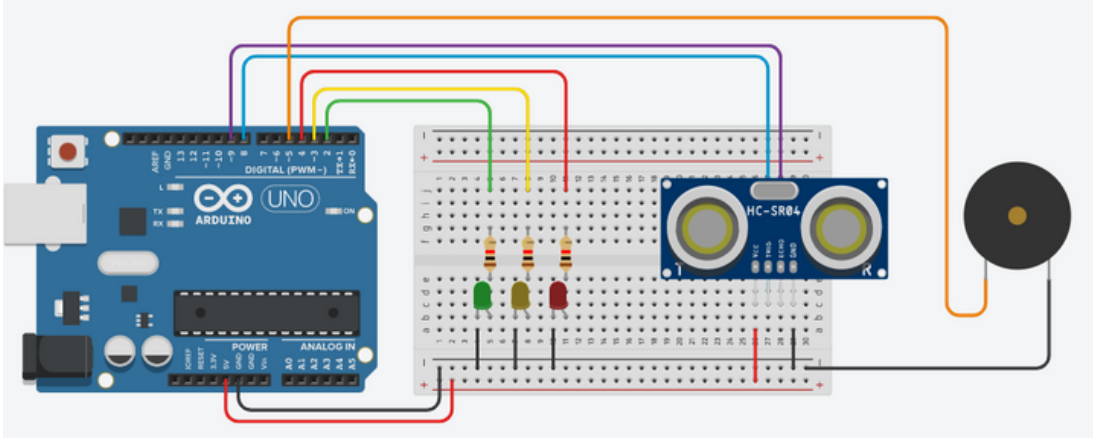


Dessa forma temos:

Trig: Porta 5
Echo: Porta 6

Conectando Buzzer

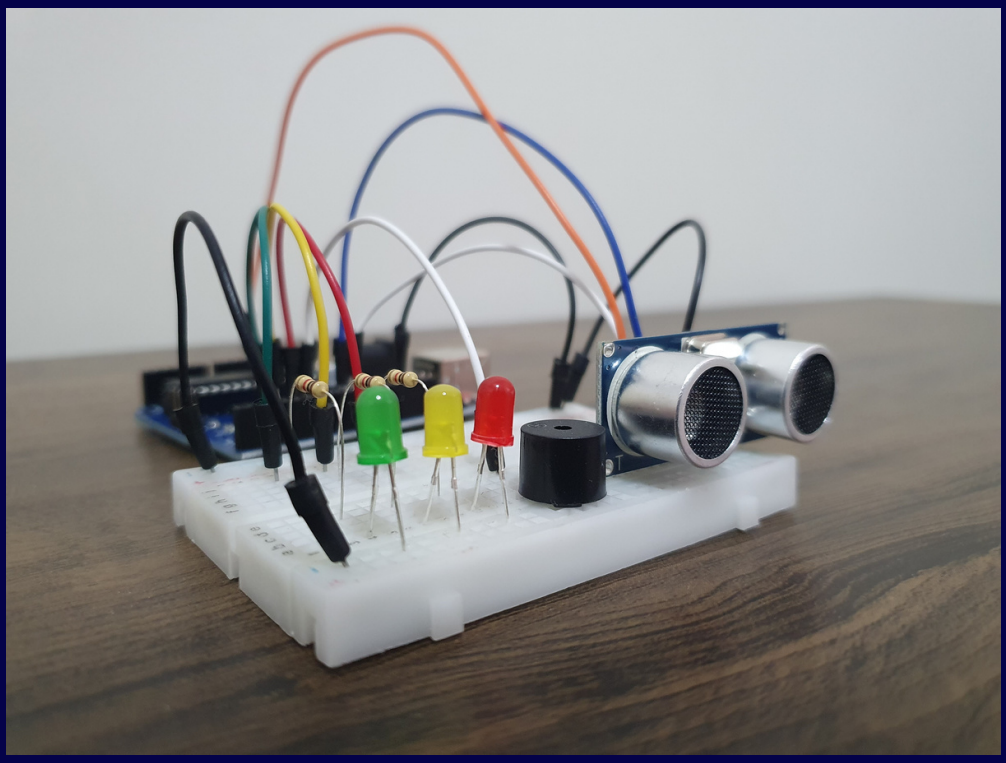
Por fim, vamos adicionar nosso **Buzzer** ao circuito para termos **efeitos sonoros**.

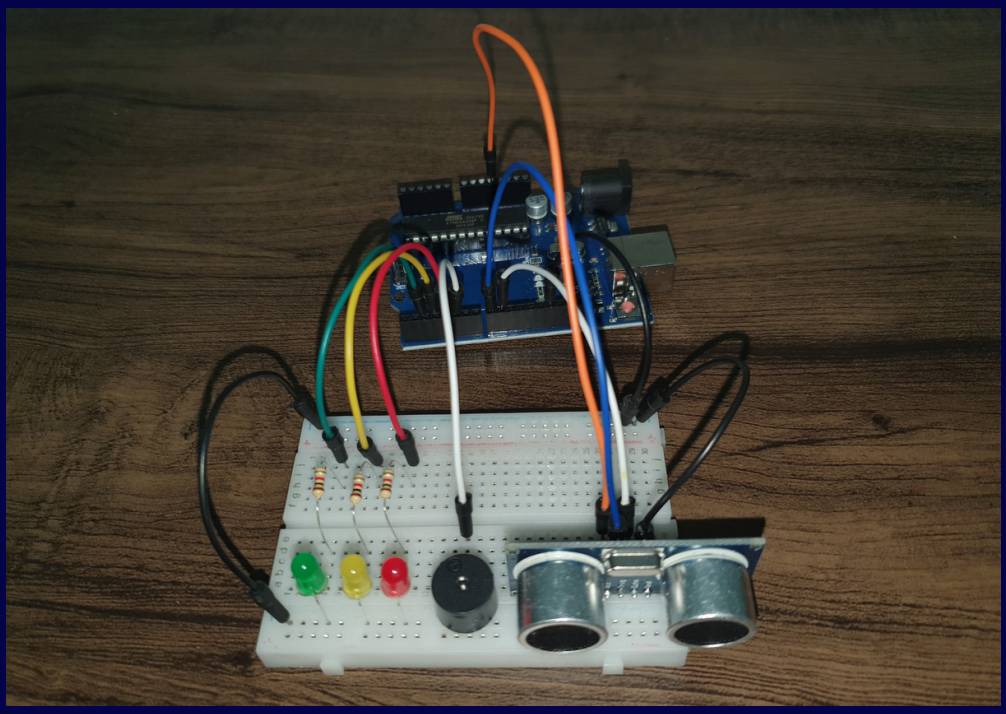


Positivo: Porta 7
Negativo: GND

OBS: A maior perna do **Buzzer** é **positiva**.

Montagem na Prática

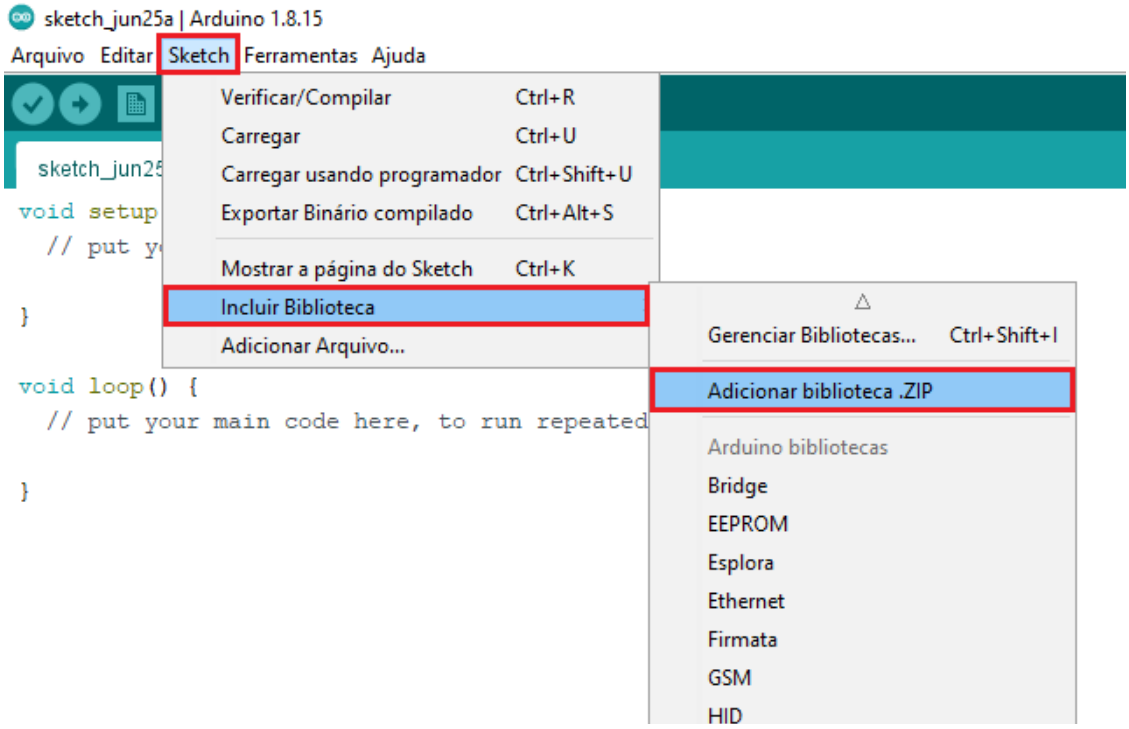




incluindo Biblioteca

Para conseguirmos programar nosso sensor, é necessário adicionarmos a biblioteca “**Ultrasonic.h**”, você pode baixá-la **Clicando Aqui.**

Para Adicionar a **biblioteca** no Arduino, **siga o passo a passo abaixo:**

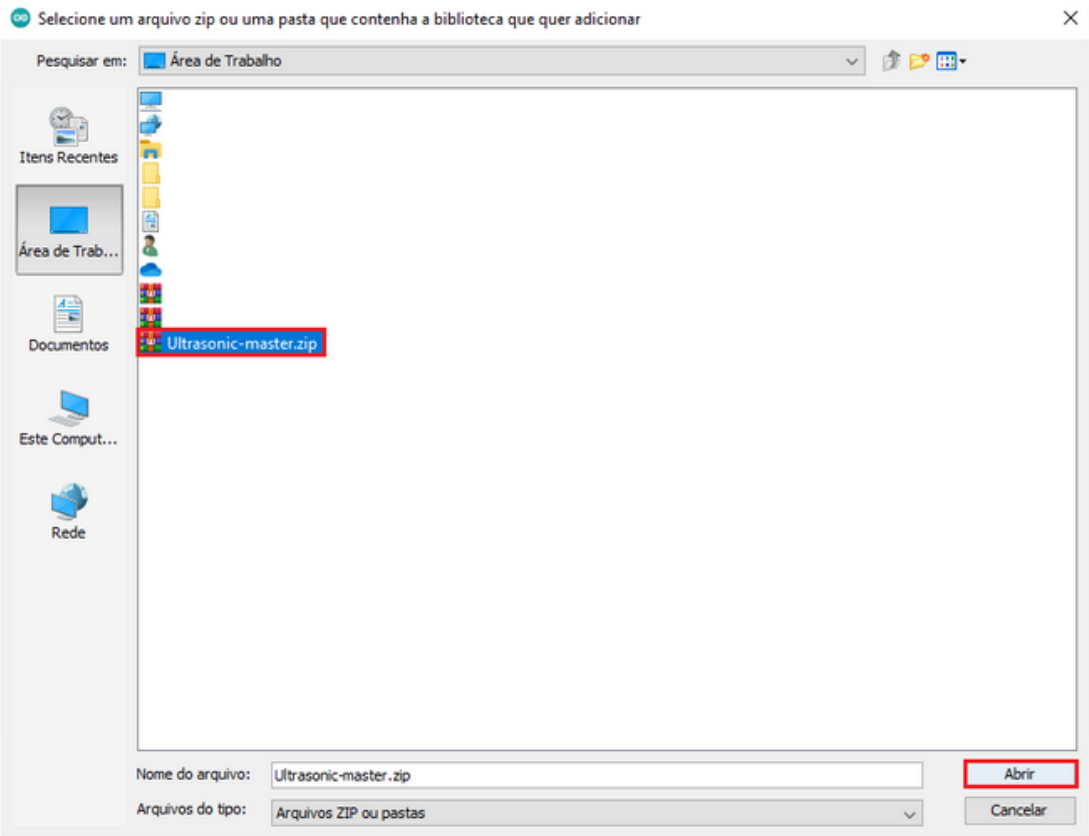


1- Abra a **IDE do Arduino**

2- Clique em **Sketch**

3- Selecione **Incluir Biblioteca**

4- Clique em **Adicionar Biblioteca.zip**



Selecione o arquivo **.zip** e clique em **abrir**.

Pronto, a biblioteca necessária já está instalada.

Programação

Lembrando que nosso projeto funcionará semelhante aos sensores de estacionamento utilizados nos carros. A programação que utilizaremos está logo abaixo.

Se você quer baixar a programação, **Clique Aqui.**

```
#include "Ultrasonic.h"  // Esse comando é para incluir a biblioteca no programa


Ultrasonic ultrasonic (8,9); // são as portas do sensor, Trigger na porta 5 e Echo
na porta 6

// declarando os LEDs e o Buzzer e suas portas onde estão conectados

int ledVerde = 2;
int ledAmarelo = 3;
int ledVermelho =4;
int buzzer = 5;

// Variáveis de controle

long microsec = 0;
float distanciaCM = 0;

void setup() {

Serial.begin(9600); // iniciando o monitor serial na velocidade 9600

// Declarando os LEDs e o Buzzer com saídas

  pinMode(ledVerde,OUTPUT);
  pinMode(ledAmarelo,OUTPUT);
  pinMode(ledVermelho,OUTPUT);
  pinMode(buzzer,OUTPUT);
}

void loop() {

// lendo o sensor
microsec = ultrasonic.timing();
```

```
// convertendo a distância para centímetros

  distanciaCM = ultrasonic.convert(microsec, Ultrasonic::CM);

  ledDistancia();

// mostrar a distância no monitor serial
Serial.print(distanciaCM);

// unidade de medida

Serial.println(" cm");
delay(500);
}

void ledDistancia() {

// se inicia com todos os LEDs apagados
digitalWrite(ledVerde,LOW);
digitalWrite(ledAmarelo,LOW);
digitalWrite(ledVermelho,LOW);

// Se a distância for menor ou igual a 30cm e maior ou igual 20

  if (distanciaCM <=30 and distanciaCM >= 20) {
 digitalWrite(ledVerde,HIGH); // LED Verde liga
 }

// Se a distância for menor ou igual a 20 cm e maior ou igual 10

 if (distanciaCM <=20 and distanciaCM >= 10) {

 digitalWrite(ledAmarelo,HIGH); // LED amarelo acende
 tone (buzzer, 2500, 100); // buzzer começa a emitir som
 }
```

```
// se a distância for menor que 10cm

  if (distanciaCM < 10) {
    digitalWrite(ledVermelho,HIGH); // LED Vermelho acende
    tone (buzzer, 2500, 1000); // Buzzer emite som mais intenso
  }
}
```

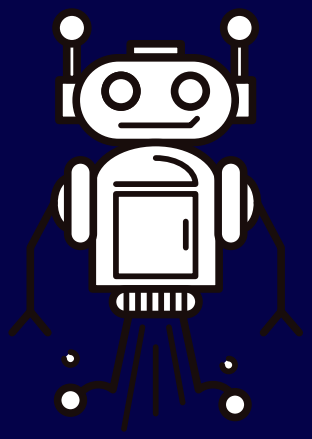
Baixe o código do projeto **Clicando Aqui!**

Vendo o Funcionamento

Quer ver o funcionamento na prática ?? **Clique aqui!!**

5 Robô Autônomo

Agora que conhecemos os componentes necessários para se construir um robô, vamos iniciar nossos **projetos principais**.

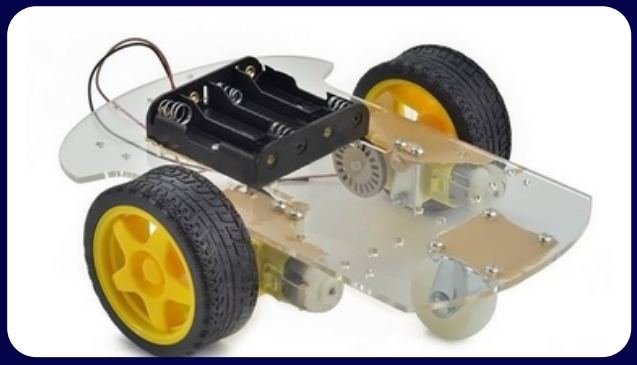


Neste primeiro projeto vamos desenvolver um **Robô Autônomo** que desvia de objetos, será utilizado todos os componentes que já conhecemos, como o **Arduino Uno**, a **Ponte H**, os **Motores DC**, o **Sensor Ultrassônico** e a **Protoboard**. Você pode comprar o **Kit Arduino Robôs** com o qual é possível montar todos os kits dessa apostila!

Montagem do Chassi

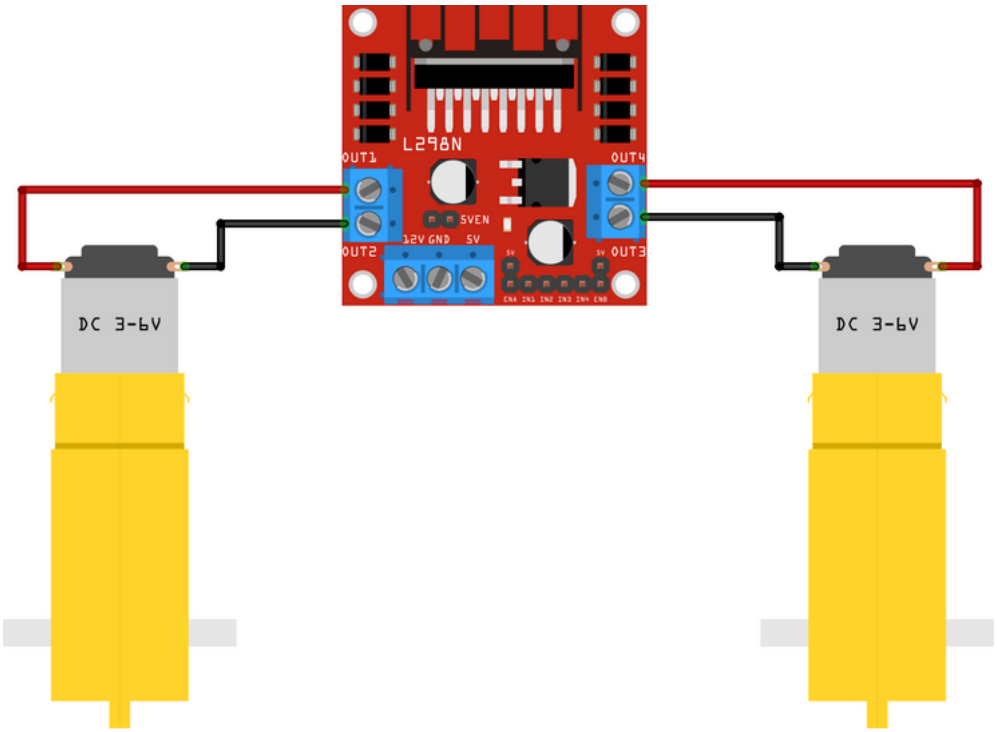
O primeiro passo é montar o **Kit Chassi 2WD**, você pode seguir o **manual** de montagem que vem no **Kit de Robótica!!**

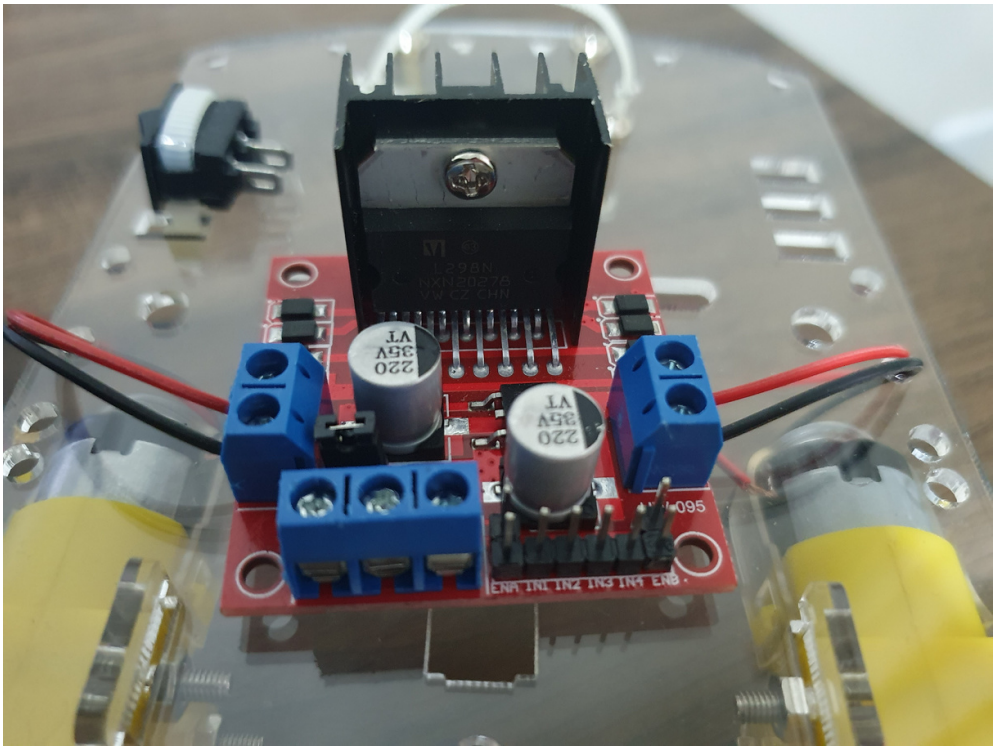
OBS: O Kit acompanha um suporte para pilhas, como nosso robô consome bastante energia, não é recomendado utilizar pilhas, e o **Arduino Uno**, irá ocupar certo espaço, assim sendo, é sugerido que não coloque o suporte.



Ligação dos Motores

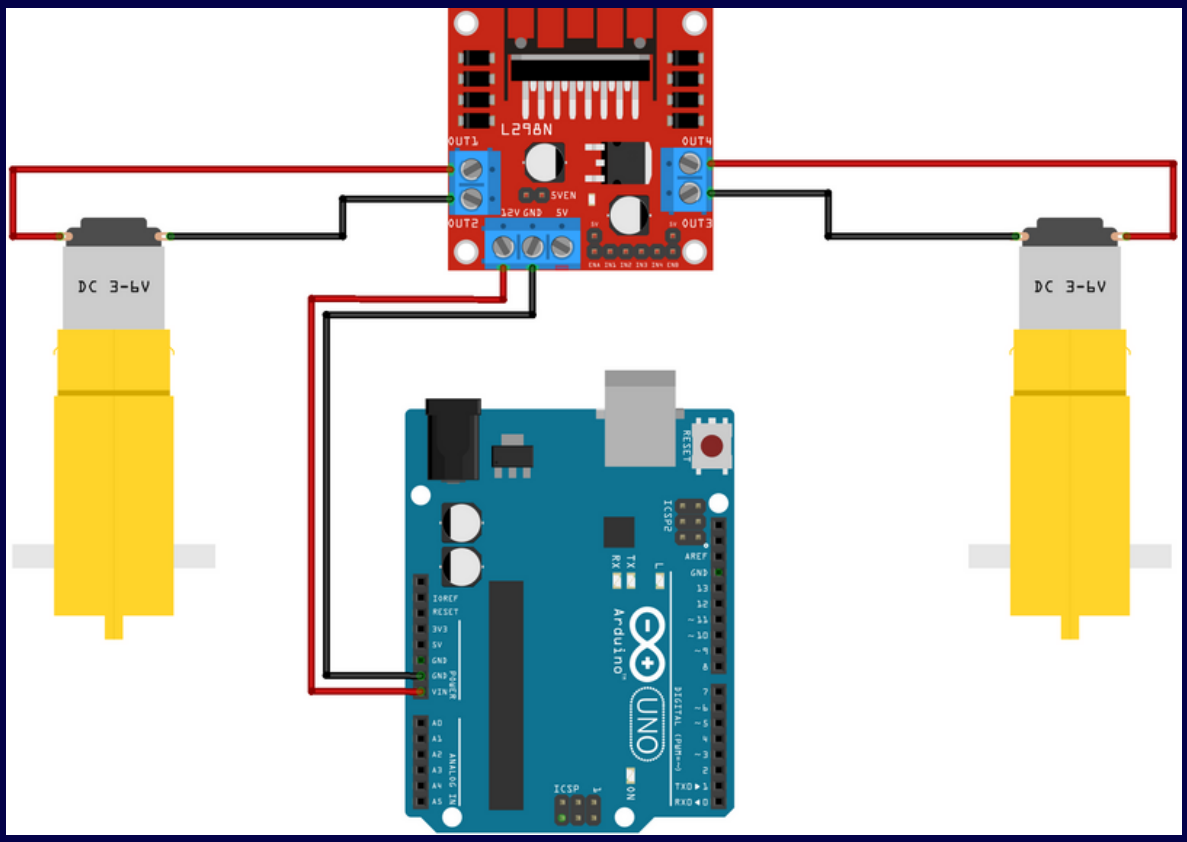
Lembre-se que os fios dos motores devem estar ligados **iguais**, na mesma ordem, para que robô ande no **sentido correto**.





OBS: Remova os jumpers de **ENA** e **ENB** e **Mantenha** o **jumper ATIVA 5V** conectado !!

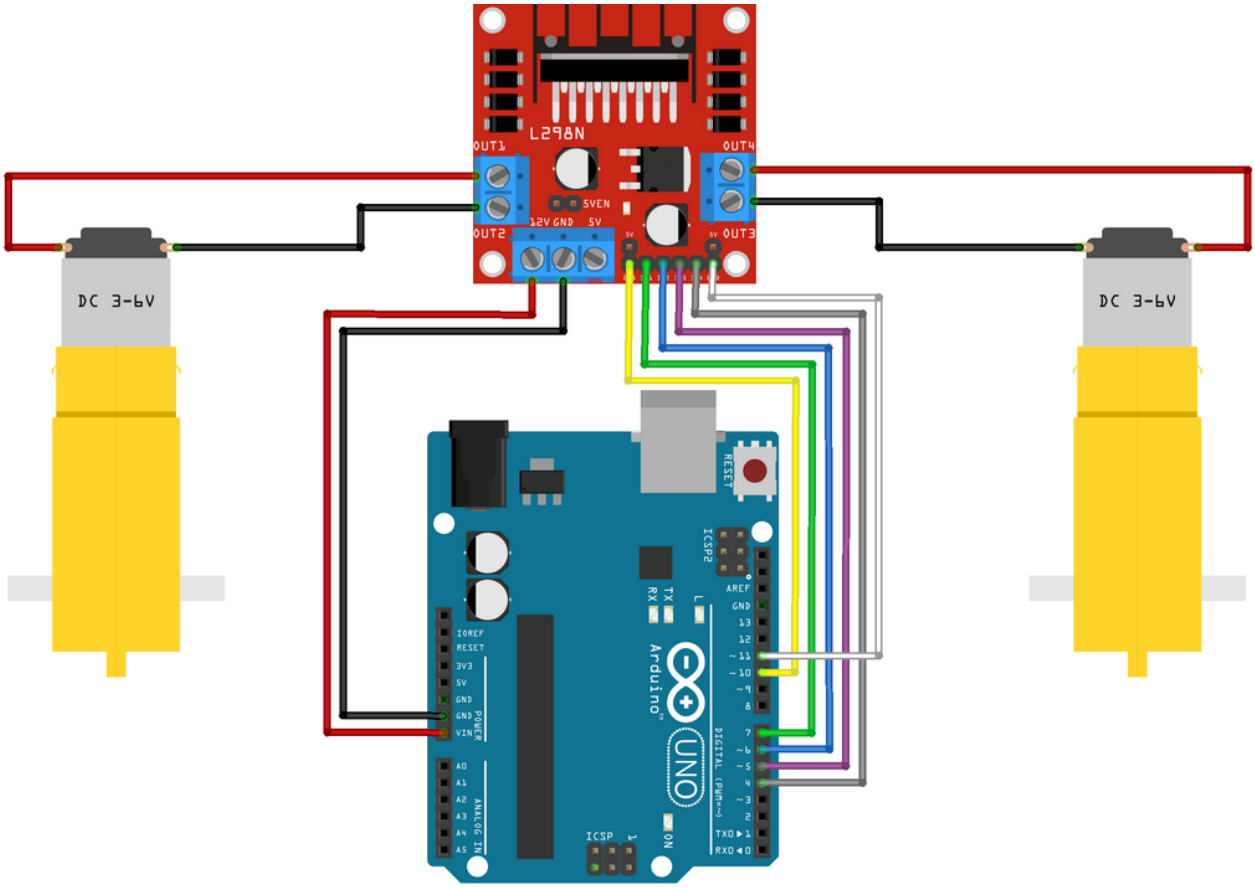
Alimentação da Ponte H



Conecte o o **VCC** da **Ponte H** no pino **Vin** do Arduino e o **GND** no **GND** do Arduino.

Ligações no Arduino

O próximo passo é realizar as conexões dos pinos da **Ponte H** no **Arduino**.



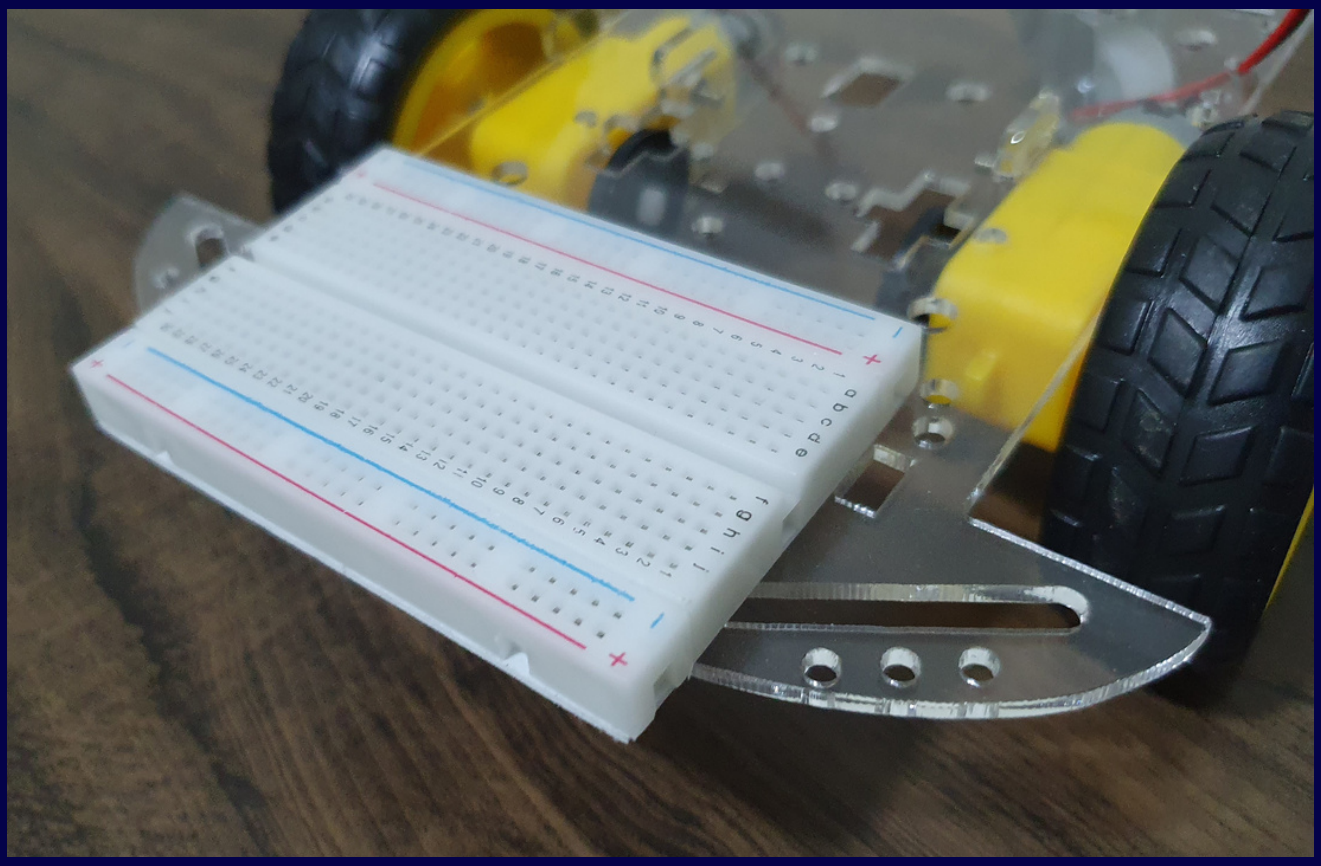
Desse modo temos:

ENA: Porta **10**
ENB: Porta **11**

IN1: Porta **7**
IN2: Porta **6**
IN3: Porta **5**
IN4: Porta **4**

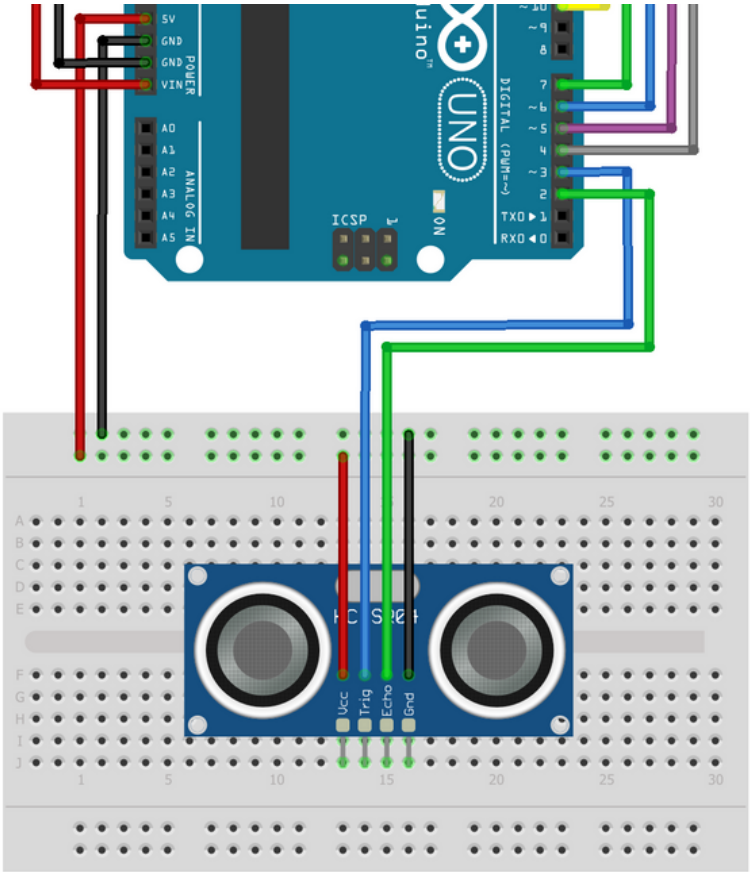
Inserindo a Protoboard

Retire a proteção do adesivo e cole a Protoboard na parte da frente do chassi. Vamos utiliza-la para conectar nosso Sensor Ultrassônico.



Conectando o Sensor Ultrassônico

Vamos conectar o Sensor Ultrassônico na Protoboard e realizar as ligações no Arduino.



1- Primeiro, vamos alimentar a **Protoboard** com o **GND** e os **5V** do Arduino.

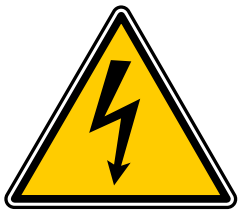
2- Conecte o pino **VCC** do Sensor no **5V** da **Protoboard** e o pino **GND** no **GND** da **Protoboard**.

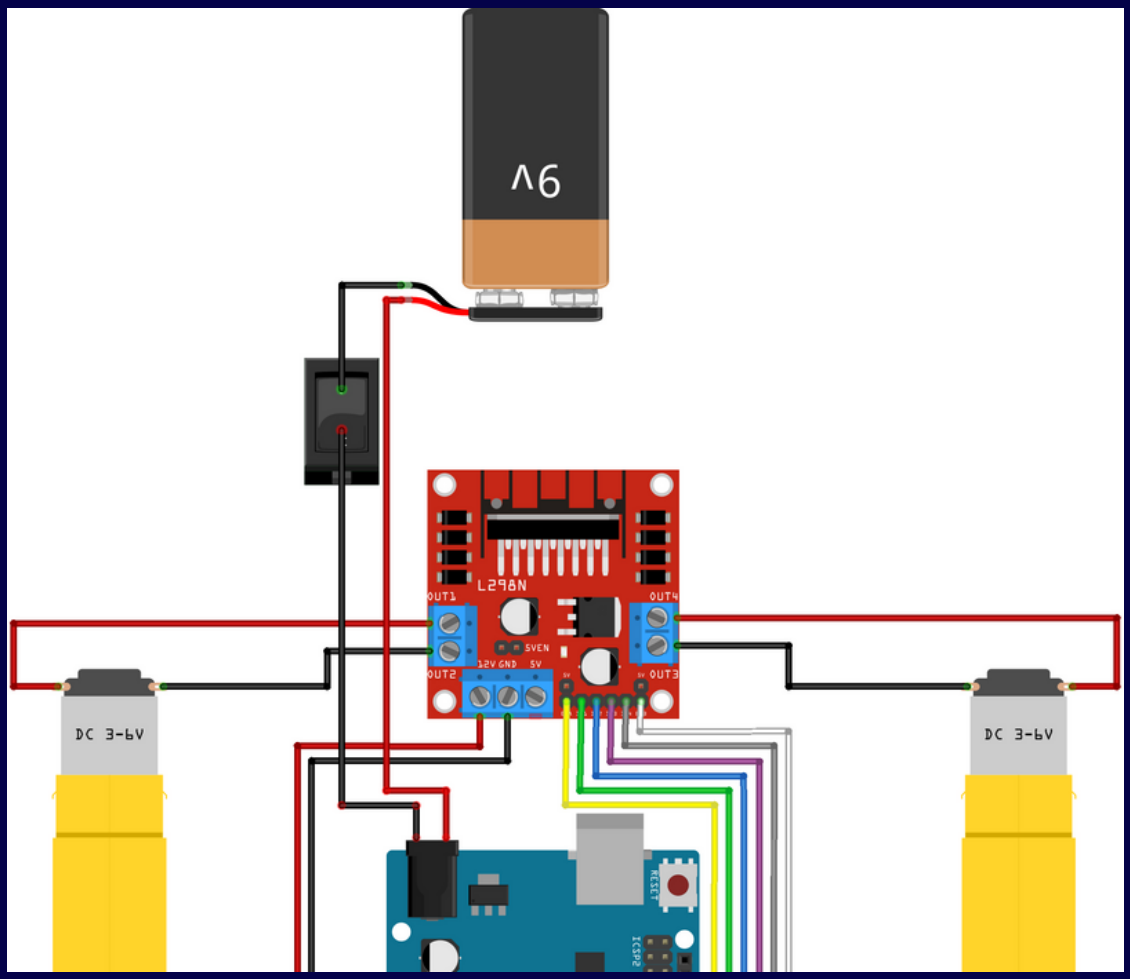
3- Conecte o Pino **Trig** na Porta **3** do **Arduino**.

4- Conecte o Pino **Echo** na Porta **2** do **Arduino**.

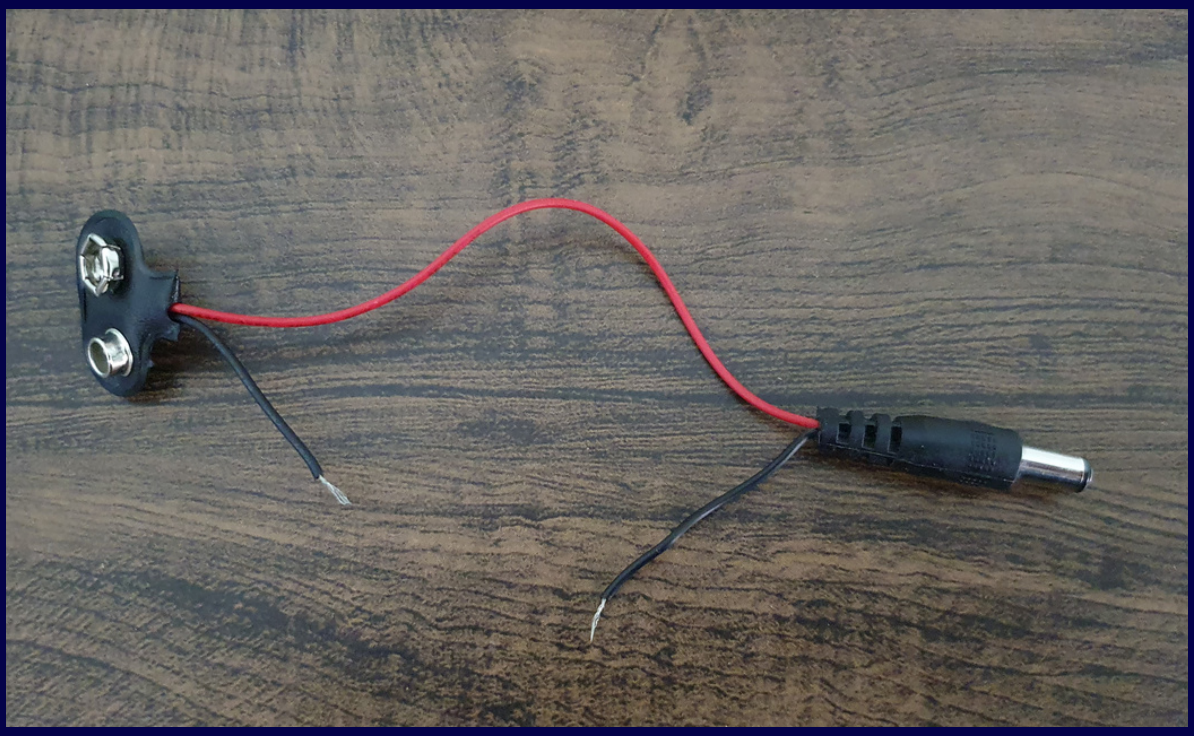
Alimentação do Circuito

Vamos utilizar uma **bateria de 9V**, o **adaptador** e a **chave on/off**.

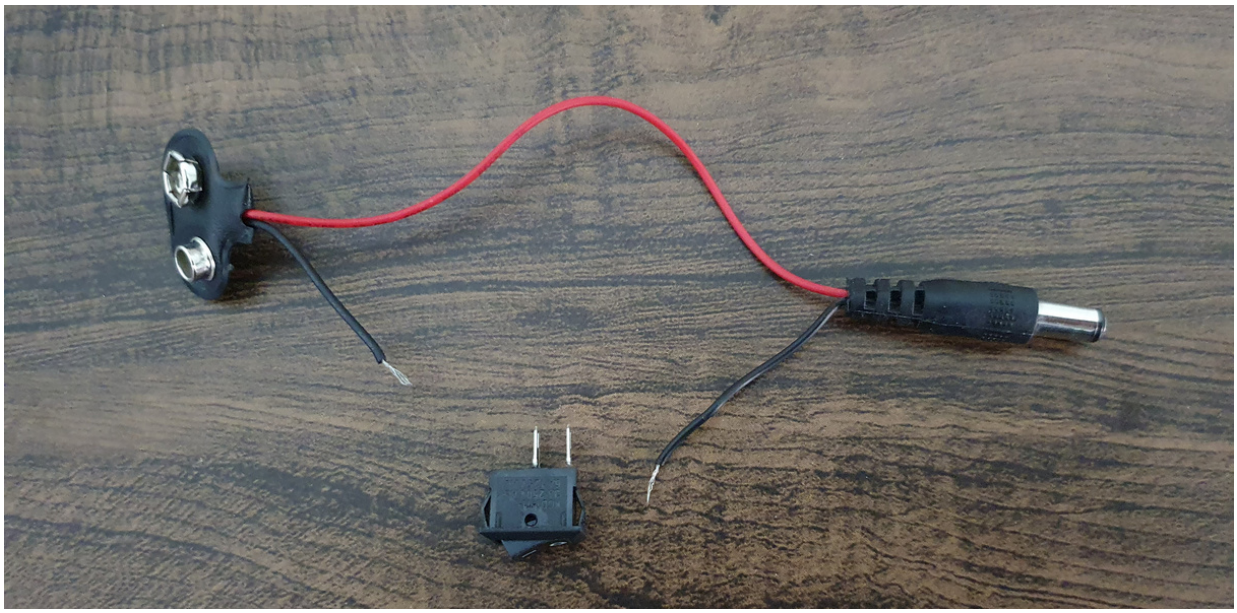


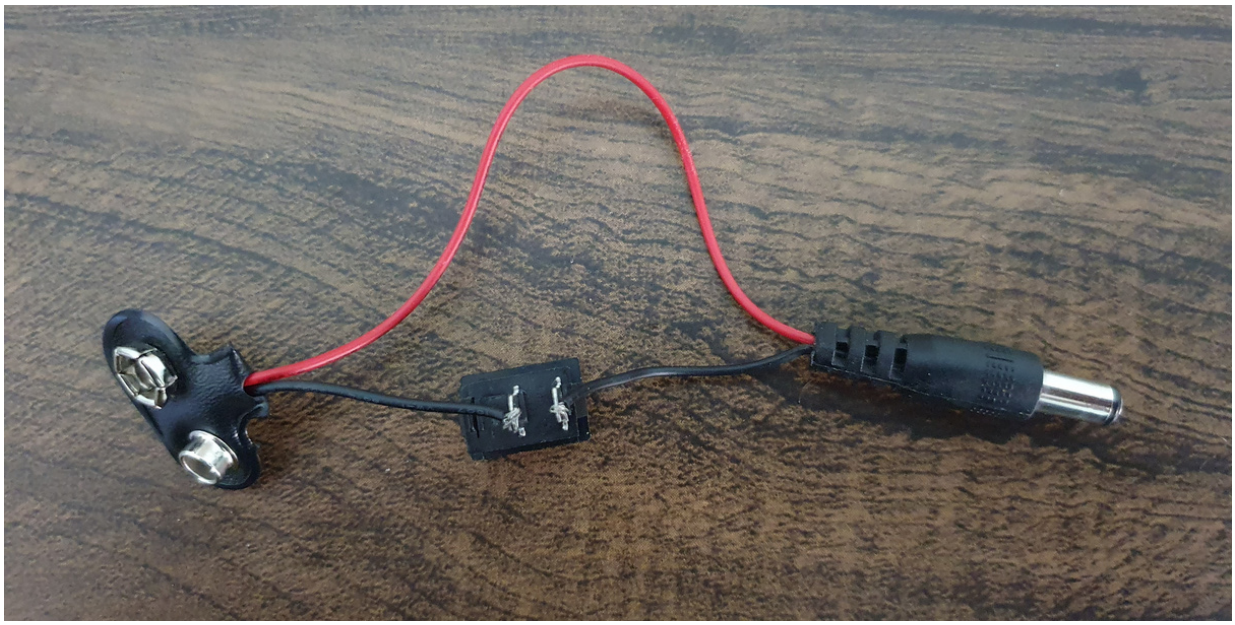


Será necessário realizar um **corte no fio negativo** do **adaptador da bateria** e decapar as duas pontas.



Agora, é necessário encaixar os fios decapados na **chave**.

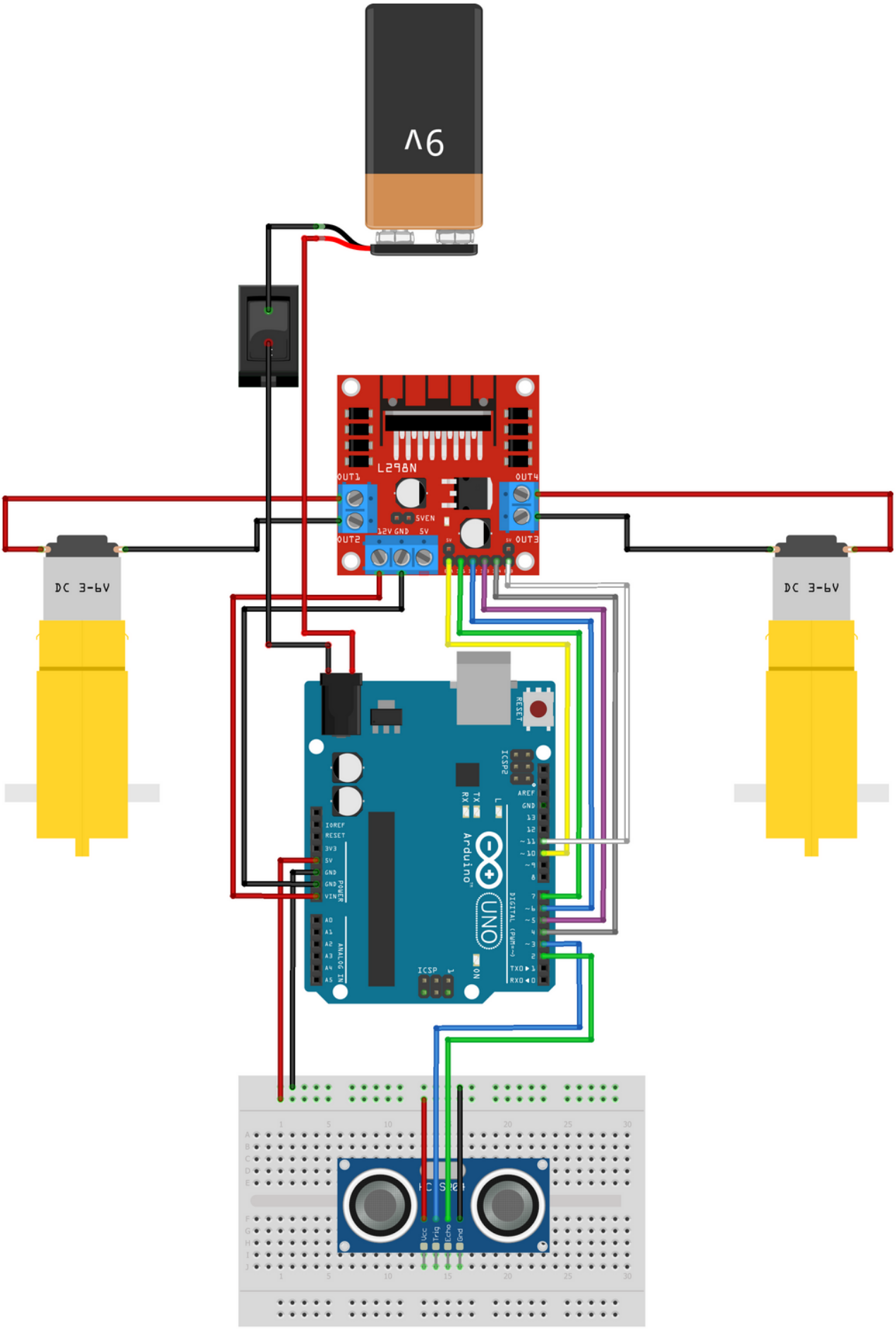


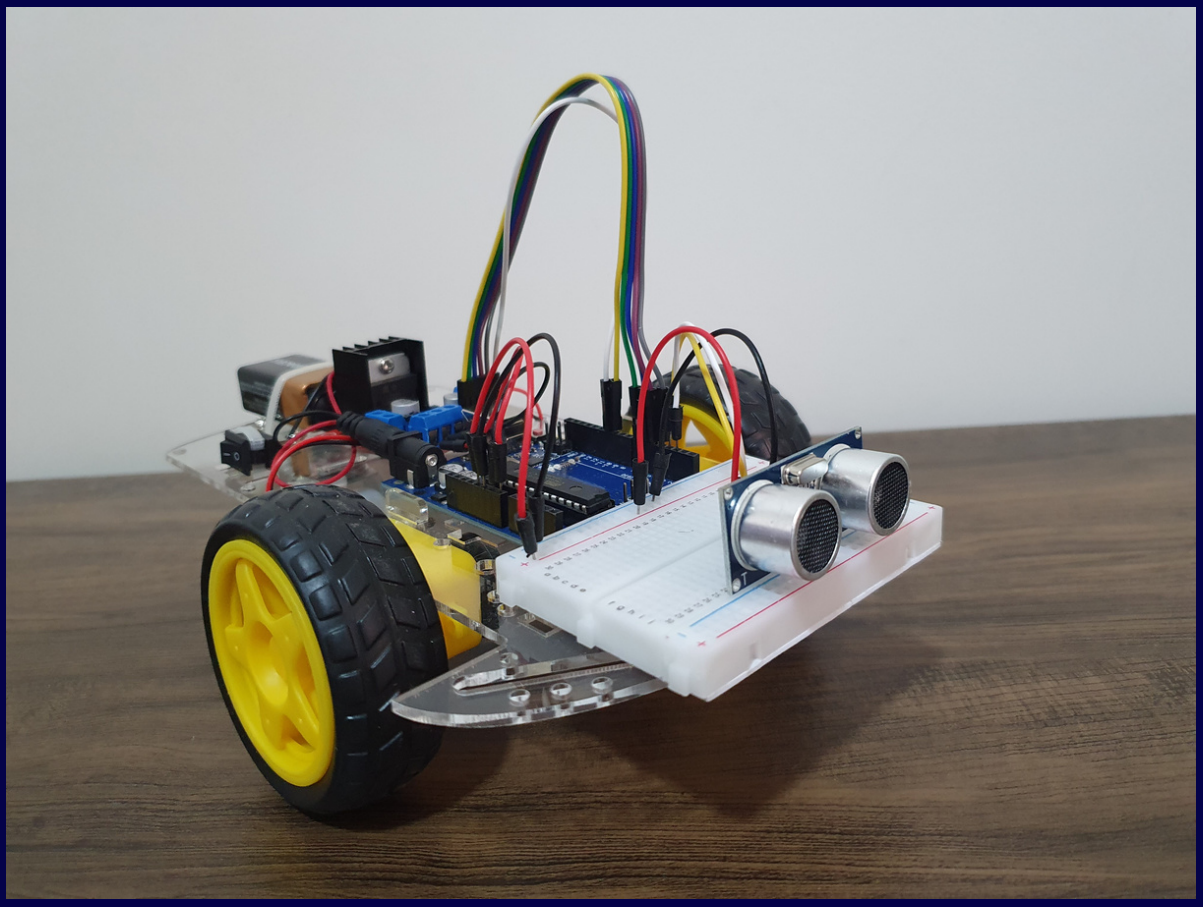


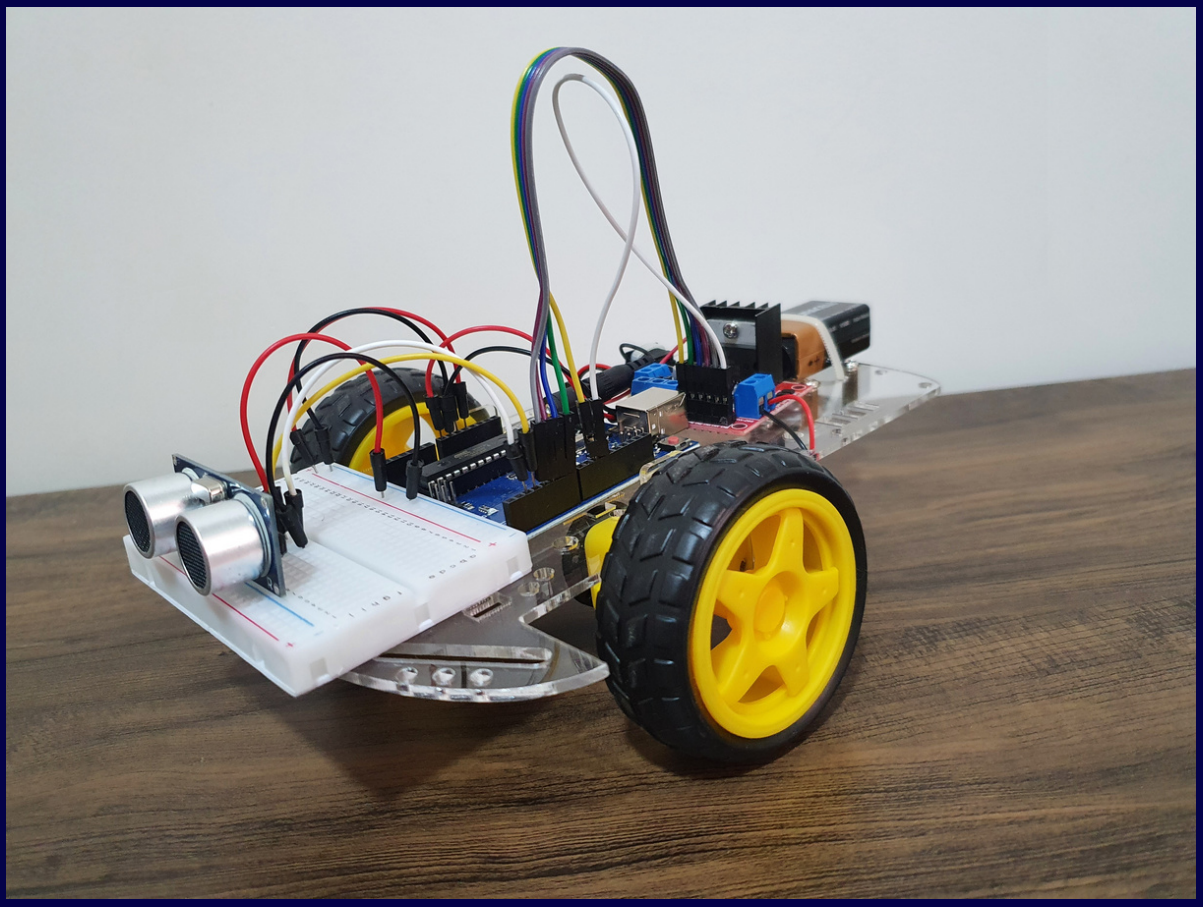
Para que não ocorra dos fios escaparem, é recomendando **soldar** os fios na chave !!

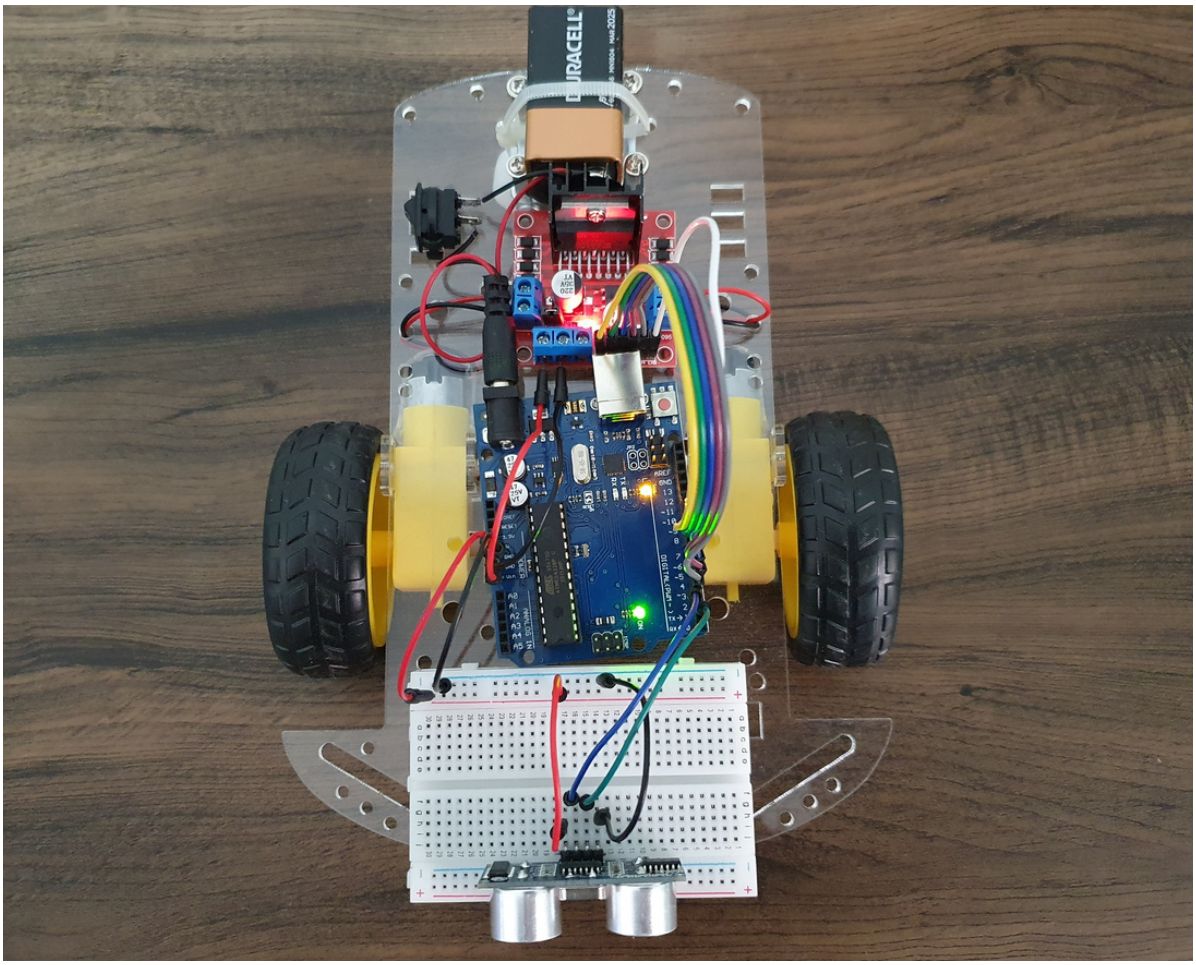
Lembre-se, se você for **menor de idade**, peça ajuda a um **adulto** para a realização da solda !!

Circuito Finalizado









Programação

Por fim, vamos iniciar a programação do nosso robô !!!

```
#include <Ultrasonic.h>; // incluindo a biblioteca para o Sensor Ultrassônico

#define TRIGGER_PIN 3 // Pino Trigger do sensor na porta 3
#define ECHO_PIN  2 // Pino Trigger do sensor na porta 2

Ultrasonic ultrasonic(TRIGGER_PIN, ECHO_PIN);

// definindo as portas onde estão conectados os motores

#define IN1 7  //Pinos motor A (Direita)
#define IN2 6  //Pinos motor A (Direita)
```

```
#define IN4 4  //Pinos motor B (Esquerda)

#define ENA 10  //Pino velocidade motor A (Enable A)
#define ENB 11  //Pino velocidade motor B (Enable B)

void setup ()
{

 //Definindo os motores como saídas

  pinMode(IN1,OUTPUT); //Saída para motor A
  pinMode(IN2,OUTPUT); //Saída para motor A
  pinMode(IN3,OUTPUT); //Saída para motor B
  pinMode(IN4,OUTPUT); //Saída para motor B
  pinMode(ENA,OUTPUT); //Controle velocidade motor A
  pinMode(ENB,OUTPUT); //Controle velocidade motor B

// Velocidade dos motores, caso note que seu robô está muito lento, altere os valores.

  analogWrite(ENA,120); //Controle PWM do motor A (0 a 255)
  analogWrite(ENB,120); //Controle PWM do motor B (0 a 255)
  delay(1000); // Aguarda 1 segundo antes de iniciar
}

void loop() // loop principal
{

//Robô inicia andando para frente

  robo_frente();

   float dist_cm = distancia(); // Declara variável que armazena a distância do obstáculo
```

```
if (dist_cm < 20) // distância menor que 20cm ?
   {
     decisao();
   }

delay(100);
}

float distancia() //função que mede a distância em cm
{

  float cmMsec; //declara a variável tipo float cmMsec

  //Associa à variável tipo long microsec o tempo em microsegundos

long microsec = ultrasonic.timing();

 // função da biblioteca para converter a distância em cm e associá-la a cmMsec

  cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);
  return(cmMsec); // retorna o valor em cm para a função delay(10);
delay(10);
}

void robo_frente()  // função para mover o robô para frente lembre-se da tabela
da Ponte H

{
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
}
```

```
void robo_esquerda() //função para mover o robô para esquerda
{
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
}

void robo_parado() //função para parar o robô

{

  digitalWrite(IN1,LOW);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
}

void decisao() //função para decidir a ação do robô
{
  robo_parado();
  delay(500);
  robo_esquerda();
  //esse tempo precisa ser avaliado para que o robô vire por volta de 90 graus
  delay(400);
  robo_parado();
  delay(500);
}
```

Baixe o código do projeto **Clicando Aqui**

Possíveis Problemas

1- O robô não anda ou está muito lento

Esse problema acontece por conta da **bateria.** Nosso robô consome **muita energia** e as baterias comuns **não suportam** muito tempo, é aconselhável utilizar baterias recarregáveis e fontes de 9V para realização dos testes.

Caso a bateria seja nova e o robô continuar lento, modifique o **código** na seguinte parte:

```
analogWrite(ENA,120);
analogWrite(ENB,120);
```

Escolha uma valor entre **0** a **255** e teste.

2- O robô anda, mas começa a rodar e não para

Esse problema também é causado pela **bateria**.

OBS: Para evitar outros tipos de problemas, siga o passo a passo para montar o circuito !!

6 Robô com Bluetooth

Vamos realizar uma pequena alteração no nosso projeto anterior e transformar nosso **Robô Autônomo** em um **Robô Controlado Por App via Bluetooth** utilizando nosso Smartphone.
Mas antes, é importante conhecer o **Módulo Bluetooth** que vamos utilizar !!!

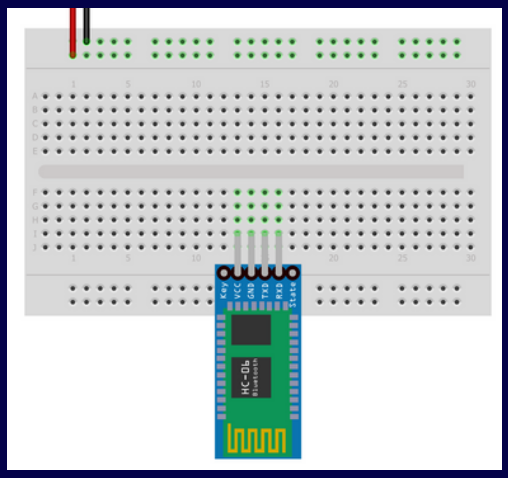
Módulo Bluetooth HC-06

O **Módulo Bluetooth HC-06** é feito para comunicação sem fio em pequeno alcance, capaz de enviar dados para outro dispositivo via Bluetooth com segurança e agilidade. Ele trabalha apenas como **Slave**, ou seja, apenas recebe informações.

Com esse módulo é possível controlar servos, motores, LEDs e outras diversas aplicações !!

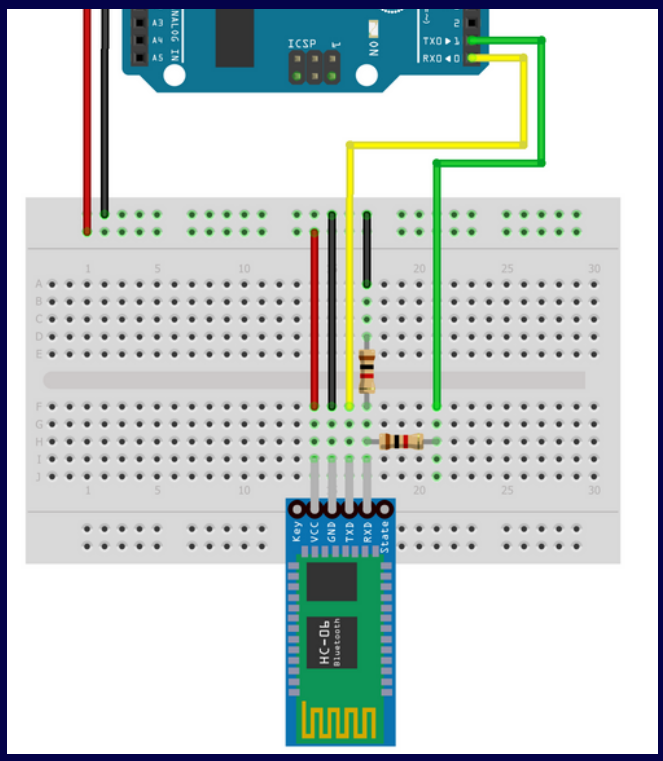
Circuito

Vamos realizar pequenas alterações no nosso circuito anterior, primeiro vamos remover o sensor ultrassônico e adicionar nosso Módulo Bluetooth.



Antes de conectarmos nosso módulo no Arduino, é importante saber que apesar dele aceitar 5V para sua alimentação, o módulo opera com a **tensão máxima de 3.3V**, e o Arduino fornece em suas portas 5V.

Assim, é necessário fazermos um **divisor de tensão**. Para isso vamos utilizar dois **resistores** de **1KΩ**, dessa forma a tensão ficará em 2,5V.



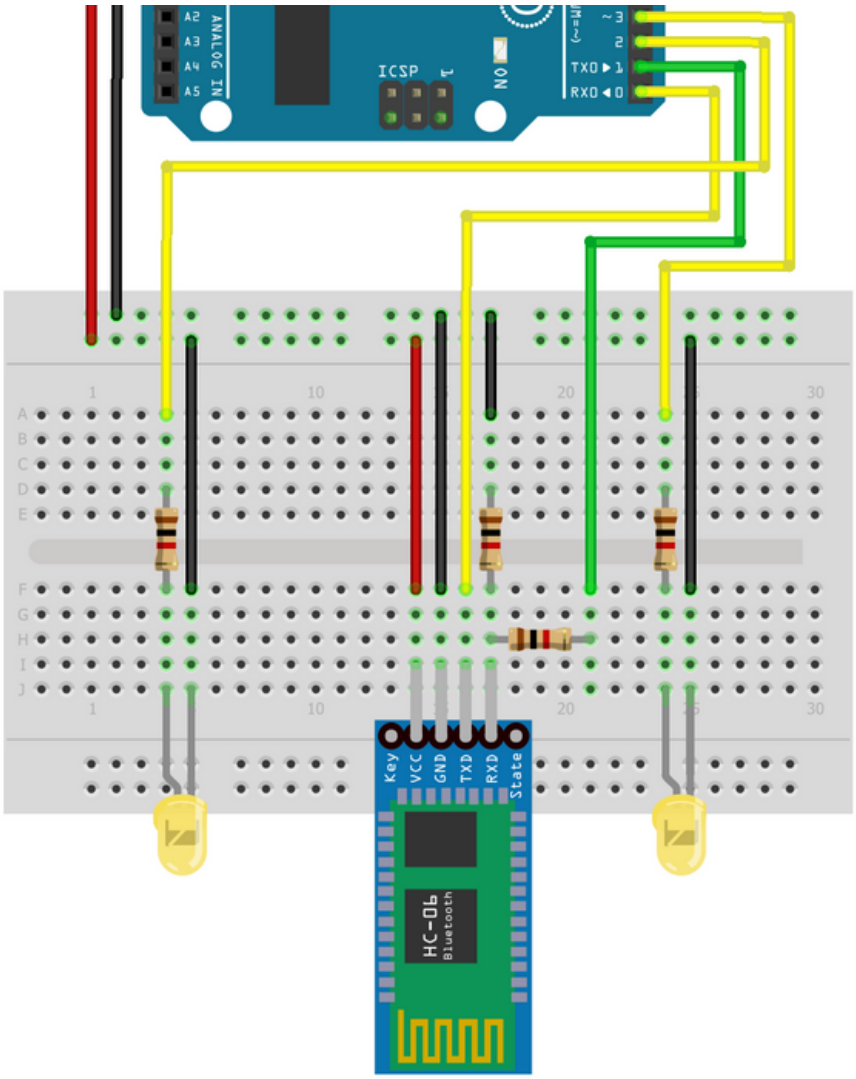
O pino **VCC** do Módulo deve estar conectado nos **5V da protoboard**, o pino **GND** no **GND**, o pino **TXD** na porta **RXD** e o pino **RXD** deve estar conectado a dois resistores de **1KΩ** e na porta **TXD** do Arduino.

OBS: um dos resistores deve estar no **GND** da **Protoboard**.

VCC ---- 5V
GND ---- GND
TXD ---- RXD
RXD ---- TXD

Extra...

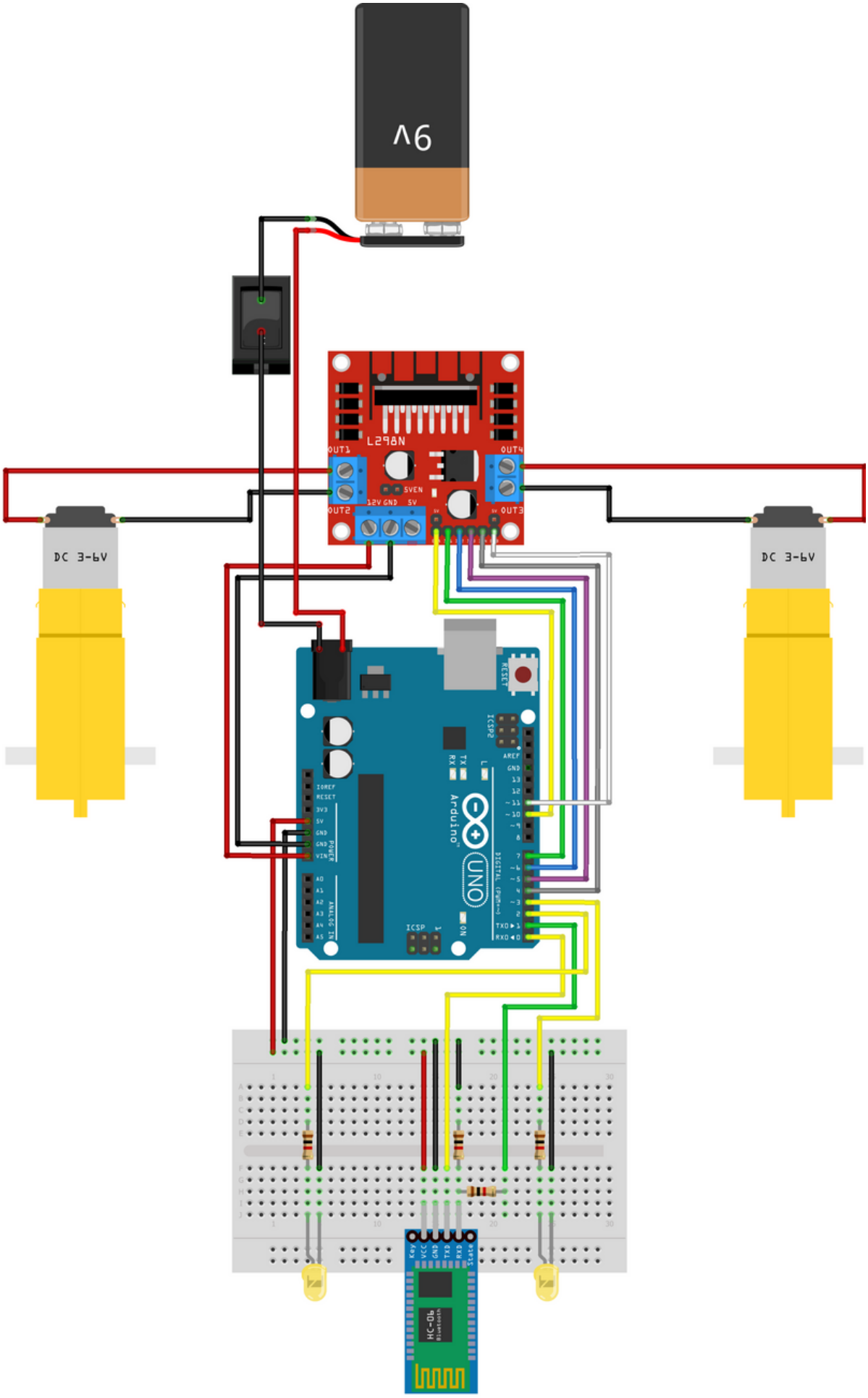
Vamos conectar dois **LEDs** para simularmos um farol no nosso robô.

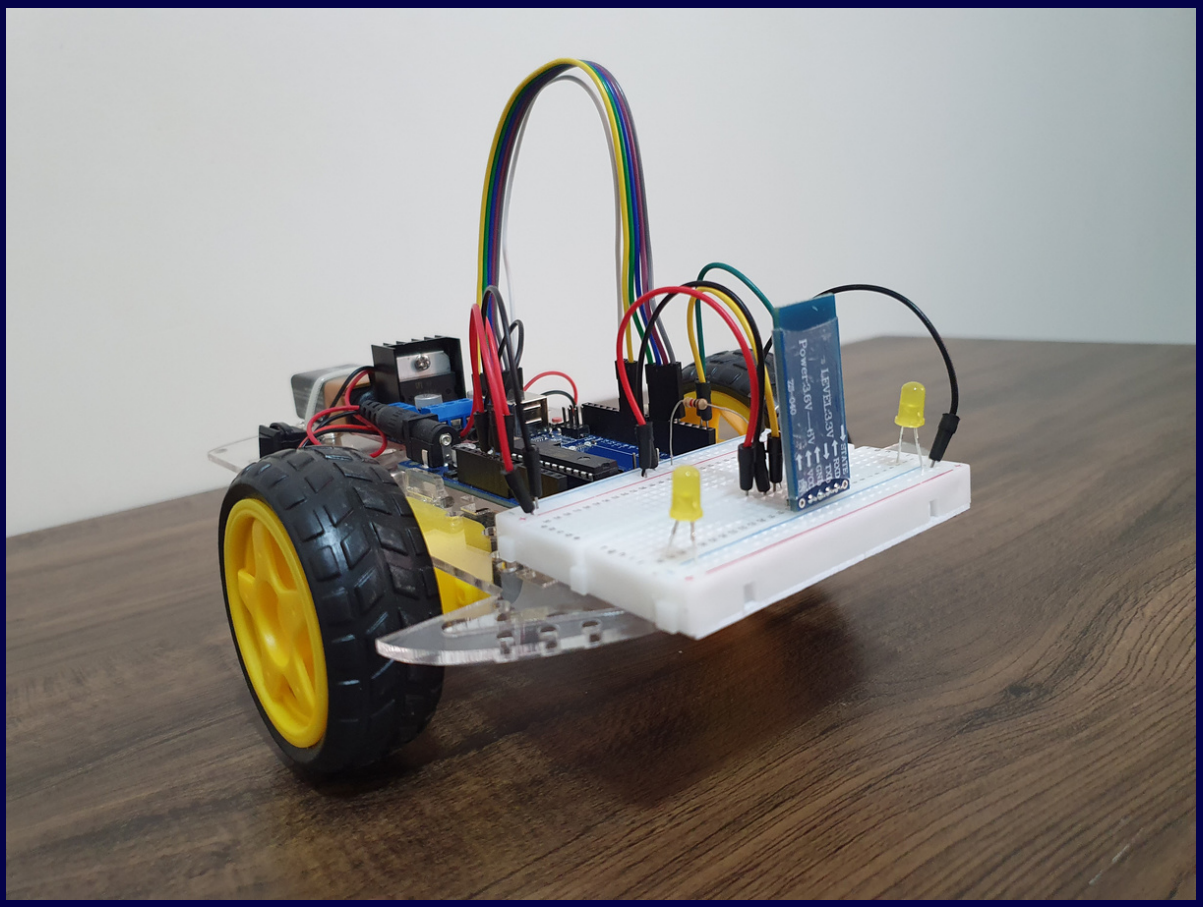


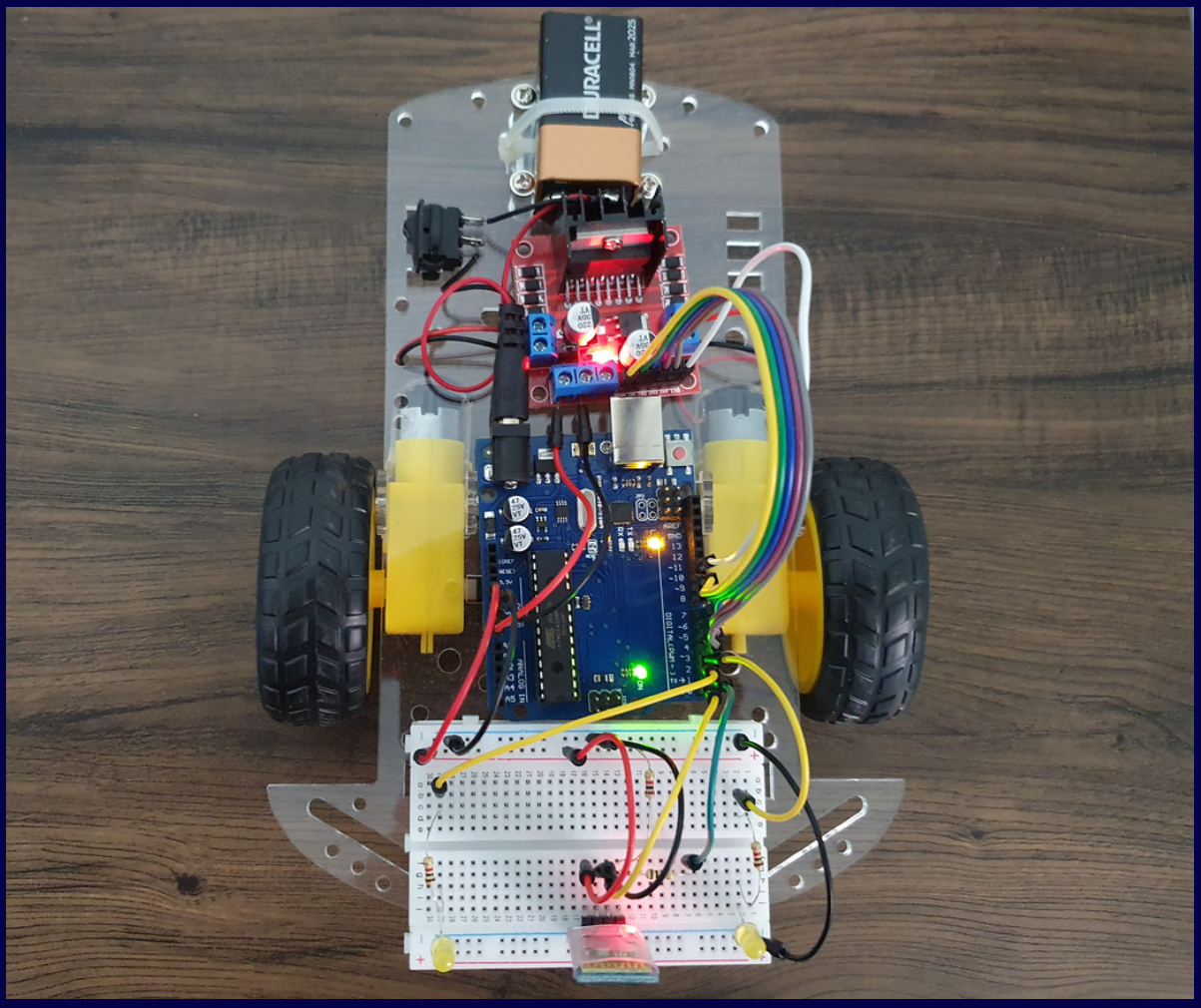
- As **menores pernas** dos **LEDs** são **negativas**, devem ser conectadas no **GND** da **Protoboard**.

- As pernas positivas devem ser conectadas aos **resistores** de **1KΩ** e nas portas **3** e **4** do Arduino.

Circuito Completo




LEVEL 4 Shift
Power 4V—7V


DURACELL

Programação

Antes de enviar a programação para o Arduino, desconecte os Pinos **RXD** e **TXD** do Arduino !!!

O cabo USB utiliza as mesmas portas para enviar a programação, caso elas estejam conectadas ao sensor, **não será possível enviar o código.**

```
// Definição dos pinos de saída para os dois motores e os LEDs

#define IN1 7
#define IN2 6
#define IN3 5
#define IN4 4
int led1 = 2;
int led2 = 3;

//Definindo pinos para o controle de velocidade dos motores

#define ENA 10
#define ENB 11

void setup() {

  Serial.begin(9600); // inicia a porta serial, configura a taxa de dados para 9600
bps

// Declarando os motores e os LEDs como Saídas
  pinMode(IN1,OUTPUT);
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(IN3,OUTPUT);
```

```
pinMode(IN4,OUTPUT);
pinMode(ENA,OUTPUT);
pinMode(ENB,OUTPUT);

// Velocidade dos Motores, Caso esteja muito lento ou muito rápido altere os números entre 0 a 255

analogWrite(ENA,110);
analogWrite(ENB,110);
}

char cha;

void loop() {

if (Serial.available() > 0) {
   cha = Serial.read();
  delay(2);

// Movimenta para frente

if(cha == 'F'){

  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
}

//Movimenta para trás

if(cha == 'B'){
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
```

```
digitalWrite(IN3,LOW);
digitalWrite(IN4,HIGH);
}

//movimenta para direita

if(cha == 'R'){

  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
}

//movimenta para esquerda

if(cha == 'L'){

  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
}

//Sem movimento

if(cha == '0'){

  digitalWrite(IN1,LOW);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);

}
```

```
// Liga os LEDs

if(cha == '3'){

  digitalWrite(led1,HIGH);
  digitalWrite(led2,HIGH);
}

//Desliga LEDs

if(cha == '4'){

  digitalWrite(led1,LOW);
  digitalWrite(led2,LOW);

}
}
}
```

Baixe o código do projeto **Clicando Aqui**

Se conectando...

Para conseguirmos controlar nosso robô, vamos precisar de um smartphone **Android**.

Se conectando...

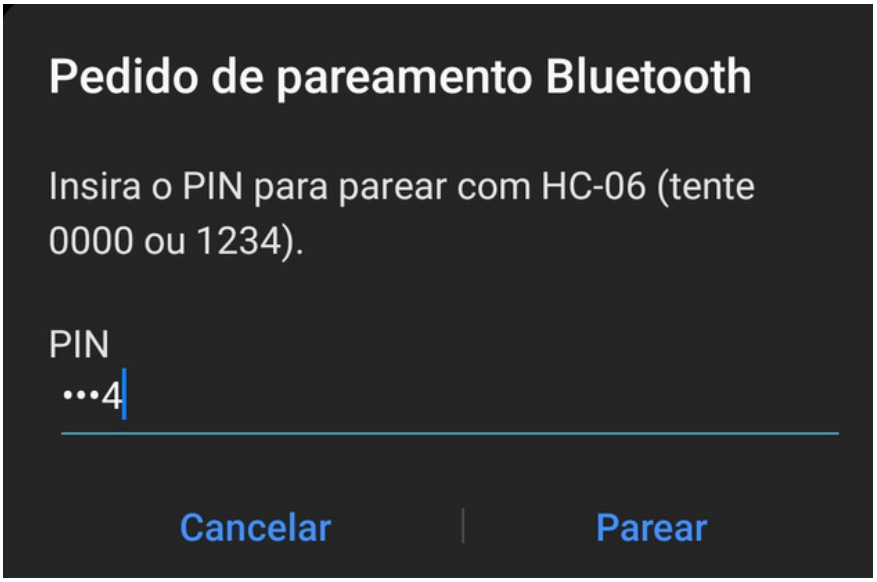
Ative o Bluetooth do celular e procure na lista o nome **HC-06**.

Clique sobre **HC-06**

Ao clicar sobre o nome **HC-06** é necessário inserir uma senha para parear o celular com o módulo.

A senha é 1234

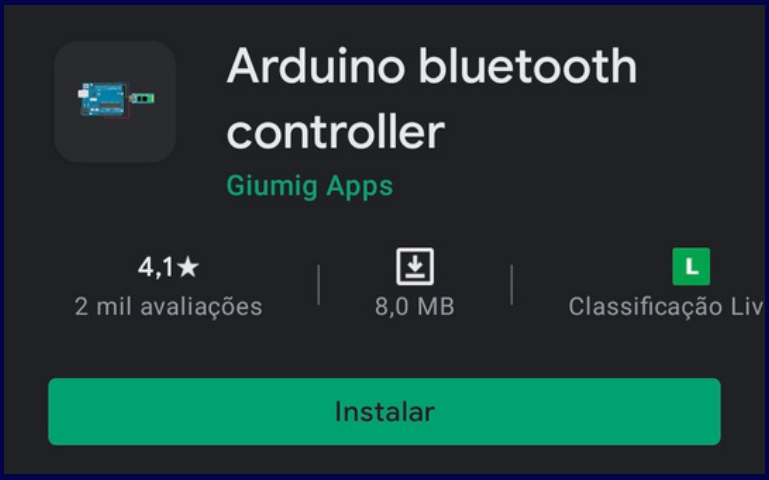
OBS: Caso essa senha não funcione tente **0000**.



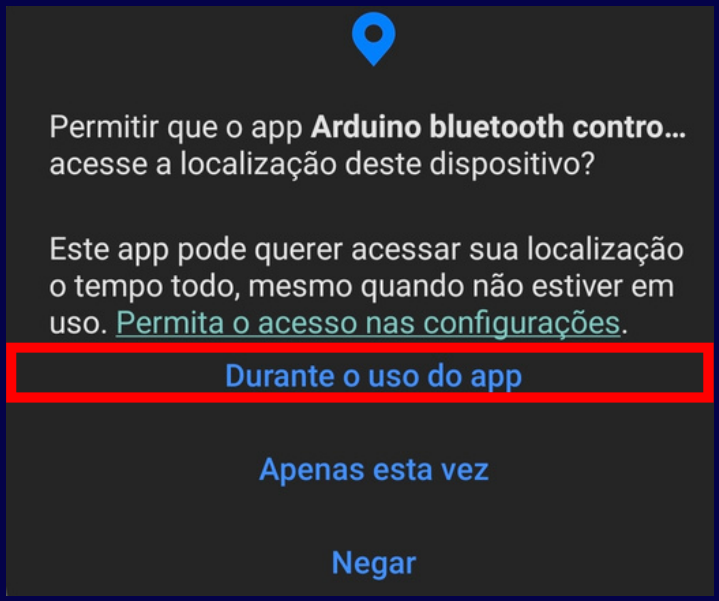
Clique em **Parear** e pronto !! Agora nosso módulo já está pareado com nosso celular.

2 Baixando o Aplicativo

Para controlar nosso robô é necessário baixar o aplicativo **Arduino Bluetooth Controller** na Play Store.

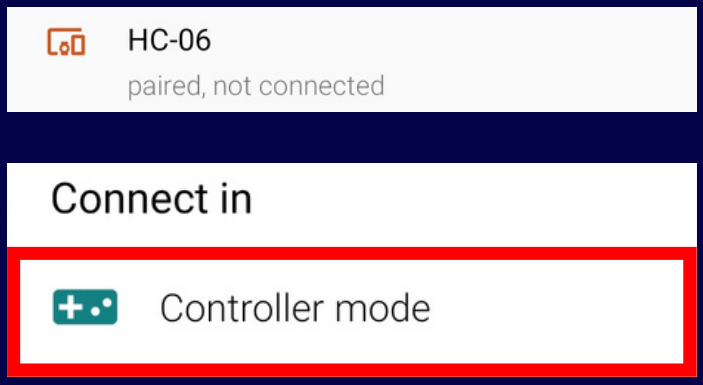


Abra o aplicativo e permita **ele utilizar a localização durante o uso**.



Clique sobre o nome **HC-06** na lista, escolha a opção **Controller Mode**.

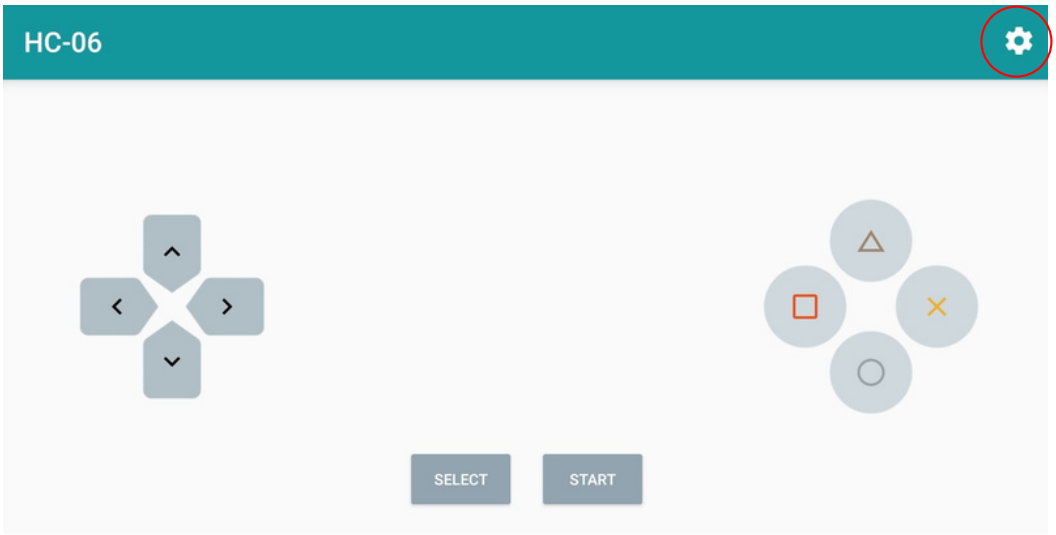
- Note que há uma luz piscando no módulo

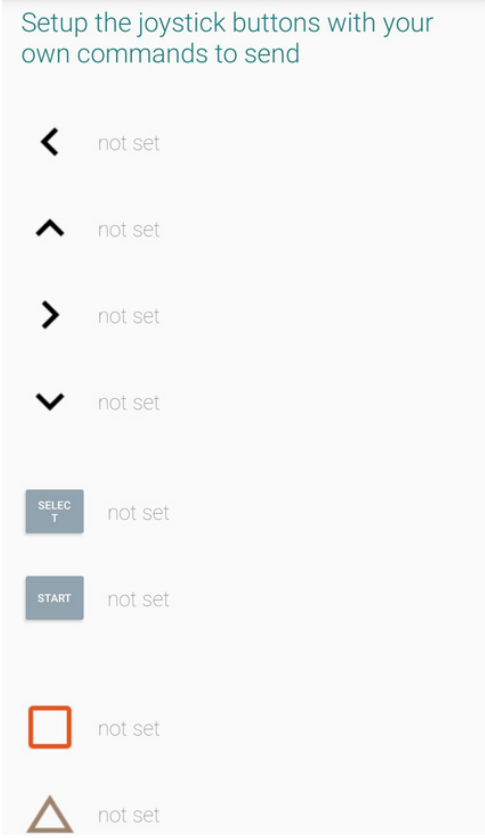


Se a **luz** do módulo parou de piscar significa que a conexão foi realizada com **sucesso**. **Agora só falta configurar os botões** !!

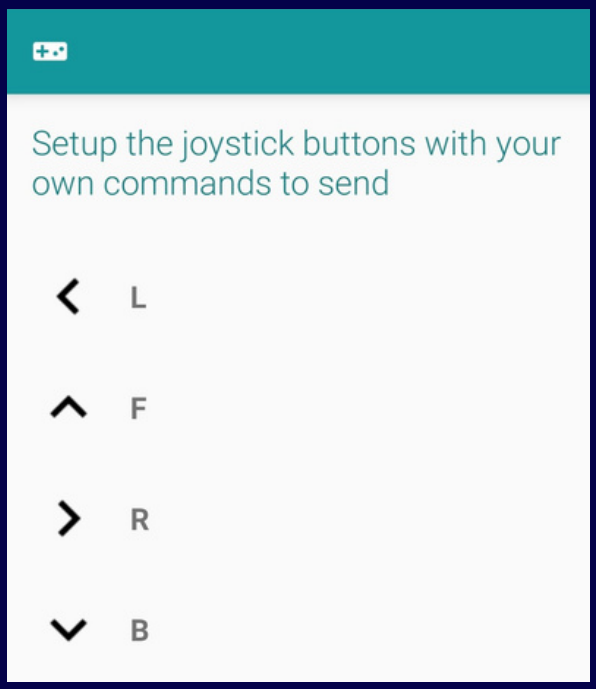
Configurando os Botões

Clique sobre a **engrenagem**.





note que não a nada registrado nos botões, vamos realizar a **configuração** de acordo com nossa **programação**.

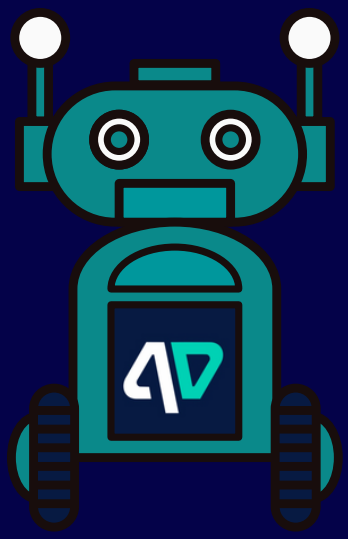


- Esquerda: **L**
- Frente: **F**
- Direita: **R**
- Trás: **B**

- **Quadrado -** Liga LED = **3**
- **Triangulo -** Desliga LED = **4**
- **X -** Para Robô = **0**

Não se esqueça de **salvar a configuração** !!

Agora nosso robô está pronto, é só clicar nos botões desejados !!!!



Possíveis Problemas

1- O robô não anda ou está muito lento

Esse problema acontece por conta da **bateria.** Nosso robô consome **muita energia** e as baterias comuns **não suportam** muito tempo, é aconselhável utilizar baterias recarregáveis e fontes de 9V para realização dos testes.

Caso a bateria seja nova e o robô continuar lento, modifique o **código** na seguinte parte:

```
analogWrite(ENA,110);
analogWrite(ENB,110);
```

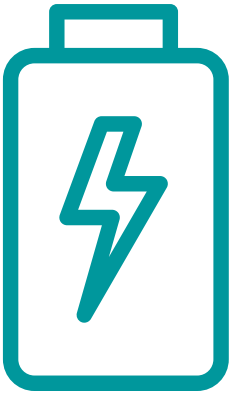
Escolha uma valor entre **0** a **255** e teste.

2- O robô só obedece um comando

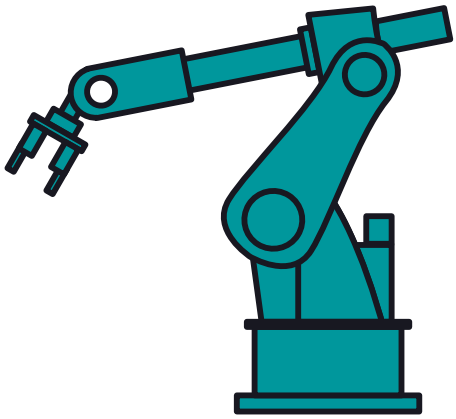
Esse problema também é causado pela **bateria**, para resolve-lo troque a bateria ou altere a programação como foi mostrado acima.

OBS: Para evitar outros tipos de problemas, siga o passo a passo para montar o circuito !!

OBS: É recomendado utilizar **bateria recarregáveis**, **fontes** para realização dos testes e até mesmo **carregadores portáteis**. Nosso robô consome muita energia e as baterias convencionais costumam não durar muito tempo .



7 Braço Robótico



Agora chegamos em nossos dois últimos projetos, primeiro vamos fazer um braço robótico controlado por **Joystick** e depois pelo celular através do nosso **Módulo Bluetooth**.

Kit Braço Robótico 3D

Se você deseja ter um braço robótico de maior qualidade, nós temos um Kit com um braço feito com impressão 3D.

O Braço Robótico 3D trata-se de um braço mecânico desenvolvido para aplicação em projetos que trabalham com **automação de robôs**, proporcionando maiores possibilidades de movimentos e desenvolvimento de tarefas.

Por ser desenvolvido com tecnologia de impressão 3D, possui um **material resistente** e **design arrojado**.

Se você quer adquirir esse **Kit**, Clique Aqui

Servo Motor

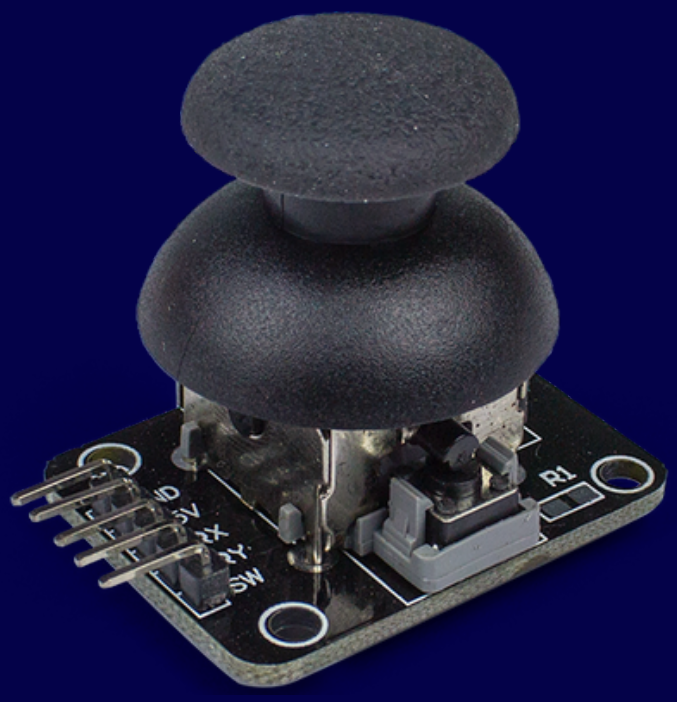
Os **servos motores** são dispositivos eletromecânicos utilizados para movimentar com precisão determinados objetos, permitindo-o girar em ângulos ou distâncias específicas, com garantia do posicionamento e velocidade.

Ele nada mais é que um motor elétrico rotativo acoplado a um sensor que passa a condição de seu posicionamento, permitindo o controle preciso da velocidade, aceleração e da posição angular.

O **servo motor** que utilizaremos é o **Micro Servo 9g SG90**, ele possui alta qualidade e é excelente para nossos projetos de robótica com Arduino.

Módulo Joystick

Este **Joystick Arduino** 3 Eixos tem seu princípio de funcionamento através do controle de 2 potenciômetros e um botão. Duas das entradas dos potenciômetros referem-se aos eixos X e Y, sendo que o botão quando pressionado refere-se ao eixo Z. Logo este Joystick contém o total de três interfaces de entradas que são utilizadas para conectar ao eixo X, Y e Z.



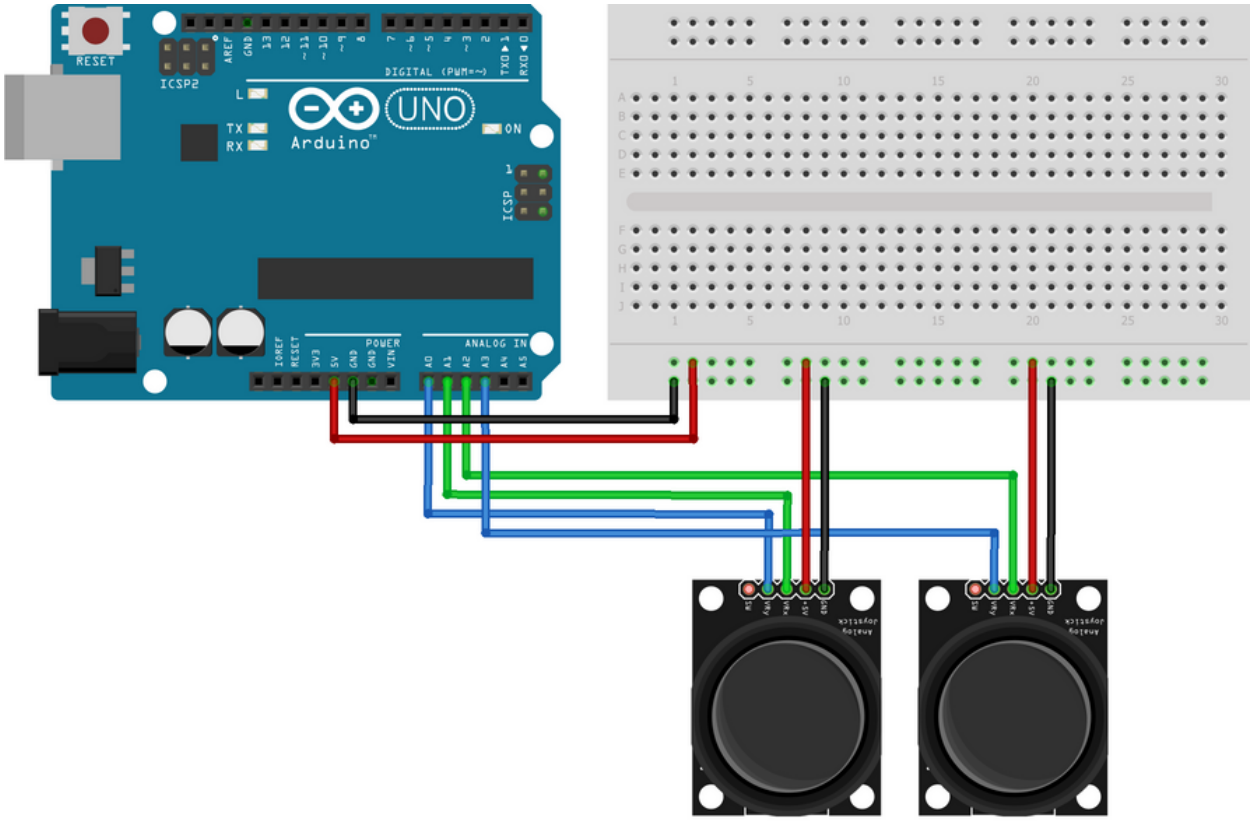
Este modelo com menos pinos torna-se uma ótima opção para projetos com Arduino pois evita muitas conexões, podendo ser usado como mouse, controle de robôs, games, joystick ps2 e projetos em geral.

Fonte Ajustável

A **Fonte Ajustável Protoboard** pode ser diretamente conectada a uma fonte DC para converter sua tensão para a saída em **3,3V** e em **5V**, tendo a opção de saída USB. Essas tensões são as mais comuns e usada por diversos componentes. Isto permite um fornecimento ideal para projetos dos mais variados, devido a sua estrutura feita para encaixar em um **Protoboard**.

Circuito

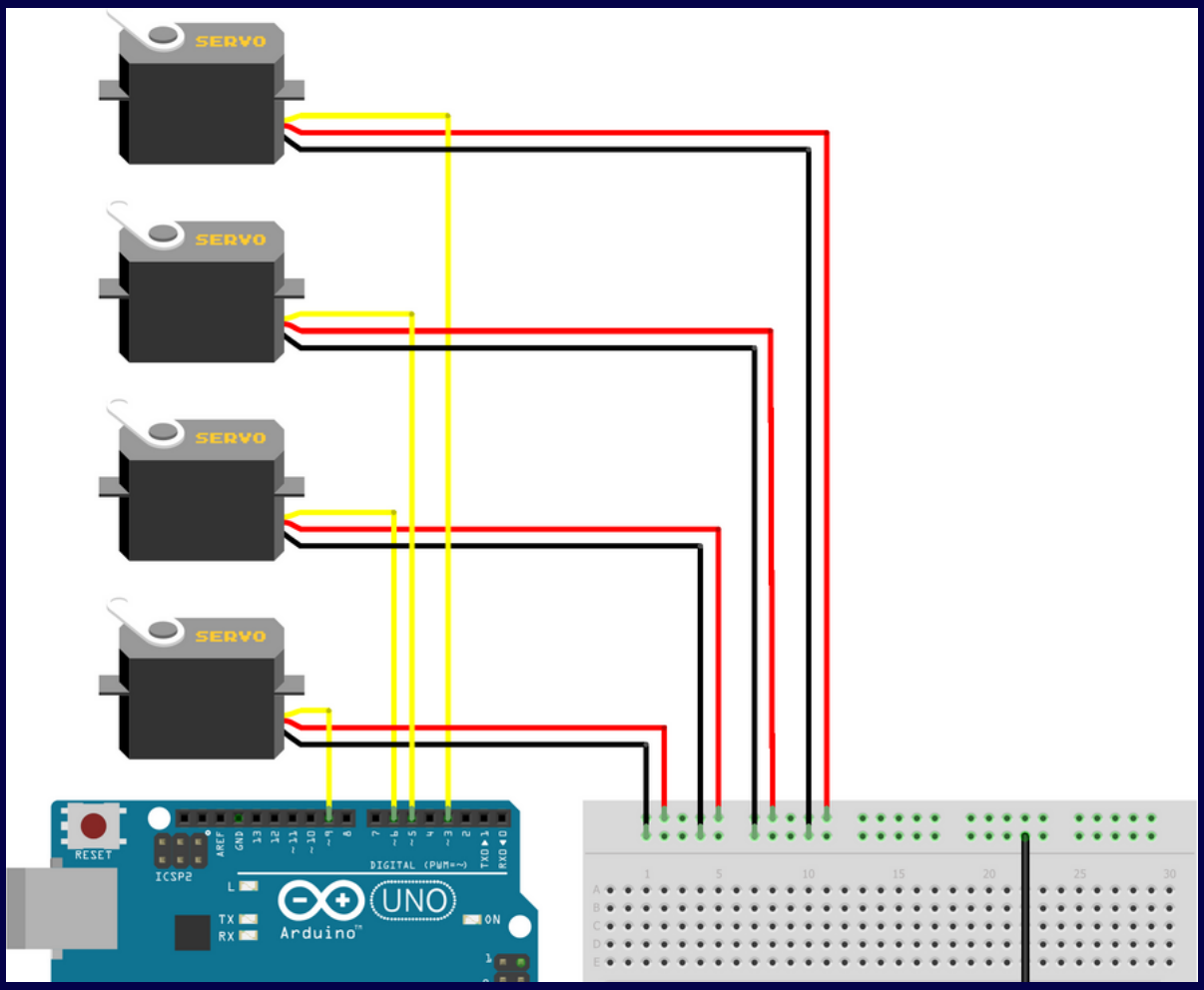
1 Conectando os Joysticks



Alimente o **GND** e o **VCC** dos joysticks com os **5V** e o **GND** da **protoboard** !!

Joystick Esquerdo	Joystick Direto
VRX ---- A1	VRX ---- A2
VRY ---- A0	VRY ---- A3

2 Conectando os Motores



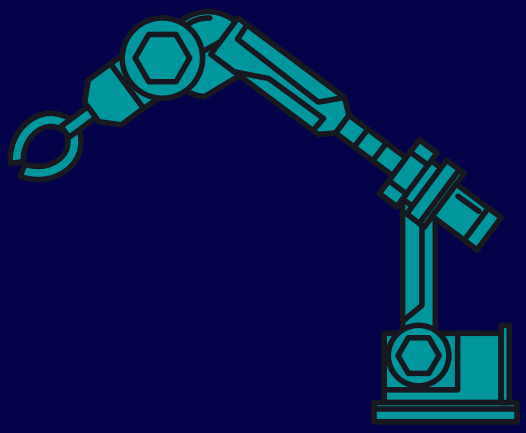
Conecte os fio preto no **GND** e o vermelho no **Positivo** da protoboard, porém não conecte ao 5V do Arduino, vamos utilizar uma **fonte externa de 9V**.

Motor **Garra:** **Porta 3**

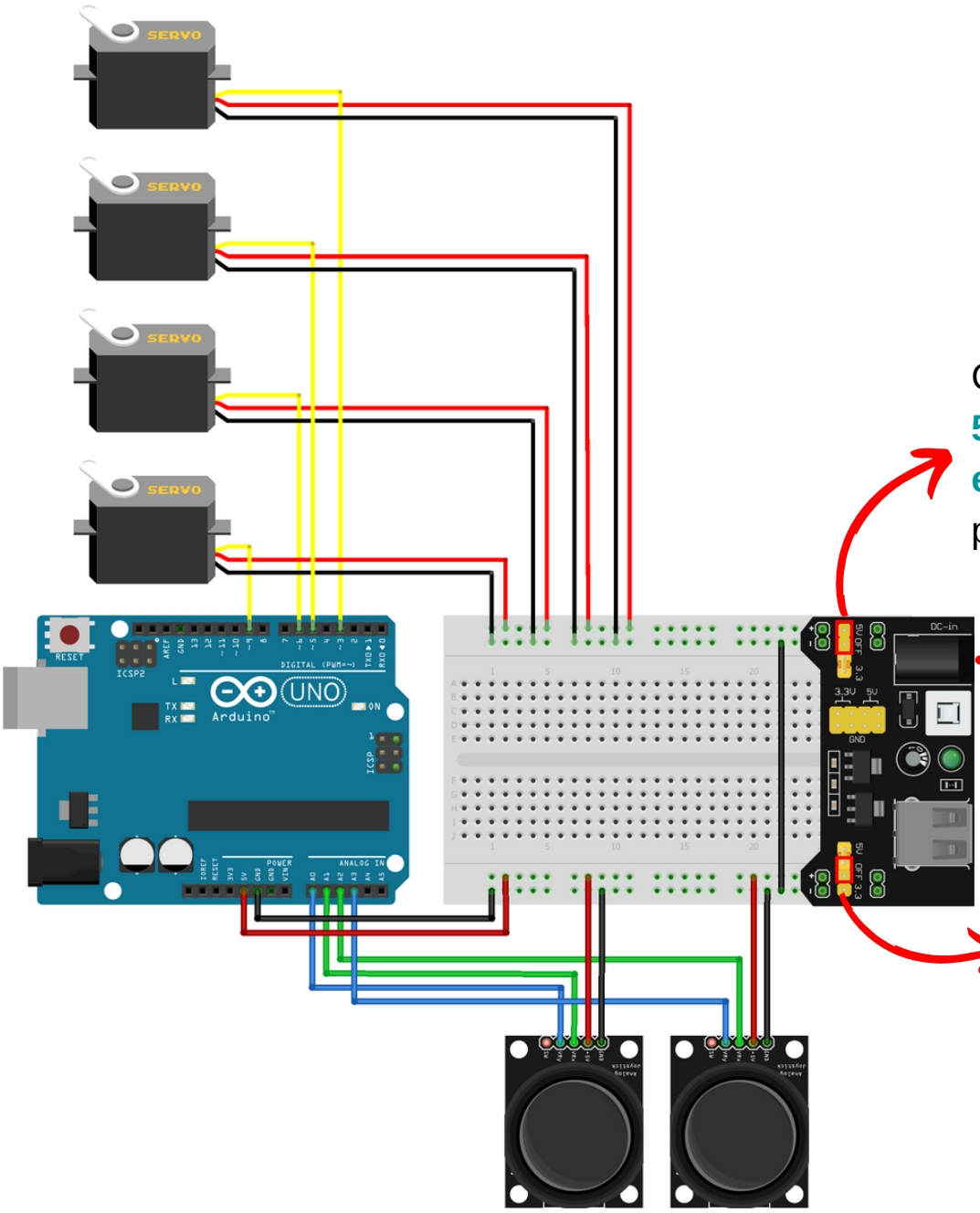
Motor **Elevação:** **Porta 5**

Motor **Avanço:** **Porta 6**

Motor **Giro** (base)**:** **Porta 9**



3 Alimentação dos Motores



Conecte o jumper em **5V**, vamos utilizar a **energia da Fonte** para os motores.

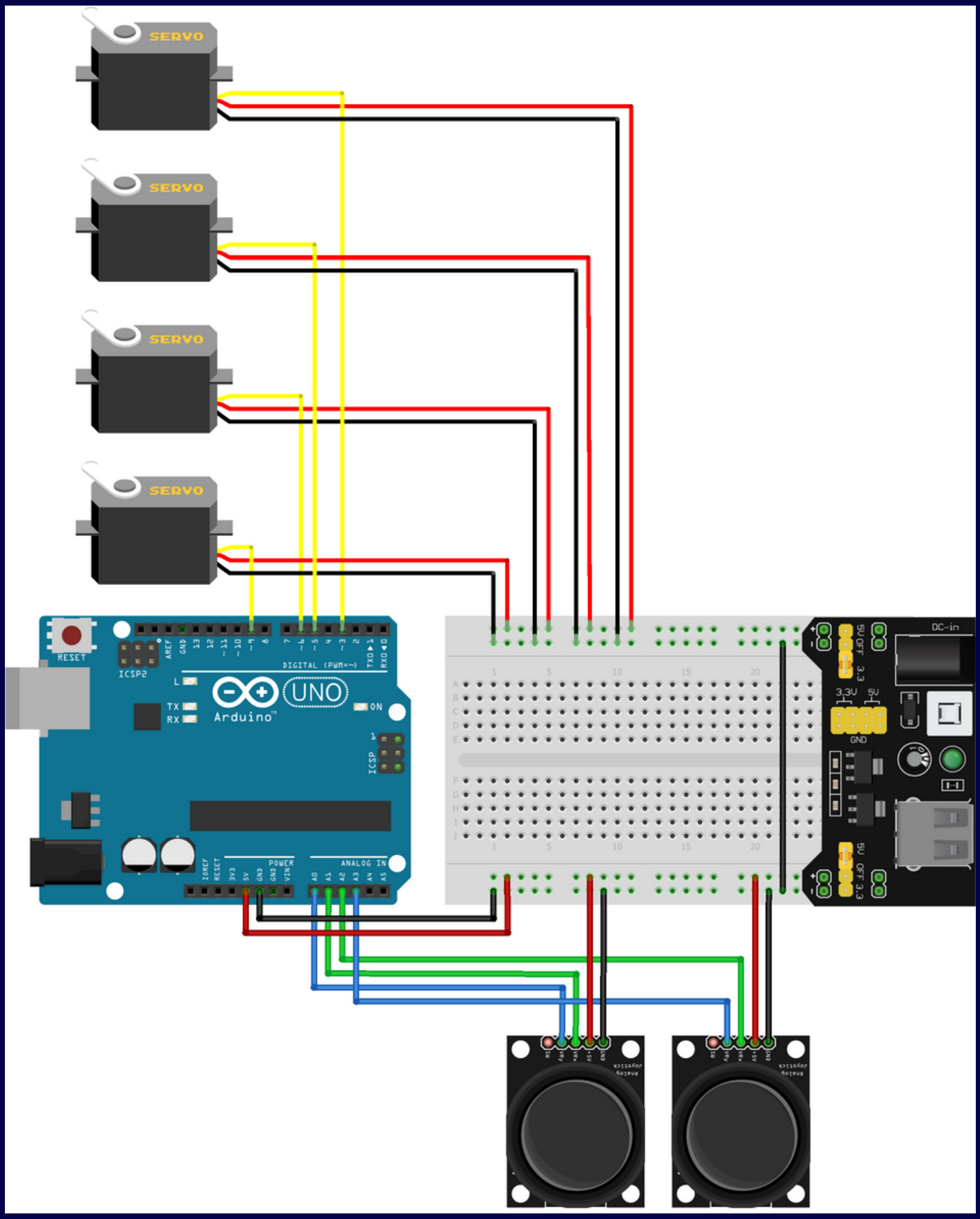
Conecte o **pino** da fonte aqui !

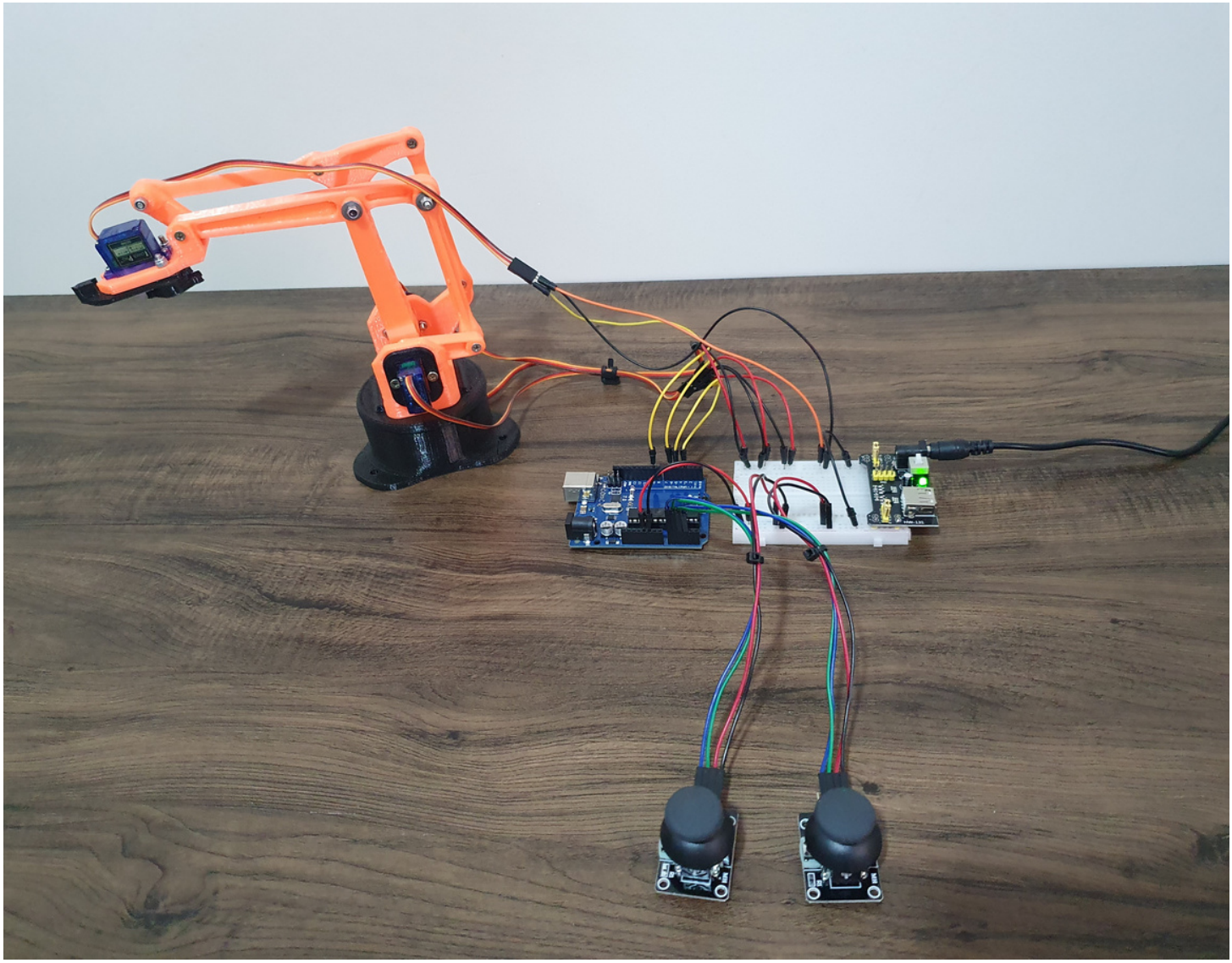
Conecte o jumper em **OFF**, vamos utilizar a **energia do Arduino** para os joysticks .

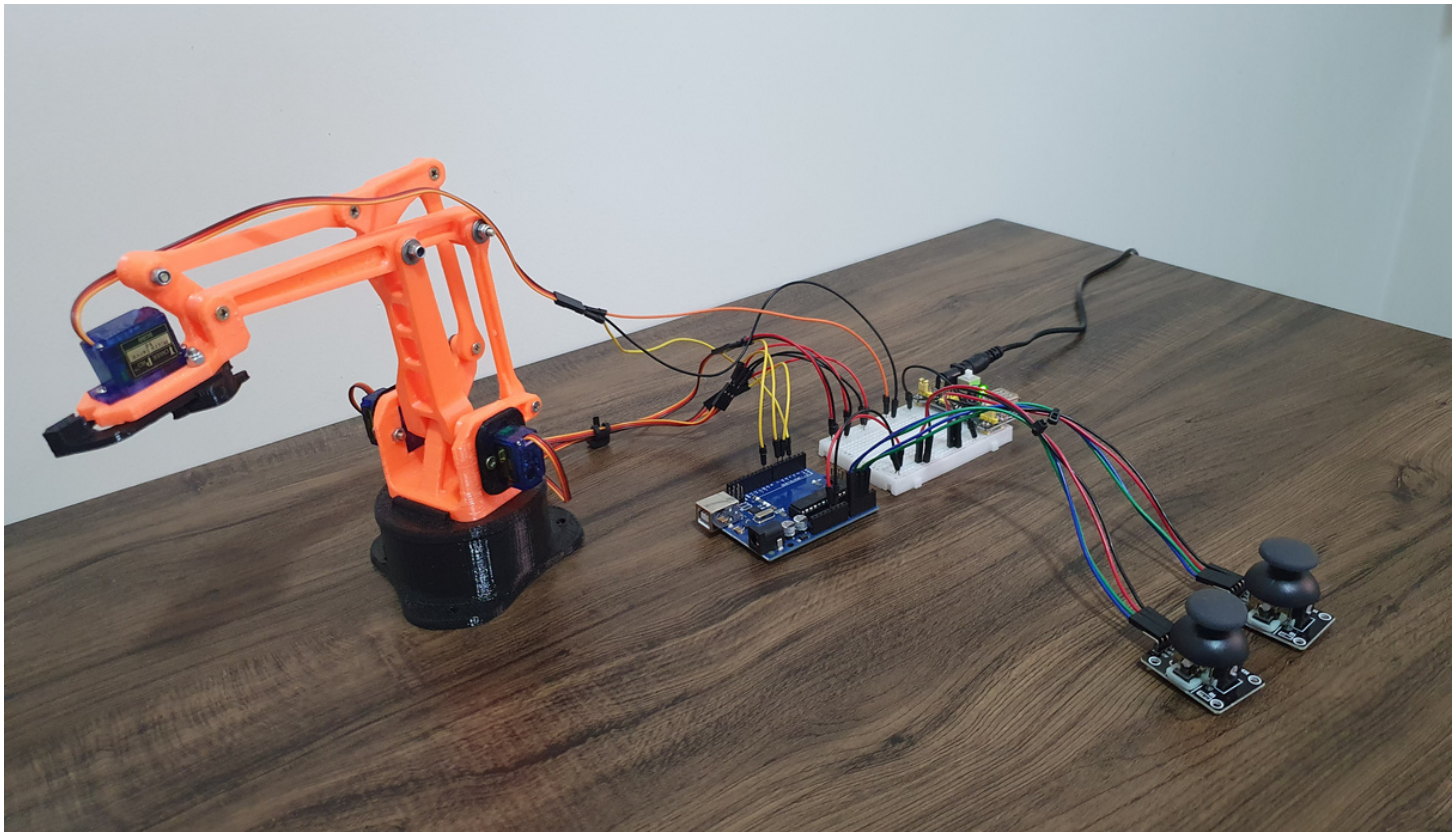
É importante conectarmos o **GND** do Arduino com o **GND** da fonte para o circuito funcionar corretamente, por isso temos um fio interligando os negativos ao lado da fonte ajustável. Para evitar **problemas** na montagem, siga o esquema acima !

Lembre-se: O Arduino deve ser alimentado via **cabo USB** no **computador**.

Circuito Completo







Programação

```
// Para usar a biblioteca VarSpeedServo

#include "VarSpeedServo.h"

// Definição dos objetos servos do braço robótico

VarSpeedServo servo1; //Base
VarSpeedServo servo2; //Extensão
VarSpeedServo servo3; //Altura
VarSpeedServo servo4; //Garra

// Definição das entradas analógicas do Joytick

// Joystick Esquerdo

int potpin1 = A0;//VRy
int potpin2 = A1;//VRx

// Joystick Direito

int potpin3 = A2;//VRx
int potpin4 = A3;//VRy

// Variáveis

int val1;
int val2;
int val3;
int val4;
```

```
// Valores iniciais de posição de cada servo

static int s1 = 70;
static int s2 = 110;
static int s3 = 100;
static int s4 = 80;

void setup()
{

  Serial.begin(9600); // Inicia a porta serial, configura a taxa de dados para 9600bps

 // Anexa o objeto servo ao pino

 servo1.attach(9); //Base, pino digital 9
 servo2.attach(6); //Extensão, pino digital 6
 servo3.attach(5); //Altura, pino digital 5
 servo4.attach(3); //Garra, pino digital 3

 // Move todo o braço para posição inicial

 servo1.write(70); //Base
 servo2.write(110); //Extensão
 servo3.write(100); //Altura
 servo4.write(80); //Garra
}

void loop()
{

 // Controle da base do braço

  val1 = analogRead(potpin1); // Lê o valor do pino analógico especificado (A0VRy, joystick Esquerdo)
```

```
// Para direita

 if (val1 < 100)
 {
   s1 = s1 - 2;
   if (s1 <= 10)
   {
     s1 = 10;
   }

 servo1.write(s1); // Posição em graus para o servo

 delay(50); // Espera 50 milessegundos

 }

// Para esquerda

 if (val1 > 900)
 {
   s1 = s1 + 2; // soma
   if (s1 >= 170)
   {
     s1 = 170;
   }

  servo1.write(s1);

  delay(50);
 }

 // Controle da extensão do braço

 val2 = analogRead(potpin2);
```

```
// Para trás

 if (val2 > 900)
 {
   s2 = s2 - 2;
   if (s2 <= 10)
   {
     s2 = 10;
   }
  servo2.write(s2);

  delay(50);
 }

 // Para frente

 if (val2 < 100)
 {
  s2 = s2 + 2;
  if (s2 >= 170)
 {
   s2 = 170;
 }
  servo2.write(s2);

  delay(50);
 }

 // Controle da altura do braço
 //Abaixar o braço

 val3 = analogRead(potpin3);
 if (val3 < 100)
 {
```

```
s3 = s3 - 2;
 if (s3 <= 10)
{
 s3 = 10;
}
 servo3.write(s3);
 delay(50);
}

//Levantar o braço
if (val3 > 900)
{
  s3 = s3 + 2;
  if (s3 >= 170)
  {
    s3 = 170;
  }
  servo3.write(s3);
  delay(50);
}

// Controle da garra do braço
val4 = analogRead(potpin4);
// Abrir a garra
if (val4 < 100)
{
  s4 = s4 - 2;
  if (s4 <= 80)
  {
   s4 = 80;
  }
 servo4.write(s4);
 delay(50);
}
```

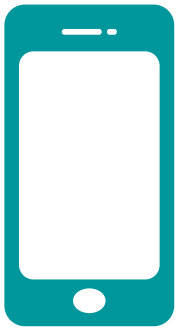
```
// Fechar a garra
 if (val4 > 900)
 {
   s4 = s4 + 2;
   if (s4 >= 130)
 {
   s4 = 130;
 }
 servo4.write(s4);
 delay(50);
 }

 // Exibe os valores analogicos na tela

 Serial.print(val1);
 Serial.print(" : ");
 Serial.print(val2);
 Serial.print(" : ");
 Serial.print(val3);
 Serial.print(" : ");
 Serial.print(val4);
 Serial.println();
}
```

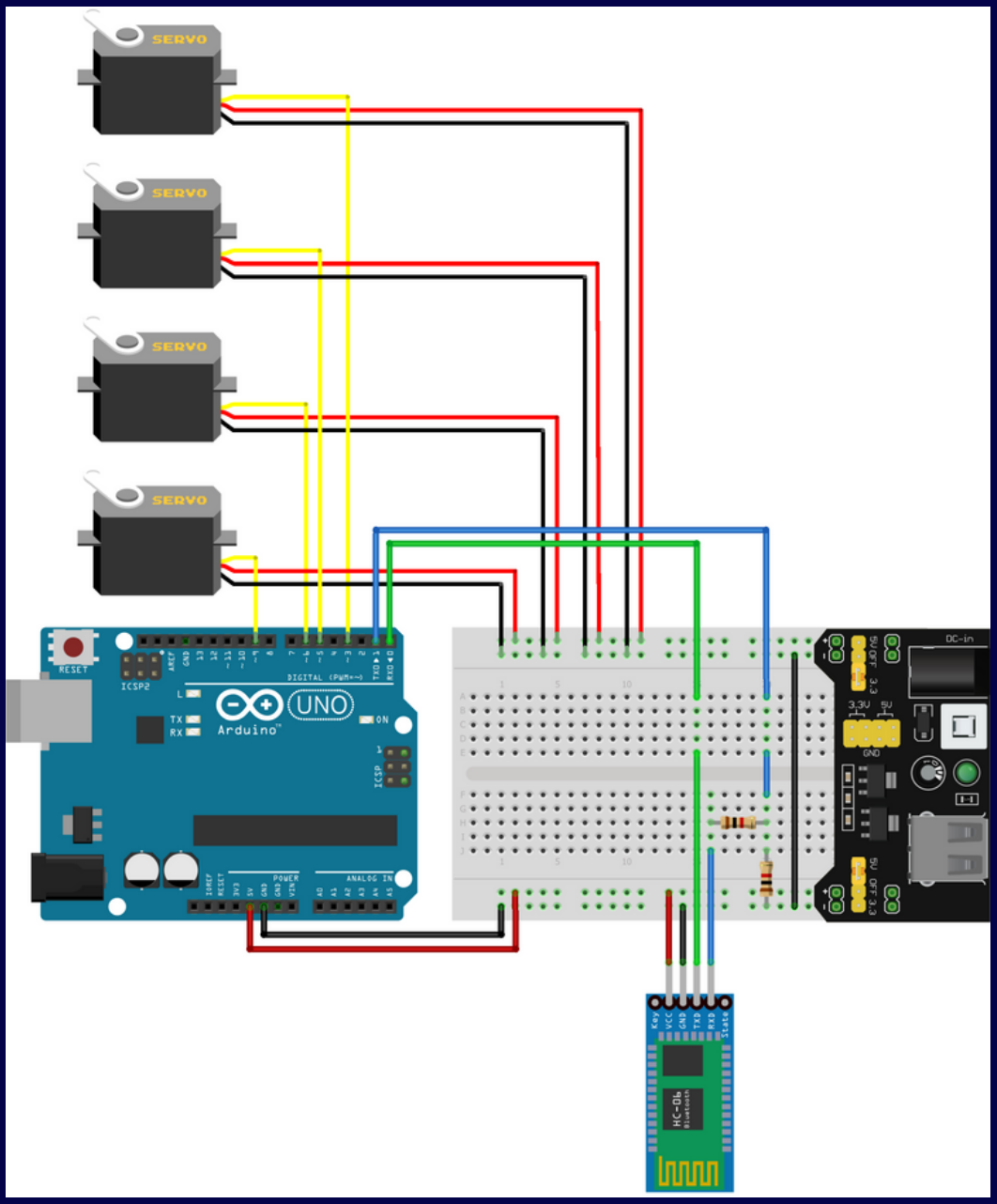
Baixe o código do projeto **Clicando Aqui**

Controlando Pelo Celular



Vamos utilizar nosso módulo **HC-06** para controlar o braço robótico pelo celular via **Bluetooth**, para isso é necessário remover os **joysticks** e colocar o módulo.

Circuito



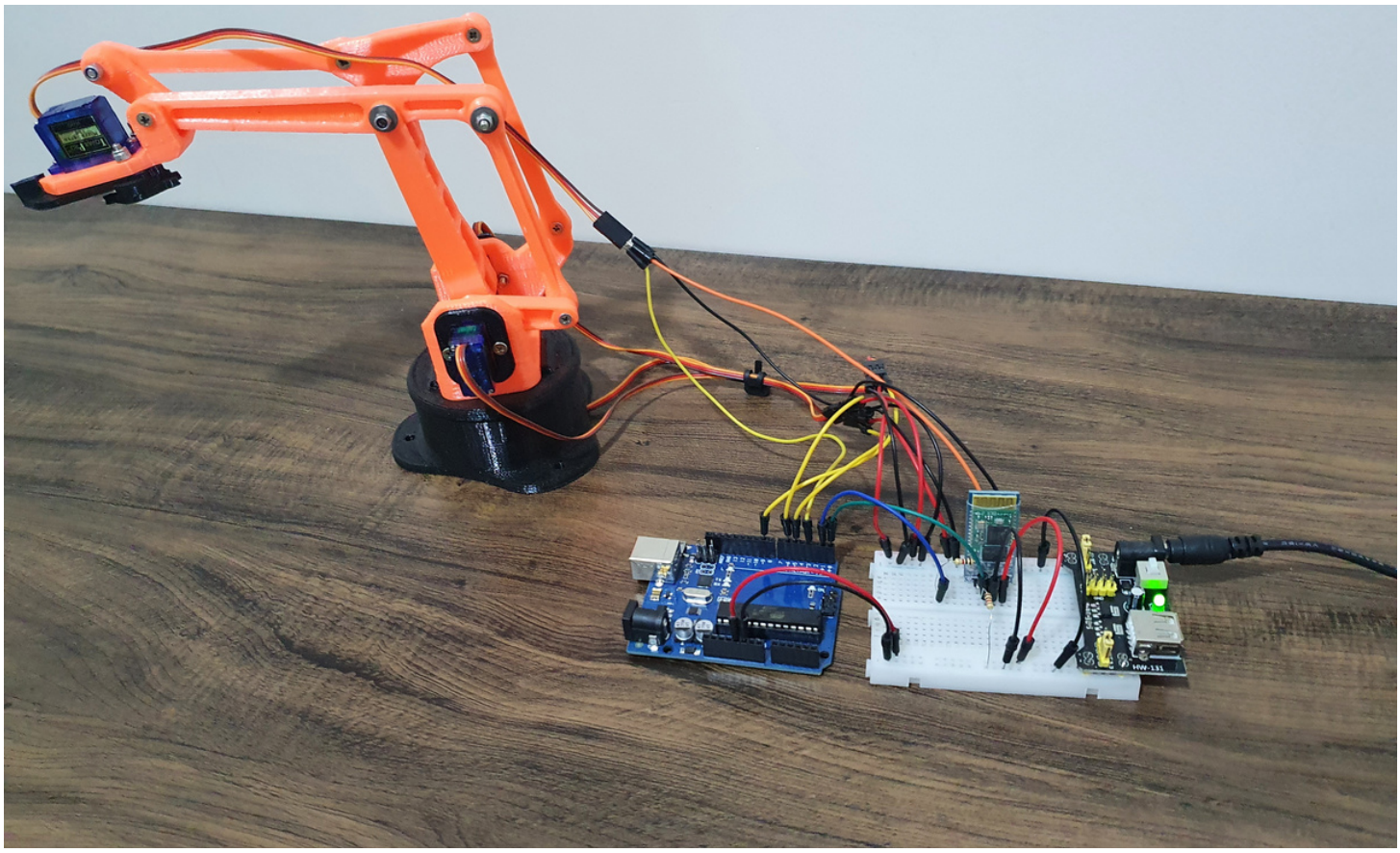
Vamos utilizar os **5V** e o **GND** da **protoboard** que vem do Arduino para alimentarmos nosso **módulo** !!

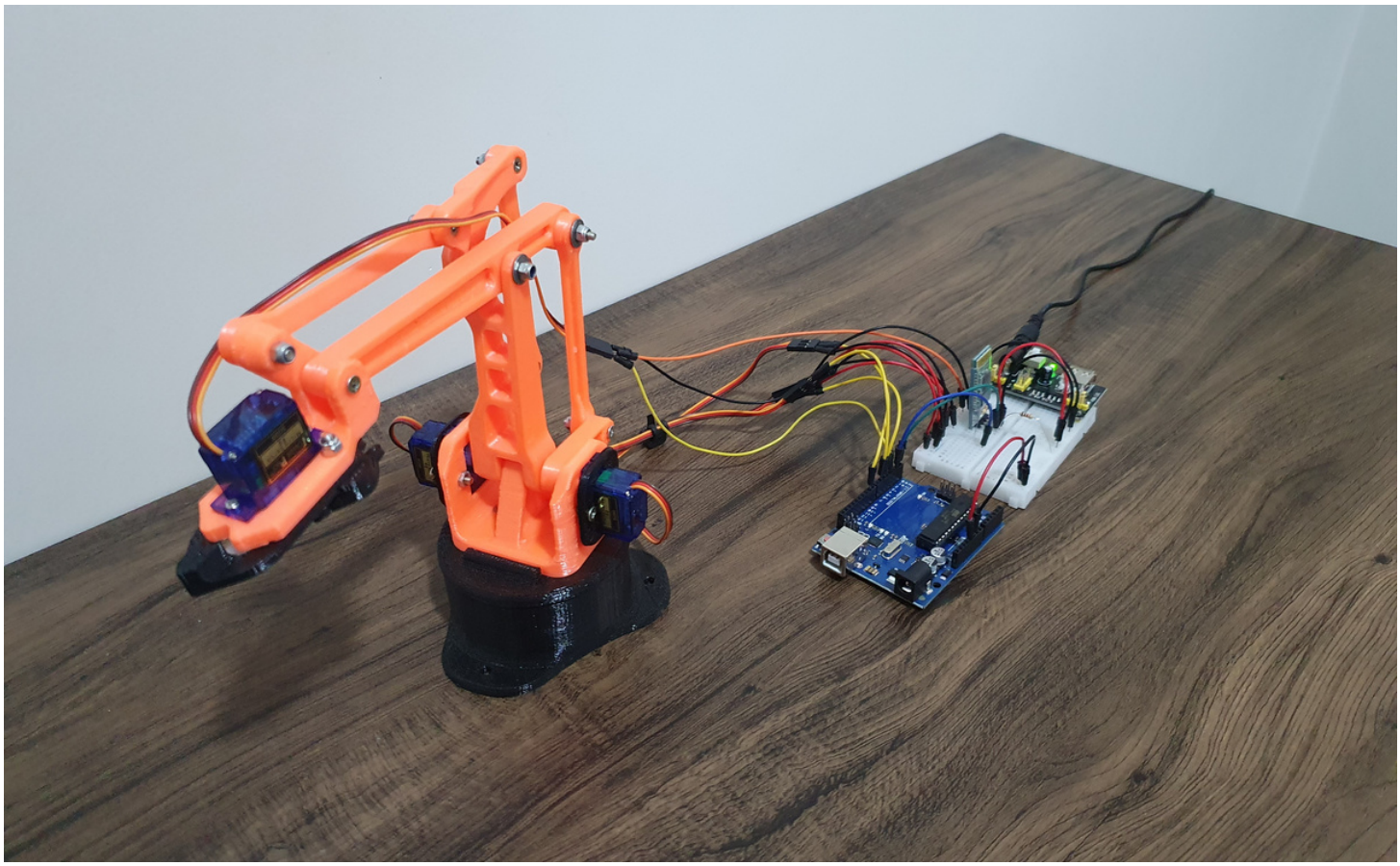
Lembre-se: O módulo **HC-06** opera em 3.3V, por isso é necessário fazer um **divisor de tensão** utilizando os resistores de **1KΩ**.

Ligações do Módulo:

TXD --- **RXD**
RXD --- **TXD**

Circuito na prática





Programação

```
#include <Servo.h> // Ativa a biblioteca do Servo

Servo garra, elevar, avancar, base; // nome dos servos

char comando; // Variável que recebe o comando vindo do aplicativo

void setup(){

 garra.attach(3); // Anexar o motor 01 ao pino 03 (GARRA)
 elevar.attach(5); // Anexar o motor 02 ao pino 05 (ELEVAÇÃO)
 avancar.attach(6); // Anexar o motor 03 ao pino 06 (AVANÇO)
 base.attach(9); // Anexar o motor 04 ao pino 09 (GIRO)

// Setup da conexão serial USB com o computador

 Serial.begin(9600);

// Posição inicial do Braço

 posInicial();
}
void loop(){

 if (Serial.available() >= 0 ){
   comando = Serial.read(); //Dado recebido do Aplicativo

  //Se o comando for  '1', , o braço avança

   if (comando == '1'){
     elevar.write(45);
     delay(10);
```

```
 avancar.write(130);
    delay(10);
    garra.write(30);
    delay(10);
  }

 // Se o comando for '2', o braço levanta

  if (comando == '2'){
    elevar.write(120);
    delay(10);
    avancar.write(60);
    delay(10);
    garra.write(0); //Escreve no servo o valor da posição
    delay(10);
  }
 // Se o comando for a letra 'D', de direita, o braço vira para direita
  if (comando == 'D') base.write(0);

 // Se o comando for a letra 'E', de esquerda, o braço vira para esquerda
  if (comando == 'E') base.write(180);

 // Se o comando for a letra 'I', de inicio, o braço vai para posição inicial
  if (comando == 'I') posInicial();
 }
}
//Função que posiciona o braço na posição inicial

void posInicial() {

 garra.write(0);
 elevar.write(90);
 avancar.write(90);
 base.write(90);
 delay(10);
}
```

Lembre-se...

Antes de enviar a programação para o Arduino, desconecte os pinos **RXD** e **TXD** do Arduino !!!

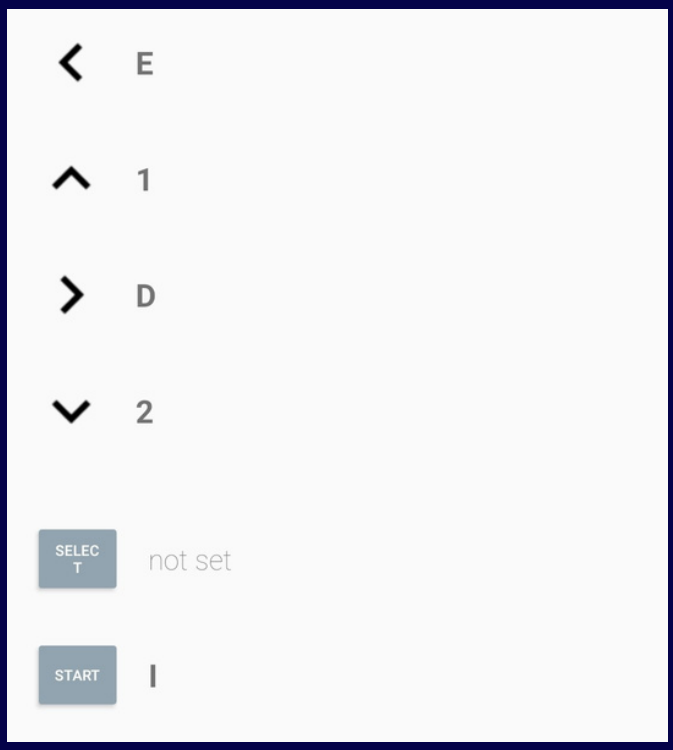
O cabo USB utiliza as mesmas portas para enviar a programação, caso elas estejam conectadas ao sensor, **não será possível enviar o código**.

Clique Aqui para baixar a programação!!!

Configurando o Aplicativo

Vamos utilizar o mesmo aplicativo que usamos no robô, só vamos alterar as **configurações**. Vá na página **67** caso queira ver o passo a passo para alterar as configurações dos botões.

OBS: Coloque as configurações iguais abaixo !



Esquerda = **E**

Avanço = **1**

Direita = **D**

Altura = **2**

Posição Inicial (Start) = **I**

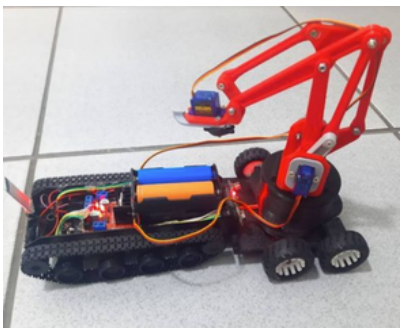
8 Finalização

Nessa apostila abordamos a **introdução a Robótica** para iniciantes, com vários projetos, nosso intuito foi desvendar um pouco sobre o mundo da robótica e te instigar a conhecer mais sobre essa área incrível e além disso, montar projetos utilizando o **Kit Arduino Robôs**. Sabemos que a robótica está se tornando cada vez mais necessária em nosso cotidiano, dessa forma é importante estudá-la.

Caso você tenha interesse em conhecer mais projetos de eletrônica usando o **Arduino**, a **Eletrônica Ômega** disponibiliza vários outros tutoriais de forma gratuita em nosso blog, temos vários outros tutorais para você mergulhar no mundo da programação !

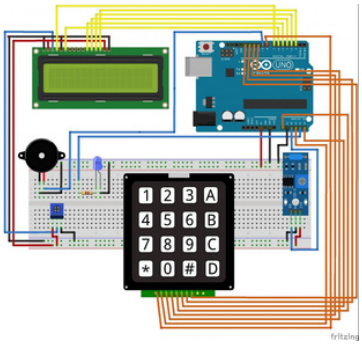
Veja abaixo exemplos de alguns tutoriais que você encontra no blog:

ELETRÔNICA ÔMEGA BLOG

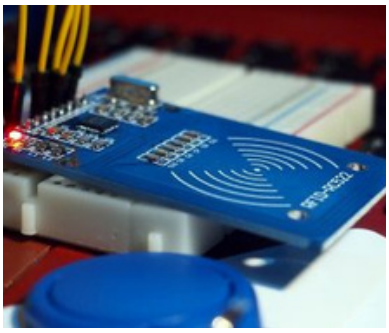


Tanque Reboque Bluetooth

Braço Robótico Controlado por voz



Sistema de Alarme Codificado



Sistema de Acesso com RF ID

Quem escreveu o E-book ?

Rangel Gabriel

Formado em **Eletroeletrônica** pelo **SENAI Shunji Nishimura**, tem experiência com eletrônica e microcontroladores.

E-mail: rangelarena@gmail.com

Eletrônica Ômega

A **Eletrônica Ômega** é uma loja virtual sediada em BH/MG, especializada em **Arduino** e **componentes eletrônicos**.

Temos a missão de **promover transformação cultural e social através da educação**, inserindo o maior número possível de pessoas no movimento maker, levando acesso a tecnologia para todos os interessados, e **mostrando que é possível desenvolver suas ideais através da tecnologia**.

Caso queria falar com a gente não deixe de mandar seu email para **contato@arduinoomega.com**.

Redes Sociais:

Referências

Tensão:

https://www.todamateria.com.br/tensao-eletrica/ **- acesso em 16/06/2021**

https://www.mundodaeletrica.com.br/tensao-eletrica-x-voltagem/ **- acesso em 16/06/2021**.

Corrente:

https://mundoeducacao.uol.com.br/fisica/corrente-eletrica.htm - **acesso em 16/06/2021**.

https://brasilescola.uol.com.br/fisica/corrente-eletrica.htm - **acesso em 16/06/2021**.

https://www.mundodaeletrica.com.br/o-que-e-corrente-eletrica/ - **acesso em 16/06/2021**.

Motor DC:

https://www.mundodaeletrica.com.br/motor-de-corrente-continua-caracteristicas-e-aplicacoes/ **- acesso em 23/06/2021**.

Ponte H:

https://www.arduinoportugal.pt/modulo-ponte-h-l298n-primeiro-passo-montar-robo-arduino/ **- acesso em 23/06/2021**.

Sensor Ultrassônico:

https://balluffbrasil.com.br/sensor-ultrassonico-como-ele-funciona-e-de-que-modo-pode-ajudar-a-sua-industria/ - **acesso em 23/06/2021.**

LED:

https://athoselectronics.com/o-que-e-led-diodo-emissor-luz/#:~:text=O%20LED%20%C3%A9%20um%20Diodo,energia%20el%C3%A9trica%20em%20energia%20luminosa.&text=Como%20se%20trata%20de%20um,corrente%20el%C3%A9trica%20em%20um%20sentido. **- acesso em 25/06/2021.**

Resistores:

https://brasilescola.uol.com.br/fisica/resistores.htm#:~:text=Resistores%20s%C3%A3o%20dispositivos%20usados%20para,esses%20s%C3%A3o%20conhecidos%20como%20diel%C3%A9tricos. **- acesso em 25/06/2021.**

Servo Motores:

https://blog.kalatec.com.br/o-que-e-servo-motor/ - **acesso em 15/07/2021.**