ELETROGATE

APOSTILA KIT

ARDUINO ROBÓTICA

Sumário

Parte I - Revisão de circuitos elétricos e componentes básicos

Introdução

A primeira parte da apostila faz uma revisão sobre circuitos elétricos, com destaque para questões práticas de montagem e segurança que surgem no dia-a-dia do usuário do Kit Arduino Robótica.

O conteúdo de circuitos elétricos aborda divisores de tensão e corrente, conversão de níveis lógicos, grandezas analógicas e digitais, níveis de tensão e cuidados práticos de montagem.
Em relação aos componentes básicos, é feita uma breve discussão sobre resistores elétricos, capacitores, leds, diodos, chaves e protoboards.

O Arduino UNO é o principal componente do Kit e é discutido e introduzido em uma seção à parte, na Parte II da apostila.

Todos os conteúdos da Parte I são focados na apostila Arduino Robótica, tendo em vista a utilização adequada dos componentes e da realização prática de montagens pelos usuários. No entanto, recomenda-se a leitura das referências indicadas ao final de cada seção para maior aprofundamento.

O leitor veterano, já acostumado e conhecedor dos conceitos essenciais de eletrônica e eletricidade, pode pular a Parte I e ir direto a Parte II, na qual são apresentadas uma seção de exemplo de montagem para cada sensor ou componente importante da apostila.

Algum conhecimento prévio é bem-vindo, mas não é necessário para utilização do kit. Caso seja seu primeiro contato, nós recomendamos que antes de fazer as montagens você leia esses três artigos do http://blog.eletrogate.com/

- https://blog.eletrogate.com/o-que-e-arduino-para-que-serve-vantagens-e-como-utilizar/
- *http://blog.eletrogate.com/programacao-arduino-parte-1/*
- *http://blog.eletrogate.com/programacao-arduino-parte-2/*

Preparado? Vamos começar!

Revisão de circuitos elétricos

A apostila Arduino Robótica, bem como todas as outras apostilas que tratam de Arduino e eletrônica em geral, tem como conhecimento de base as teorias de circuitos elétricos e de eletrônica analógica e digital.

Do ponto de vista da teoria de circuitos elétricos, é importante conhecer os conceitos de grandezas elétricas: Tensão, corrente, carga, energia potência elétrica. Em todos os textos sobre Arduino ou qualquer assunto que envolva eletrônica, você sempre terá que lidar com esses termos. Para o leitor que se inicia nessa seara, recomendamos desde já que mesmo que a eletrônica não seja sua área de formação, que conheça esses conceitos básicos.

Vamos começar pela definição de "circuito elétrico". *Um circuito elétrico/eletrônico é uma interconexão de elementos elétricos/eletrônicos*. Essa interconexão pode ser feita para atender a uma determinada tarefa, como acender uma lâmpada, acionar um motor, dissipar calor em uma resistência e tantos outras.

O circuito pode estar **energizado** ou **desenergizado**. Quando está energizado, é quando uma fonte de tensão externa ou interna está ligada aos componentes do circuito. Nesse caso, uma corrente elétrica fluirá entre os condutores do circuito. Quando está desenergizado, a fonte de tensão não está conectada e não há corrente elétrica fluindo entre os condutores.

Mas atenção, alguns elementos básicos de circuitos, como os capacitores ou massas metálicas, são elementos que armazenam energia elétrica. Em alguns casos, mesmo não havendo fonte de tensão conectada a um circuito, pode ser que um elemento que tenha energia armazenada descarregue essa energia dando origem a uma corrente elétrica transitória no circuito. Evitar que elementos do circuito fiquem energizados mesmo sem uma fonte de tensão, o que pode provocar descargas elétricas posteriores (e em alguns casos, danificar o circuito ou causar choques elétricos) é um dos motivos dos sistemas de aterramento em equipamentos como osciloscópios e em instalações residenciais, por exemplo.

Em todo circuito você vai ouvir falar das grandezas elétricas principais, assim, vamos aprender o que é cada uma delas.

Carga e corrente elétrica

A grandeza mais básica nos circuitos elétricos é a carga elétrica. Carga é a propriedade elétrica das partículas atômicas que compõem a matéria, e é medida em Coulombs. Sabemos da física elementar que a matéria é constituída de elétrons, prótons e neutros. **A carga elementar é a carga de 1 elétron, que é igual a 1,602 x10-19 C**.

Do conceito de carga elétrica advém o **conceito de corrente elétrica, que nada mais é do que a taxa de variação da carga em relação ao tempo**, ou seja, quando você tem um fluxo de carga em um condutor, a quantidade de carga (Coulomb) que atravessa esse condutor por unidade de tempo, é chamada de corrente elétrica. A medida utilizada para corrente é o **Ampére(A).**

Aqui temos que fazer uma distinção importante. Existem corrente elétrica contínua e alternada:

- **Corrente elétrica contínua:** É uma corrente que permanece constante e em uma única direção durante todo o tempo.
- **Corrente elétrica alternada**: É uma corrente que varia senoidalmente (ou de outra forma) com o tempo.

Com o Arduino UNO, lidamos como correntes elétricas contínuas, pois elas fluem sempre em uma mesma direção. É diferente da corrente e tensão elétrica da tomada de sua casa, que são alternadas.

Ou seja, os seus circuitos com Arduino UNO sempre serão alimentados com grandezas contínuas (corrente e tensão contínuas).

Tensão elétrica

Para que haja corrente elétrica em um condutor, é preciso que os elétrons se movimentem por ele em uma determinada direção, ou seja, é necessário "alguém" para transferir energia para as cargas elétricas para movê-las. Isso é feito por uma força chamada **força eletromotriz (fem),** tipicamente representada por uma bateria. Outros dois nomes comuns para força eletromotriz são **tensão elétrica** e **diferença de potencial.**

O mais comum é você ver apenas "*tensão*" nos artigos e exemplos com Arduino. Assim, definindo formalmente o conceito: Tensão elétrica é a energia necessária para mover uma unidade de carga através de um elemento, e é medida em **Volts (V).**

Potência e energia

A tensão e a corrente elétrica são duas grandezas básicas, e juntamente com a potência e energia, são as grandezas que descrevem qualquer circuito elétrico ou eletrônico. A potência é definida como a variação de energia (que pode estar sendo liberada ou consumida) em função do tempo, e é medida em **Watts (W).** A potência está associada ao calor que um componente está dissipando e a energia que ele consume.

Nós sabemos da vida prática que uma lâmpada de 100W consome mais energia do que uma de 60 W. Ou seja, se ambas estiverem ligadas por 1 hora por exemplo, a lâmpada de 100W vai implicar numa conta de energia mais cara.
A potência se relaciona com a tensão e corrente pela seguinte equação:

$$P = V \times I$$

Essa é a chamada potência instantânea. Com essa equação você saber qual a potência dissipada em um resistor por exemplo, bastando que saiba a tensão nos terminais do resistor e a corrente que flui por ele. O conceito de potência é importante pois muitos hobbystas acabam não tendo noção de quais valores de resistência usar, ou mesmo saber especificar componentes de forma adequada.

Um resistor de 33 ohms de potência igual a 1/4W, por exemplo, não pode ser ligado diretamente em circuito de 5V, pois nesse caso a potência dissipada nele seria maior que a que ele pode suportar.

Vamos voltar a esse assunto em breve, por ora, tenha em mente que é importante ter uma noção da potência dissipada ou consumida pelos elementos de um circuito que você vá montar.

Por fim, a energia elétrica é o somatório da potência elétrica durante todo o tempo em que o circuito esteve em funcionamento. A energia é dada em **Joules (J)** ou **Wh (watt-hora).** A unidade Wh é interessante pois mostra que a energia é calculada multiplicando-se a potência pelo tempo (apenas para os casos em que a potência é constante).

Essa primeira parte é um pouco conceitual, mas é importante saber de onde vieram toda a terminologia que você sempre vai ler nos manuais e artigos na internet. Na próxima seção, vamos discutir os componentes básicos de circuito que compõem o Kit Arduino Robótica.

Conversão de níveis lógicos e alimentação correta de circuitos

É muito comum que hobbistas e projetistas em geral acabem por cometer alguns erros de vez em quando. Na verdade, mesmo alguns artigos na internet e montagens amplamente usadas muitas vezes acabam por não utilizar as melhores práticas de forma rigorosa. Isso acontece muito no caso dos níveis lógicos dos sinais usados para interfacear o Arduino com outros circuitos.

Como veremos na seção de apresentação do Arduino UNO, o mesmo é alimentado por um cabo USB ou uma fonte externa entre 6V e 12V. O Circuito do Arduino possui reguladores de tensão que convertem a alimentação de entrada para 5V e para 3V. Os sinais lógicos das portas de saída(I/Os) do Arduino, variam entre 0 e 5V.

Isso significa que quando você quiser usar o seu Arduino UNO com um sensor ou CI que trabalhe com 3.3V, é preciso fazer a adequação dos níveis de tensão, pois se você enviar um sinal de 5V (saída do Arduino) em um circuito de 3.3V (CI ou sensor), você poderá queimar o pino daquele componente.
Em geral, sempre que dois circuitos que trabalhem com níveis de tensão diferentes forem conectados, é preciso fazer a conversão dos níveis lógicos. O mais comum é ter que abaixar saídas de 5V para 3.3V. Subir os sinais de 3.3V para 5V na maioria das vezes não é necessário pois o Arduino entende 3.3V como nível lógico alto, isto é, equivalente a 5V.

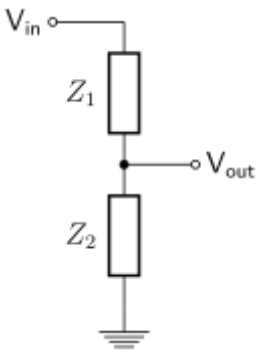
Para fazer a conversão de níveis lógicos você tem duas opções:

- Usar um divisor de tensão;
- Usar um CI conversor de níveis lógicos;

O divisor de tensão é a solução mais simples, mas usar um CI conversor é mais elegante e é o ideal. O divisor de tensão consiste em dois resistores ligados em série (Z1 e Z2) , em que o sinal de 5V é aplicado a o terminal de um deles. O terminal do segundo resistor é ligado ao GND, e o ponto de conexão entre os dois resistores é a saída do divisor, cuja tensão é dada pela seguinte relação:

$$V_{\text{out}} = \frac{Z_2}{Z_1 + Z_2} \cdot V_{\text{in}}$$

Em que Z1 e Z2 são os valores dos resistores da figura abaixo.



Um divisor de tensão muito comum é fazer Z1 igual 330 ohms e Z2 igual 680 ohms. Dessa forma a sáida Vout fica sendo 3.336 V. Como no Kit Arduino Robótica não há resistor de 680ohms, você pode ligar um de 330ohms como Z1 e dois de 330ohms como Z2. Os dois de 330 ligados em série formam um resistor de 660 ohm, o que resulta numa saída de 3.33V.

Vamos exemplificar como fazer um divisor de tensão como esse na seção de exemplos da parte II da apostila.

Revisão de componentes eletrônicos

O Kit Arduino Robótica possui os seguintes componentes básicos para montagens de circuitos:

- Buzzer Ativo 5V,
- LED Vermelho/ Verde/ Amarelo,
- Resistor 330Ω/ 1KΩ/ 10KΩ,
- Diodo 1N4007,
- Potenciômetro 10KΩ,
- Capacitor Cerâmico 10 nF/ 100 nF,
- Capacitor Eletrolítico 10uF/ 100uF,
- Chave Táctil (Push-button).

Vamos revisar a função de cada um deles dentro de um circuito eletrônica e apresentar mais algumas equações fundamentais para ter em mente ao fazer suas montagens.

Resistores elétricos

Os resistores são componentes que se opõem à passagem de corrente elétrica, ou seja, oferecem uma resistência elétrica. Dessa forma, quanto maior for o valor de um resistor,

menor será a corrente elétrica que fluirá por ele e pelo condutor a ele conectada. A unidade de resistência elétrica é o Ohm(Ω), que é a unidade usada para especificar o valor dos resistores.

Os resistores mais comuns do mercado são construídos com fio de carbono e são vendidos em várias especificações. Os resistores do Kit são os tradicionais de 1/4W e 10% de tolerância. Isso significa que eles podem dissipar no máximo 1/4W (0,25 watts) e seu valor de resistência pode variar em até 10%. O resistor de 1KΩ pode então ter um valor mínimo de 900Ω e um valor máximo de 1100Ω.

Em algumas aplicações você pode precisar de resistores com precisão maior, como 5% ou 1%. Em outros casos, pode precisar de resistores com potência maior, como 1/2W ou menor, como 1/8W. Essas variações dependem da natureza de cada circuito.

Em geral, para as aplicações típicas e montagens de prototipagem que podem ser feitos com o Kit Arduino Robótica, os resistores tradicionais de 1/4W e 10% de tolerância são suficientes.

Outro ponto importante de se mencionar aqui é a Lei de Ohm, que relaciona as grandezas de tensão, corrente e resistência elétrica. A lei é dada por:

$$V = R \times I$$

Ou seja, se você sabe o valor de um resistor e a tensão aplicada nos seus terminais, você pode calcular a corrente elétrica que fluirá por ele. Juntamente com a equação para calcular potência elétrica, a lei de Ohm é importante para saber se os valores de corrente e potência que os resistores de seu circuito estão operando estão adequados.

Para fechar, você deve estar se perguntando, como saber o valor de resistência de um resistor? Você tem duas alternativas: Medir a resistência usando um multímetro ou determinar o valor por meio do código de cores do resistor.

Se você pegar um dos resistores do seu kit, verá que ele possui algumas faixas coloridas em seu corpo. Essas faixas são o código de cores do resistor. As duas primeiras faixas dizem os dois primeiros algarismos decimais. A terceira faixa colorida indica o multiplicador que devemos usar. E a última faixa, que fica um pouco mais afastada, indica a tolerância.

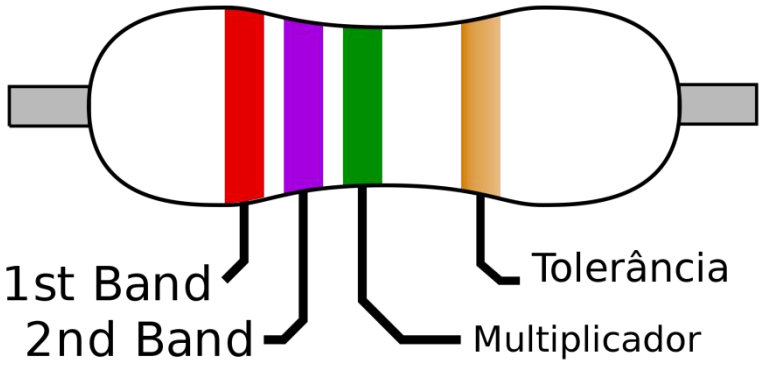


Figura 1: Faixas coloridas em um resistor

Na figura 2 apresentamos o código de cores para resistores. Cada cor está associada a um algarismo, um multiplicador e uma tolerância, conforme a tabela. Com a tabela você pode determinar a resistência de um resistor sem ter que usar o multímetro.

Mas atenção, fazer medições com o multímetro é recomendado principalmente se o componente já tiver sido utilizado, pois o mesmo pode ter sofrido algum dano ou mudança que não esteja visível.

Cor	Dígito	Multiplicador	Tolerância
Prata	-	x 0,01	± 10%
Dourado	-	x 0,1	± 5%
Preto	0	x 1	-
Marrom	1	x 10	± 1%
Vermelho	2	x 100	± 2%
Laranja	3	x 1K	-
Amarelo	4	x 10K	-
Verde	5	x 100K	± 0,5%
Azul	6	x 1M	± 0,25%
Violeta	7	x 10M	± 0,1%
Cinza	8	-	± 0,05%
Branco	9	-	-

Figura 2: Código de cores para resistores

Aplicando a tabela da figura 2 na imagem da figura 1, descobrimos que o resistor é de 2,7MΩ (Mega ohms) com tolerância de 5% (relativo à cor dourado da última tira).

Capacitores

Os capacitores são os elementos mais comuns nos circuitos eletrônicos depois dos resistores. São elementos que armazenam energia na forma de campos elétricos. Um

capacitor é constituído de dois terminais condutores e um elemento dielétrico entre esses dois terminais, de forma que quando submetido a uma diferença de potencial, um campo elétrico surge entre esses terminais, causando o acúmulo de cargas positivas no terminal negativo e cargas negativas no terminal positivo.

São usados para implementar filtros, estabilizar sinais de tensão, na construção de fontes retificadores e várias outras aplicações.

O importante que você deve saber para utilizar o Kit é que os capacitores podem ser de quatro tipos:

- Eletrolíticos,
- Cerâmicos,
- Poliéster,
- Tântalo.

Capacitor eletrolítico

As diferenças de cada tipo de capacitor são a tecnologia construtiva e o material dielétrico utilizado. Capacitores eletrolíticos são feitos de duas longas camadas de alumínio (terminais) separadas por uma camada de óxido de alumínio (dielétrico). Devido a sua construção, eles possuem **polaridade,** o que significa que você obrigatoriamente deve ligar o terminal positivo (quem tem uma "perninha" maior) no polo positivo da tensão de alimentação, e o terminal negativo (discriminado no capacitor por uma faixa com símbolos de "-") obrigatoriamente no polo negativo da bateria. Do contrário, o capacitor será danificado.

Capacitores eletrolíticos costumam ser da ordem de micro Farad, sendo o **Farad** a unidade de medida de capacitância, usada diferenciar um capacitor do outro. A Figura 3 ilustra um típico capacitor eletrolítico.

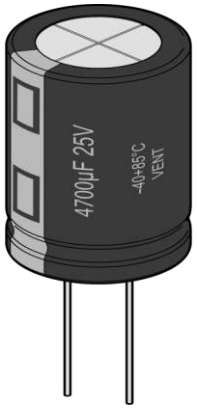


Figura 3: Capacitor eletrolítico 4700 micro Farads / 25 V

Capacitores cerâmicos

Capacitores cerâmicos não possuem polaridade, e são caracterizados por seu tamanho reduzido e por sua cor característica, um marrom claro um tanto fosco. Possuem capacitância da ordem de pico Farad. Veja nas imagens abaixo um típico capacitor cerâmico e sua identificação:

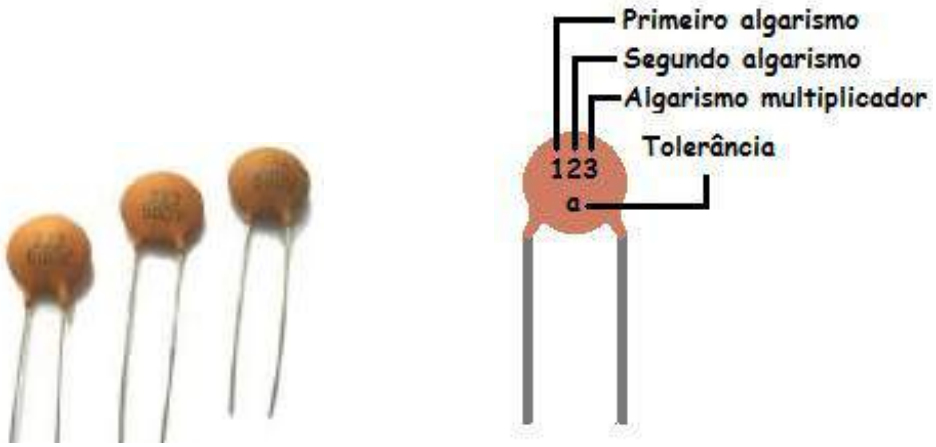


Figura 4: Capacitores cerâmicos

Na imagem da esquerda, os capacitores possuem o valor de 22 nano Farads para a faixa de tensão de até 500V.

$$223 = 22 \times 1000 = 22.000\ pF = 22\ nF$$

Por fim, há também os capacitores de poliéster e de tântalo. No Kit Arduino Robótica, você receberá apenas exemplares de capacitores eletrolíticos e cerâmicos.

Diodos e Leds

Diodos e Leds são tratados ao mesmo tempo pois tratam na verdade do mesmo componente. Diodos são elementos semicondutores que só permitem a passagem de corrente elétrica em uma direção.

São constituídos de dois terminais, o Anodo (+) e o catodo (-), sendo que para que possa conduzir corrente elétrica, é preciso conectar o Anodo na parte positiva do circuito, e o Catodo na parte negativa. Do contrário, o diodo se comporta como um circuito aberto.

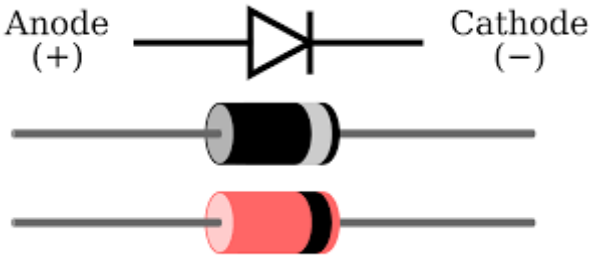


Figura 4: Diodo e seus terminais

Na figura 4, você pode ver que o componente possui uma faixa indicadora do terminal de catodo. O diodo do kit Arduino Robótica é um modelo tradicional e que está no mercado há muitos anos, o 1N4007.

O LED é um tipo específico de diodo - **Light Emitter Diode**, ou seja, diodo emissor de luz. Trata-se de um diodo que quando polarizado corretamente, emite luz para o ambiente externo. O Kit Arduino BEGINNIG vem acompanhado de leds na cor vermelha, verde e amarelo, as mais tradicionais.

Nesse ponto, é importante você saber que sempre deve ligar um led junto de um resistor, para que a corrente elétrica que flua pelo led não seja excessiva e acabe por queimá-lo. Além disso, lembre-se que por ser um diodo, o led só funciona se o Anodo estiver conectado ao polo positivo do sinal de tensão.

Para identificar o Anodo do Led, basta identificar a perna mais longa do componente, como na imagem abaixo:

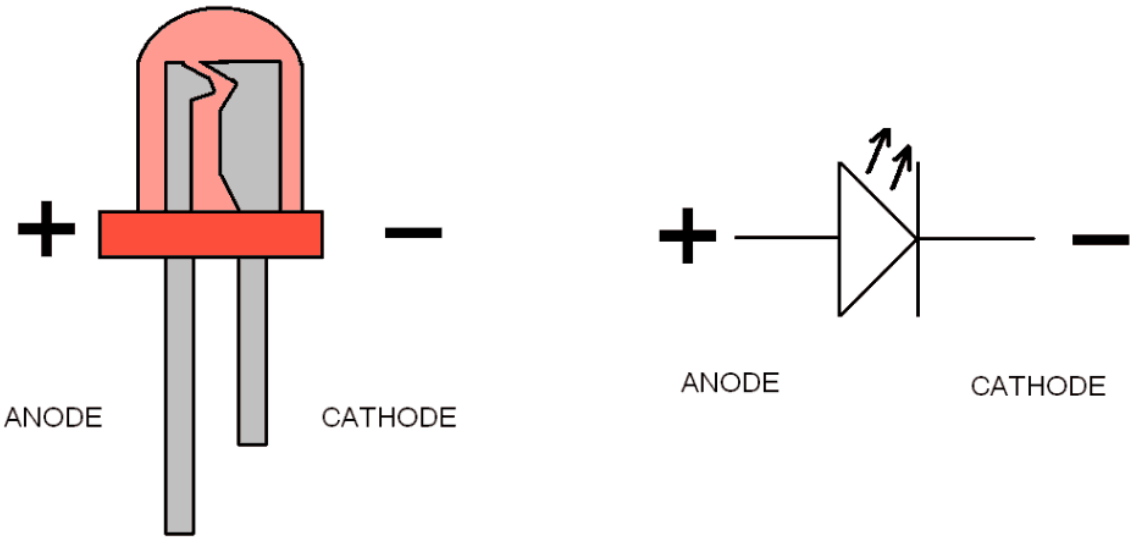


Figura 5: Terminais de um Led. Créditos: Build-eletronic-circuits.com

Os demais componentes da apostila, Buzzer, potenciômetro e push-buttons, serão explicados em seções de exemplos, nas quais iremos apresentar um circuito prático para montagem onde será explicado o funcionamento dos mesmos.

Os sensores do Kit Arduino Robótica, quais sejam: Sensor de Luz LDR e sensor de temperatura NTC, juntamente com o Micro Servo 9g SG90 TowerPro, também terão uma seção de exemplo dedicada a cada um deles.

Parte II - Seção de Exemplos Práticos

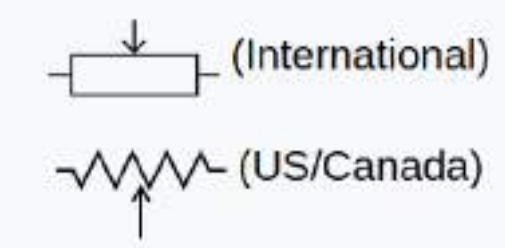
Agora vamos entrar nas seções de exemplos em si. Os conceitos da Parte I são importantes caso você esteja começando a trabalhar com eletrônica. No entanto, devido ao aspecto prático da montagem, não necessariamente você precisa de ler toda a parte introdutória para montar os circuitos abaixo.

Em cada exemplo vamos apresentar os materiais utilizados, o diagrama de circuito e o código base com as devidas explicações e sugestões de referências.

Exemplo 1 - Usando potenciômetro para fazer leituras analógicas

O potenciômetro nada mais do que um resistor cujo valor de resistência pode ser ajustado de forma manual. Existem potenciômetros slides e giratórios. Na figura abaixo mostramos um potenciômetro giratório dos mais comumente encontrados no mercado.

Em ambos, ao variar a posição da chave manual, seja ela giratória ou slide, o valor da resistência entre o terminal de saída e um dos outros terminais se altera. O símbolo do potenciômetro é mostrado na seguinte imagem:



Nesse exemplo, vamos ligar a saída de um potenciômetro a uma entrada analógica da Arduino UNO. Dessa forma, vamos ler o valor de tensão na saída do potenciômetro e vê-la variando de
0 a 1023. Mas como assim, 0 a 1023?

Isso se deve ao seguinte. Vamos aplicar uma tensão de 5V nos terminais do potenciômetro. A entrada analógica do Arduino, ao receber um sinal de tensão externo, faz a conversão do mesmo para um valor digital, que é representado por um número inteiro de 0 a 1023. Esses números são assim devido à quantidade de bits que o conversor analógico digital do Arduino trabalha, que são 10 bits (2 elevado a 10 = 1024).

Ou seja, o Arduino divide o valor de tensão de referência em 1024 unidades (0 a 1023) de 0,00488 volts. Assim, se a tensão lida na entrada analógica for de 2,5V, o valor capturado pelo Arduino será metade de 2,5/0,00488 = 512. Se for 0V, será 0, e ser for 5V, será 1023, e assim proporcionalmente para todos os valores. Assim, digamos que o valor de tensão se dado por V. O valor que o Arduino vai te mostrar é:

$$Valor = (V/5) * 1024$$

Em que 5V é o valor de referência configurado no conversor analógico-digital (uma configuração já padrão, não se preocupe com ela) e 1024 é igual 2 elevado a 10.

No nosso código, queremos saber o valor de tensão na saída do potenciômetro, e não esse número de 0 a 1023, assim, rearranjar a equação para o seguinte:

$$Tensão = Valor * (5/1024)$$

Bacana, né? Agora, vamos à montagem em si.

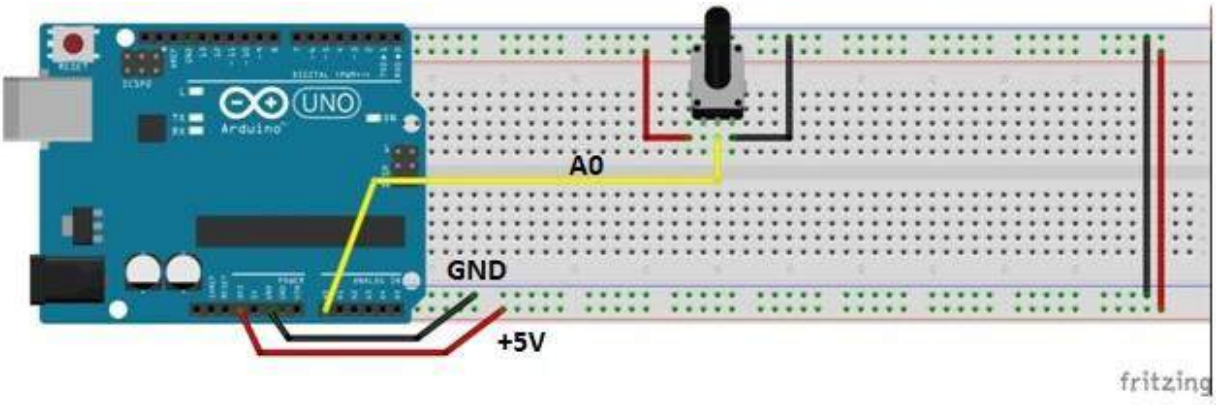
Lista de materiais:

Para esse exemplo você vai precisar:

- Arduino UNO;
- Protoboard;
- Potenciômetro 10K;
- Jumpers de ligação;

Diagrama do circuito

Monte o circuito conforme diagrama abaixo e carregue o código de exemplo:



O Potenciômetro possui 3 terminais, sendo que o do meio é o que possui resistência variável. A ligação consiste em ligar os dois terminais fixos a uma tensão de 5V. Assim, o terminal intermediário do potenciômetro terá um valor que varia de 0 a 5V à medida que você gira seu knob.

O terminal intermediário é ligado diretamente a uma entrada analógica do Arduino (A0). Como a tensão é de no máximo 5V, então não há problema em ligar direto.

Carregue o código abaixo no Arduino e você verá as leituras no Monitor serial da IDE Arduino.

```
// Exemplo 1 - Usando potenciometro para fazer leituras analógicas
// Apostila Eletrogate - KIT ROBÓTICA

#define sensorPin A0      // define entrada analógica A0

int sensorValue = 0;      // variável inteiro igual a zero
float voltage;            // variável numero fracionario

void setup()
{
  Serial.begin(9600);     // monitor serial - velocidade 9600 Bps
  delay(100);             // atraso de 100 milisegundos
}

void loop()
{
  sensorValue = analogRead(sensorPin);       // leitura da entrada analógica A0
  voltage = sensorValue * (5.0 / 1024);      // cálculo da tensão

  Serial.print("Tensão do potenciometro: "); // imprime no monitor serial
  Serial.print(voltage);                     // imprime a tensão
  Serial.print("    Valor: ");               // imprime no monitor serial
  Serial.println(sensorValue);               // imprime o valor
  delay(500);                                // atraso de 500 milisegundos
}
```

No código acima, nós declaramos a variável **sensorValue** para armazenar as leituras da entrada analógica A0 e a variável do tipo **float Voltage** para armazenar o valor lido convertido para tensão.

Na função **void Setup()**, nós inicializamos o terminal serial com uma taxa de transmissão de 9600 kbps. Na função **void Loop(),** primeiro faz-se a leitura da entrada analógica A0 com a função **analogRead(SensorPin)** e armazenamos a mesma na variável **sensorValue**. Em seguida, aplicamos a fórmula para converter a leitura (que é um número entre 0 e 1023) para o valor de tensão correspondente. O resultado é armazenado na variável **Voltage** e em seguido mostrado na interface serial da IDE Arduino.

Referência:

- https://www.arduino.cc/en/Tutorial/ReadAnalogVoltage

Exemplo 2 - Divisor de tensão

Esse exemplo é para ilustrar como usar um divisor de tensão. Sempre que você precisar abaixar um sinal lógico de 5V para 3.3V você pode usar esse circuito. Explicamos o divisor de tensão na seção de introdução, mais especificamente, quando conversamos sobre conversão de níveis lógicos.

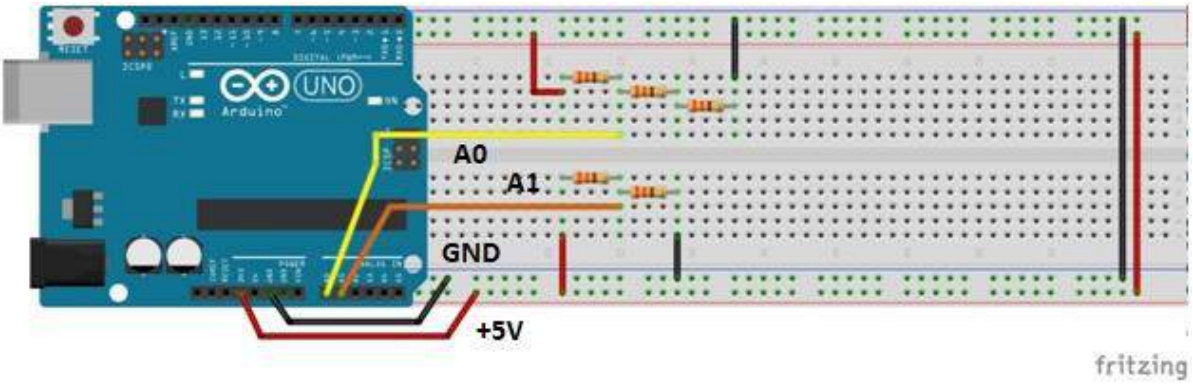
Esse circuito será útil sempre que você tiver que abaixar as saídas do Arduino de 5V para 3.3V.

Lista de materiais:

Para esse exemplo você vai precisar:

- 3 resistores de 330 ohms;
- 2 resistores de 1K ohm;
- 1 Arduino UNO;
- Protoboard;
- Jumpers de ligação;

Diagrama do circuito:



Esse é um diagrama com dois divisores de tensão. O primeiro é composto por três resistores, sendo cada um de 330. Assim, o resistor Z1 é de 330 e o resistor Z2 é a associação série dos outros dois, resultando numa resistência de 660. Dessa forma, a tensão de saída do divisor é:

$$(660 / 330 + 660) * 5 = 3,33V$$

O segundo divisor é formado por dois resistores de 1K, dessa forma, a tensão de saída é a tensão de entrada dividida pela metade:

$$(1000 / 1000 + 1000) * 5 = 2,5\ V$$

O código para o exemplo 2 é uma extensão do código usada na seção anterior para ler valores de tensão do potenciômetro. Nesse caso, nós fazemos a leitura de dois canais analógicos (A0 e A1), fazemos as conversões para as tensões e mostramos os resultados de cada divisor na interface serial.

Referências:

- https://www.arduino.cc/en/Tutorial/ReadAnalogVoltage
- http://www.ufjf.br/fisica/files/2013/10/FIII-05-07-Divisor-de-tens%C3%A3o.pdf
- http://www.sofisica.com.br/conteudos/Eletromagnetismo/Eletrodinamica/associacaoderesistores.php

Carregue o código abaixo e observe os dois valores no Monitor Serial da IDE Arduino.

```
// Exemplo 2 - Divisor de tensão
// Apostila Eletrogate - KIT ROBÓTICA

#define sensorPin1 A0           // define entrada analógica A0
#define sensorPin2 A1           // define entrada analógica A1

int sensorValue1 = 0;           // variavel inteiro igual a zero
int sensorValue2 = 0;           // variavel inteiro igual a zero
float voltage1;                 // variavel numero fracionario
float voltage2;                 // variavel numero fracionario

void setup()
{
  Serial.begin(9600);           // monitor serial - velocidade 9600 Bps
  delay(100);                   // atraso de 100 milisegundos
}

void loop()
{
  sensorValue1 = analogRead(sensorPin1);   // leitura da entrada analógica A0
  sensorValue2 = analogRead(sensorPin2);   // leitura da entrada analógica A1
  voltage1 = sensorValue1 * (5.0 / 1024);  // cálculo da tensão 1
  voltage2 = sensorValue2 * (5.0 / 1024);  // cálculo da tensão 2
  Serial.print("Tensao do divisor 1: ");   // imprime no monitor serial
  Serial.print(voltage1);                  // imprime a tensão 1
  Serial.print("  Tensao do divisor 2: "); // imprime no monitor serial
  Serial.println(voltage2);                // imprime a tensão 2
  delay(500);                              // atraso de 500 milisegundos
}
```

Exemplo 3 - Entradas e saídas digitais - push-button + led

Os push-buttons (chaves botão) e leds são elementos presentes em praticamente qualquer circuito eletrônico. As chaves são usadas para enviar comandos para o Arduino e os Leds são elementos de sinalização luminosa.

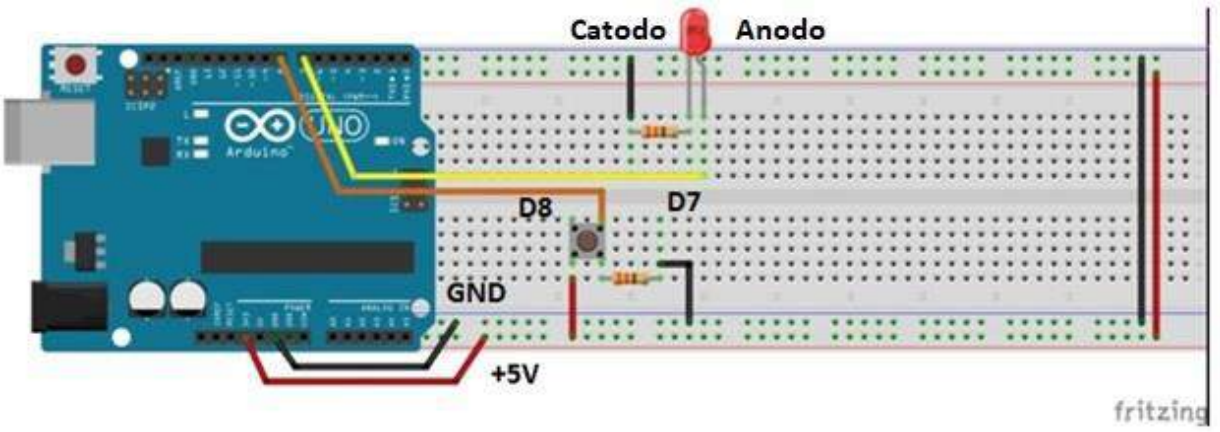
Esses dois componentes são trabalhados por meio das entradas e saídas digitais do Arduino. Neste exemplo vamos fazer uma aplicação básica que você provavelmente vai repetir muitas vezes. Vamos ler o estado de um push-button e usá-la para acender ou apagar um led. Ou seja, sempre que o botão for acionado, vamos apagar ou acender o Led.

Lista de materiais:

Para esse exemplo você vai precisar:

- 2 resistores de 330 ohms;
- 1 Led vermelho (ou de outra cor de sua preferência);
- Push-button (chave botão);
- 1 Arduino UNO;
- Protoboard;
- Jumpers de ligação;

Diagrama do circuito:



Esse pequeno código abaixo mostra como ler entradas digitais e com acionar as saídas digitais. Na função **void Setup(),** é preciso configurar qual pino será usado como saída e qual será usado como entrada.

Depois de configurar os pinos, para acioná-los basta chamar a função **digitalWrite(pino,HIGH).** A função **digitalWrite()** aciona ou desaciona um pino digital dependendo do valor passado no argumento. Se for "HIGH", o pino é acionado. Se for "LOW", o pino é desligado.

Na função **void Loop(),** fizemos um if no qual a função **digitalRead** é usada para saber se o pushButton está acionado ou não. Caso ele esteja acionado, nós acendemos o Led, caso ele esteja desligado, nós desligamos o led.

Carregue o código abaixo e pressione o botão para acender o LED.

```
// Exemplo 3 - Entradas e saídas digitais - push-button + led
// Apostila Eletrogate - KIT ROBÓTICA

#define PinButton 8                    // define pino digital D8
#define ledPin 7                       // define pino digital D7

void setup()
{
  pinMode(PinButton, INPUT);           // configura D8 como entrada digital
  pinMode(ledPin, OUTPUT);             // configura D7 como saída digital
  Serial.begin(9600);                  // monitor serial - velocidade 9600 Bps
  delay(100);                          // atraso de 100 milisegundos
}

void loop()
{
  if ( digitalRead(PinButton) == HIGH)  // se chave = nivel alto
  {
    digitalWrite(ledPin, HIGH);         // liga LED com 5V
    Serial.print("Acendendo Led");      // imprime no monitor serial
  }
  else                                  // senão chave = nivel baixo
  {
    digitalWrite(ledPin, LOW);          // desliga LED com 0V
    Serial.print("Desligando led");     // imprime no monitor serial
  }
  delay(100);                           // atraso de 100 milisegundos
}
```

Exemplo 4 - Sensor de luz LDR

O sensor LDR é um sensor de luminosidade. LDR é um **Light Dependent Resistor**, ou seja, um resistor cuja resistência varia com a quantidade de luz que incide sobre ele. Esse é seu princípio de funcionamento.

Quanto maior a luminosidade em um ambiente, menor a resistência do LDR. Essa variação na resistência é medida através da queda de tensão no sensor, que varia proporcionalmente (de acordo com a lei Ohm, lembra?) com a queda na resistência elétrica.

A imagem abaixo mostra o sensor em mais detalhes:

Fotoresistor (LDR)

É importante considerar a potência máxima do sensor, que é de 100 mW. Ou seja, com uma tensão de operação de 5V, a corrente máxima que pode passar por ele é 20 mA. Felizmente, com 8K ohms (que medimos experimentalmente com o ambiente bem iluminado), que é a resistência mínima, a corrente ainda está longe disso, sendo 0,625mA. Dessa forma, podemos interfacear o sensor diretamente com o Arduino.

**Nota: Nas suas medições, pode ser que você encontre um valor de resistência mínimo diferente, pois depende da iluminação local.*

Especificações do LDR:

- Modelo: GL5528
- Diâmetro: 5mm
- Tensão máxima: 150VDC
- Potência máxima: 100mW

- Temperatura de operação: -30°C a 70°C
- Comprimento com terminais: 32mm
- Resistência no escuro: 1 MΩ (Lux 0)
- Resistência na luz: 10-20 KΩ (Lux 10)

Este sensor de luminosidade pode ser utilizado em projetos com Arduino e outros microcontroladores para alarmes, automação residencial, sensores de presença e vários outros.

Nesse exemplo, vamos usar uma entrada analógica do Arduino para ler a variação de tensão no LDR e, consequentemente, saber como a luminosidade ambiente está se comportando. Veja na especificação que com muita luz, a resistência fica em torno de 10-20 KΩ, enquanto no escuro pode chegar a 1MΩ.

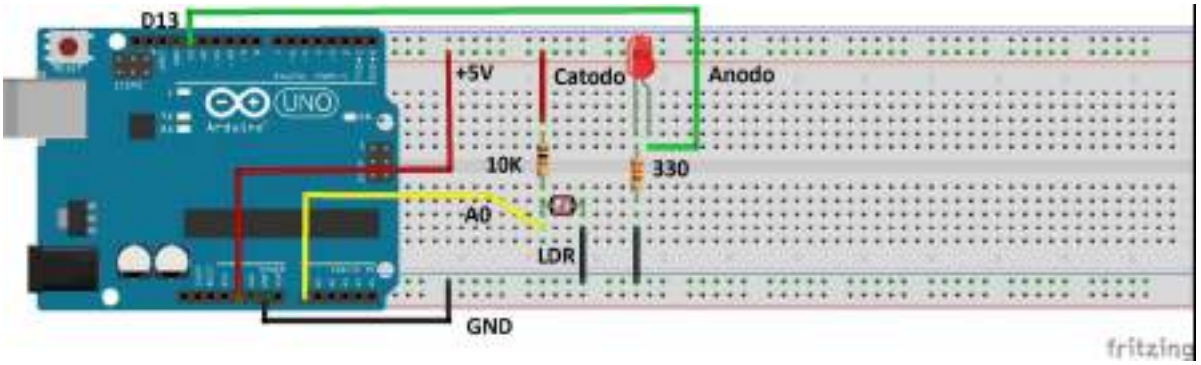
Para podermos ler as variações de tensão resultantes da variação da resistência do LDR, vamos usar o sensor como parte de um divisor de tensão. Assim, a saída do divisor será dependente apenas da resistência do sensor, pois a tensão de entrada e a outra resistência são valores conhecidos. No nosso caso, vamos usar um resistor de 10K e uma tensão de operação de 5V. Assim, o sinal que vamos ler no Arduino terá uma variação de 2,2V (quando o LDR for 8K) e 5V (quando o LDR tiver resistências muito maiores que o resistor de 10K).

Lista de materiais:

Para esse exemplo você vai precisar:

- LDR;
- Resistor de 10k;
- 1 Arduino UNO;
- Protoboard;
- Jumpers de ligação;

Diagrama do circuito:



No diagrama, o sensor é ligado como parte de um divisor de tensão no pino analógico A0, de forma que a tensão de saída do divisor varia de acordo com a variação da resistência do sensor. Assim, vamos identificar as variações na intensidade de luz pelas variações na tensão do sensor.

Quanto maior a intensidade de luz, menor a resistência do sensor e, consequentemente, menor a tensão de saída.

Carregue o código abaixo e varie a luminosidade sobre o LDR.

```
// Exemplo 4 - Sensor de luz LDR
// Apostila Eletrogate - KIT ROBÓTICA

#define AnalogLDR A0                    // define pino analógico A0
#define Limiar 1.5                      // define constante igual a 1.5
#define ledPin 13                       // define pino digital D13

int Leitura = 0;                        // variavel inteiro igual a zero
float VoltageLDR;                       // variavel numero fracionario
float ResLDR;                           // variavel numero fracionario

void setup()
{
  pinMode(ledPin, OUTPUT);                  // configura D13 como saída digital
  Serial.begin(9600);                       // monitor serial - velocidade 9600 Bps
  delay(100);                               // atraso de 100 milisegundos
}

void loop()
{
  Leitura = analogRead(AnalogLDR);          // leitura da tensão no pino analogico A0
  VoltageLDR =  Leitura * (5.0/1024);       // calculo da tensão no LDR
  Serial.print("Leitura sensor LDR = ");    // imprime no monitor serial
  Serial.println(VoltageLDR);               // imprime a tensão do LDR

  if (VoltageLDR > Limiar)                  // se a tensão LDR maior do que limiar
    digitalWrite(ledPin, HIGH);             // liga LED com 5V
  else                                      // senão a tensão LDR < limiar
    digitalWrite(ledPin, LOW);              // desliga LED com 0V
  delay(500);                               // atraso de 500 milisegundos
}
```

No código acima usamos funcionalidades de todos os exemplos anteriores. Como o sensor é um elemento de um divisor de tensão, fazemos a sua leitura do mesmo modo que nos exemplos do potenciômetro e divisor de tensão.

Nesse caso, definimos uma tensão de limiar, a partir da qual desligamos ou ligamos um led para indicar que a intensidade da luz ultrapassou determinado valor. Esse limiar pode ser ajustado por você para desligar o led em intensidades diferentes de luz ambiente.

É importante que o sensor esteja exposto à iluminação ambiente e não sofra interferência de fontes luminosas próximas, mas que não sejam parte do ambiente.

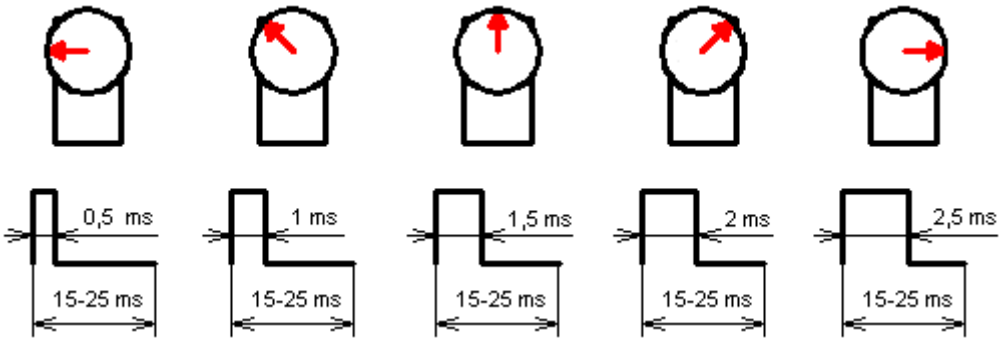
Referências:

- https://maker.pro/education/using-an-ldr-sensor-with-arduino-a-tutorial-for-beginners
- http://blog.eletrogate.com/controle-de-luminosidade-com-arduino-e-sensor-ldr/

Exemplo 5 - Acionando o Micro Servo 9g SG90 TowerPro

Um servomotor é um equipamento eletromecânico que possui um encoder e um controlador acoplado. Diferentemente de motores tradicionais, como de corrente contínua, o servo motor apresenta movimento rotativo proporcional a um comando de forma a atualizar sua posição. Ao invés de girar continuamente como os motores de corrente contínua, o servo ao receber um comando, gira até a posição especificada pelo mesmo.

Ou seja, o servo motor é um atuador rotativo para controle de posição, que atua com precisão e velocidade controlada em malha fechada. De acordo com a largura do pulso aplicado no pino de controle PWM, a posição do rotor é definida (0 a 180 graus). Os pulsos devem variar entre 0,5 ms e 2,5 ms.



Posição do Servo de acordo com a largura do pulso
Fonte: electronics.stackexchange.com

Existem dois tipos de Servomotores:

- Servomotor analógico
- Servomotor digital

Servomotores analógicos são os mais comuns e mais baratos. O controle da posição do rotor utiliza um método analógico através da leitura de tensão sobre um potenciômetro interno.

No caso dos servomotores digitais, mais caros e mais precisos, o controle da posição do rotor utiliza um encoder digital.

A figura abaixo mostra um típico **servo motor analógico**. Trata-se do **Micro Servo Tower Pro SG90 9G** que acompanha o kit Arduino Robótica.

Os Servos são acionados por meio de três fios, dois para alimentação e um correspondente ao sinal de controle para determinar a posição. Em geral, os três fios são:

- Marrom: GND,
- Vermelho: Alimentação positiva,
- Laranja: Sinal de controle PWM.

Lista de materiais:

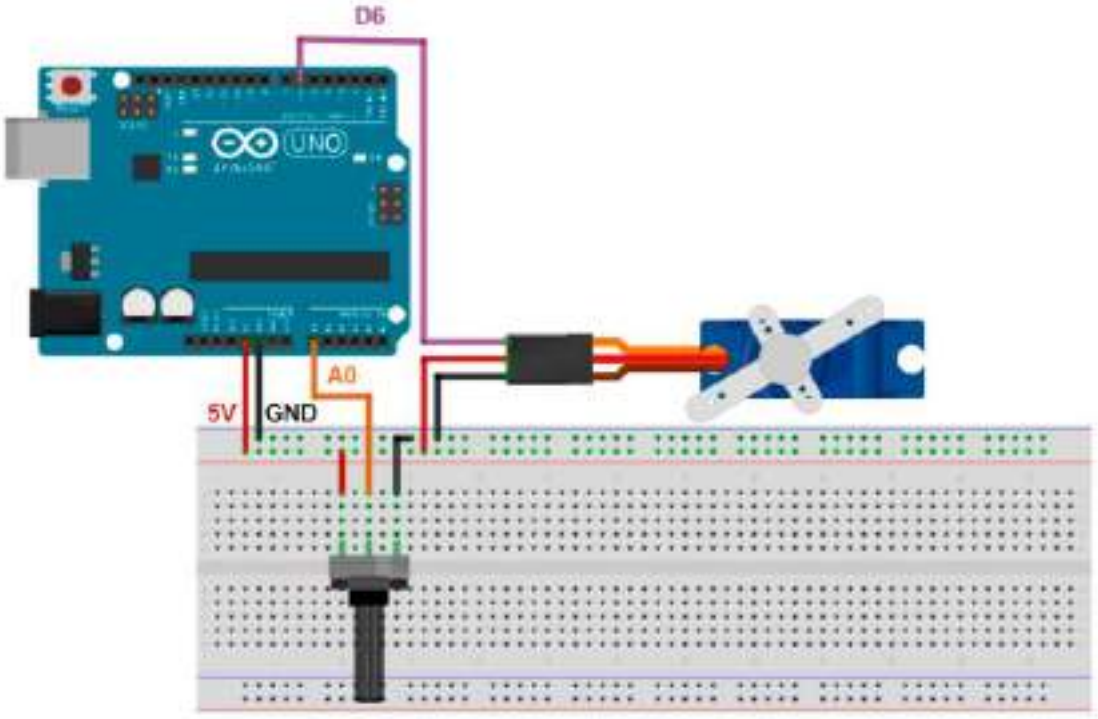
Antes de mais nada, vamos separar os componentes que precisamos para montar o servomotor junto com o Arduino. A nossa lista de componentes é a seguinte:

- Micro Servo 9g SG90 TowerPro;
- Arduino UNO + cabo USB;
- Potenciômetro de 10k;
- Jumpers para conexão no protoboard;
- Push button;

A montagem é simples. O servomotor em si deve ser ligado à alimentação conforme as cores apresentadas na introdução. De acordo com as especificações de tensão e corrente dos pinos do Arduino UNO, os pinos de VCC e GND da placa conseguem fornecer até 200 mA. Ao utilizar vários servomotores, é recomendado que você utilize uma fonte externa.

Diagrama do circuito:

A montagem para uma aplicação de controle do servo em qualquer posição, fica da seguinte forma:



Carregue o código abaixo e gire o potenciômetro para o Servo motor girar:

```
// Exemplo 5 - Acionando o Micro Servo TowerPro
// Apostila Eletrogate - KIT ROBÓTICA

#include <Servo.h>                    // usando bilbioteca Servo
#define potpin A0                     // define pino analógico A0
Servo myservo;                        // cria o objeto myservo
int val;                              // variavel inteiro

void setup()
{
  myservo.attach(6);                  // configura pino D6 - controle do Servo
}

void loop()
{
  val = analogRead(potpin);           // leitura da tensão no pino A0
  val = map(val, 0, 1023, 0, 179);    // converte a leitura em números (0-179)
  myservo.write(val);                 // controle PWM do servo
  delay(15);                          // atraso de 15 milisegundos
}
```

O aspecto mais importante desse software é a utilização da biblioteca **servo.h**. Esta biblioteca possui as funções necessárias para posicionar a servo para a posição desejada. Na função **Void Setup()** nós associamos o objeto servomotor, do tipo Servo, a um pino do arduino. Na mesma função, nós inicializamos o servo na posição 0º, utilizando o método **write** do objeto que acabamos de criar.

Na função **Void Loop(),** o procedimento consiste em ler o valor do potenciômetro e usá-lo como referência para atualizar a posição do servo. A leitura analógica do potenciômetro retorna um valor entre 0 e 1023. Para controlar o servo nós usamos valores de 0 a 179, correspondendo ao meio giro de 180º do mesmo. Assim, é necessário usar a função **Map()** para traduzir a escala de 0-1023 para a escala de 0-179. Dessa forma, os valores lidos do potenciômetro podem ser usados como referência para determinar a posição do servomotor.

Para atualizar a posição do servo a cada iteração do loop, nós chamamos o método **write()** do objeto servomotor, passando como parâmetro o valor lido do potenciômetro traduzido para a escala de 0-179. Assim, sempre que mexermos no potenciômetro, o servo motor irá atuar e atualizar a sua posição.

Referências:

- http://blog.eletrogate.com/servo-motor-para-aplicacoes-com-arduino/
- http://blog.eletrogate.com/kit-braco-robotico-mdf-com-arduino/
- https://www.arduino.cc/en/Tutorial/Knob
- https://www.allaboutcircuits.com/projects/servo-motor-control-with-an-arduino/
- http://www.instructables.com/id/Arduino-Servo-Motors/

Exemplo 6 - Sensor de temperatura NTC

O sensor de temperatura NTC pertence a uma classe de sensores chamada de Termistores. São componentes cuja resistência é dependente da temperatura. Para cada valor de temperatura absoluta há um valor de resistência.

Assim, o princípio para medir o sinal de um termistor é o mesmo que usamos para medir o sinal do LDR. Vamos medir na verdade, a queda de tensão provocada na resistência do sensor.

Existem dois tipos básicos de termistores:

- **PTC (Positive temperature coeficient):** Nesse sensor, a resistência elétrica aumenta à medida que a temperatura aumenta;

- **NTC (Negative Temperature Coeficient):** Nesse sensor, a resistência elétrica diminui à medida que a temperatura aumenta.

O termistor que acompanha o kit é do tipo NTC, como o próprio nome diz. Esse é o tipo mais comum do mercado.

O grande problema dos termistores é que é necessária fazer uma função de calibração, pois a relação entre temperatura e resistência elétrica não é linear. Existe uma equação, chamada equação de **Steinhart-Hart** que é usada para descrever essa relação (vide referências).

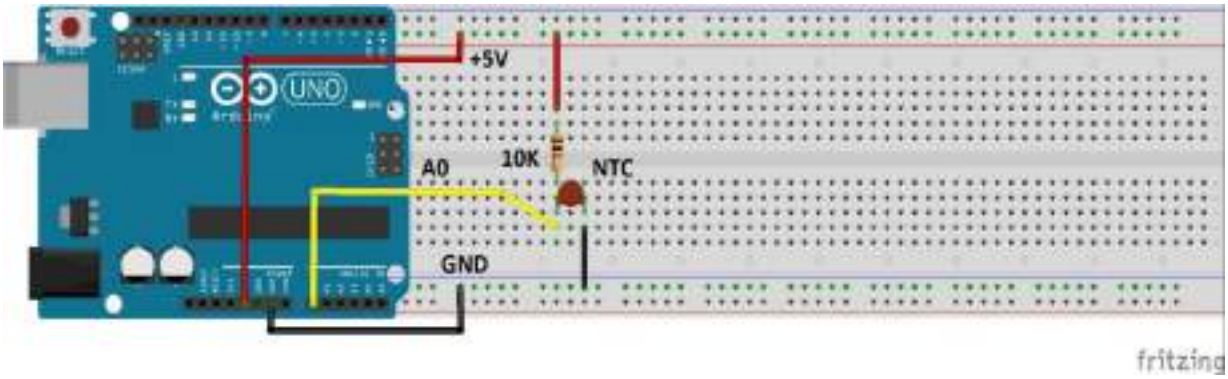
O código que vamos usar nesse exemplo utiliza a equação de Steinhart-Hart conforme a referência 1.

Lista de materiais:

Antes de mais nada, vamos separar os componentes que precisamos para montar o NTC junto com o Arduino. A nossa lista de componentes é a seguinte:

- 1 Sensor de temperatura NTC;
- Arduino UNO + cabo USB;
- 1 resistor de 10k;
- Protoboard;
- Jumpers para conexão no protoboard;

Diagrama do circuito:



Para uma melhor precisão nas medidas de temperatura, meça a tensão de 5V no barramento do protoboard, usando um multímetro (não contém no KIT). Essa tensão é a mesma usada como referência do conversor analógico-digital do Arduino. Especifique esse valor de tensão na primeira linha do Sketch:

```
#define Vin 5.0
```

Por exemplo, se a tensão for 4,85 V:

```
#define Vin 4.85
```

Carregue o código abaixo e meça a temperatura com o NTC:

```
// Exemplo 6 - Sensor de temperatura NTC
// Apostila Eletrogate - KIT ROBÓTICA

#define Vin 5.0                 // define constante igual a 5.0
#define T0 298.15               // define constante igual a 298.15 Kelvin
#define Rt 10000                // Resistor do divisor de tensao
#define R0 10000                // Valor da resistencia inicial do NTC
#define T1 273.15               // [K] in datasheet 0º C
#define T2 373.15               // [K] in datasheet 100° C
#define RT1 35563               // [ohms] resistencia in T1
#define RT2 549                 // [ohms] resistencia in T2
float beta = 0.0;               // parametros iniciais [K]
float Rinf = 0.0;               // parametros iniciais [ohm]
float TempKelvin = 0.0;         // variable output
float TempCelsius = 0.0;        // variable output
float Vout = 0.0;               // Vout in A0
float Rout = 0.0;               // Rout in A0

void setup()
{
  Serial.begin(9600);                       // monitor serial - velocidade 9600 Bps
  beta = (log(RT1 / RT2)) / ((1 / T1) - (1 / T2));     // calculo de beta
  Rinf = R0 * exp(-beta / T0);              // calculo de Rinf
  delay(100);                               // atraso de 100 milisegundos
}

void loop()
{
  Vout = Vin * ((float)(analogRead(0)) / 1024.0); // calculo de V0 e leitura de A0
  Rout = (Rt * Vout / (Vin - Vout));              // calculo de Rout
  TempKelvin = (beta / log(Rout / Rinf));         // calculo da temp. em Kelvins
  TempCelsius = TempKelvin - 273.15;              // calculo da temp. em Celsius
  Serial.print("Temperatura em Celsius: ");       // imprime no monitor serial
  Serial.print(TempCelsius);                      // imprime temperatura Celsius
  Serial.print("   Temperatura em Kelvin: ");     // imprime no monitor serial
  Serial.println(TempKelvin);                     // imprime temperatura Kelvins
  delay(500);                                     // atraso de 500 milisegundos
}
```

Referências e sugestões de leitura:

- http://www.instructables.com/id/NTC-Temperature-Sensor-With-Arduino/
- https://www.ametherm.com/thermistor/ntc-thermistors-steinhart-and-hart-equation
- http://www.thinksrs.com/downloads/PDFs/ApplicationNotes/LDC%20Note%204%20NTC%20Calculator.pdf
- https://www.mundodaeletrica.com.br/sensor-de-temperatura-ntc-ptc/

Exemplo 7 - Sensor Óptico Reflexivo TCRT5000

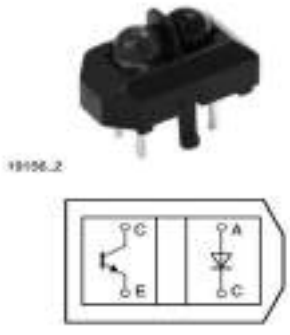
O sensor ótico reflexivo **TCRT5000** é um dos mais populares para utilização em projetos com Arduino. O sensor é fabricado pela Vishay, uma tradicional fabricante de componentes eletrônicos. Recomendamos fortemente, a leitura do datasheet do sensor:

https://www.vishay.com/docs/83760/tcrt5000.pdf

Trata-se de um sensor reflexivo que possui um emissor infravermelho e um fototransistor. O emissor é um led infravermelho que emite um sinal nessa faixa do espectro. Já o fototransistor é o receptor que faz a leitura do sinal refletido. Ou seja, o led emite um feixe infravermelho que pode ou não ser refletido por um objeto. Caso o feixe seja refletido, o fototransistor identifica o sinal refletido e gera um pulso em sua saída.

Tensão direta do LED emissor é de 1,25 V com uma corrente máxima de 60 mA. A corrente máxima do fototransistor é de 100 mA.

A distância máxima de detecção não é grande, ficando em torno de 25 mm (dois centímetros e meio), o que pode limitar um pouco a sua aplicação. O fototransistor vem com um filtro de luz ambiente, o que maximiza a identificação do feixe infravermelho refletido.



Sensor TCRT5000 visto por cima Fonte: Vishay

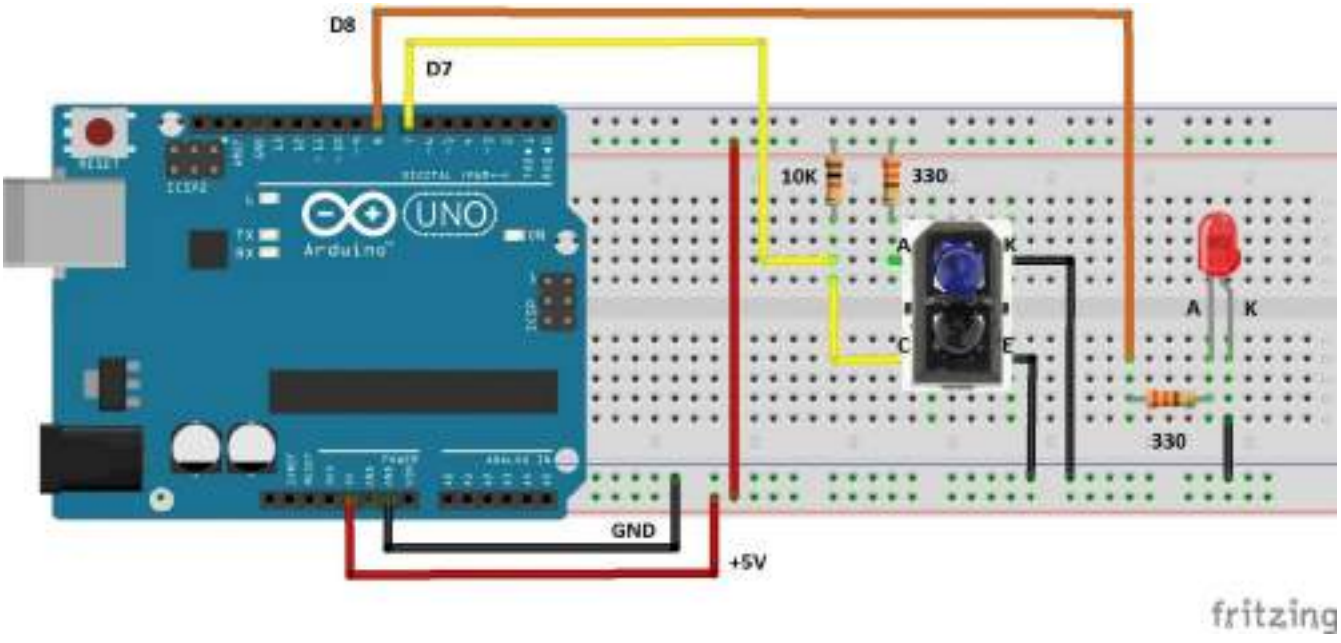
Lista de materiais:

- Arduino UNO R3;
- Protoboard;
- Sensor TCRT5000;
- Led vermelho 5mm;
- Resistor de 330;
- Resistor de 10K;
- Jumpers para ligação no protoboard;

Diagrama de Circuito:

O terminal Anodo (A) positivo do LED infravermelho está ligado por meio de um resistor de 330R ao VCC (5V) e o seu terminal catodo (K) negativo está ligado em GND. O Coletor (C) do fototransistor está ligado ao resistor de 10K (que esta conectado no VCC) e também na entrada digital D7 do Arduino. Essa entrada é que vamos usar para identificar se o sensor percebeu algum objeto ou não. O Emissor (E) do fototransistor está ligado ao GND.

O LED vermelho para indicação do Sensor, tem o terminal Anodo (A) positivo conectado à um resistor de 330R que está conectado à porta digital D8 do Arduino. O terminal catodo (K) negativo está ligado ao GND.



Código:

```
// Exemplo 7 - Sensor Ótico Reflexivo TCRT5000
// Apostila Eletrogate - KIT ROBOTICA

int leituraSensor;                    // variavel de leitura do sensor
int led = 8;                          // led indicador = D8 do Arduino
int fotoTransistor = 7;               // coletor do fototransistor = D7 do Arduino

void setup()
{
  pinMode(led, OUTPUT);               // pino do led indicador = saida
  pinMode(fotoTransistor, INPUT);     // pino do coletor do fototransistor = entrada
}

void loop()
{
  leituraSensor = digitalRead(fotoTransistor);  // leitura do sensor TCRT5000
  if (leituraSensor == 0 )                      // se o led refletir a luz
    digitalWrite(led, HIGH);                    // acende LED indicador
  else                                          // senão
    digitalWrite(led, LOW);                     // apaga LED indicador
  delay(500);                                   // atraso de 0,5 segundos
}
```

O código para utilização do sensor TCRT5000 é bem simples. Com o circuito montado, basta monitorar o estado do pino no qual a saída do sensor (coletor do fototransistor) está ligada. Se esse pino estiver em nível baixo, é porque algum objeto está presente dentro do raio de medição do sensor (a luz infra-vermelha do LED esta sendo refletida). Se o pino estiver em nível alto, então não há objeto nenhum no raio de medição do sensor.

Nesse exemplo, a presença do sensor é identificada pelo acionamento de um led, ou seja, sempre que um objeto for identificado pelo sensor, o led ficará aceso.

Referências:

- http://blog.eletrogate.com/sensor-optico-tcrt5000-com-arduino/
- http://www.instructables.com/id/Using-IR-Sensor-TCRT-5000-With-Arduino-and-Program

Exemplo 8 - Sensor de Distância Ultrassônico HC-SR04

O princípio de funcionamento do Sensor HC-SR04 consiste na emissão de sinais ultrassônicos e na leitura do sinal de retorno (reflexo/eco) desse mesmo sinal. A distância entre o sensor e o objeto que refletiu o sinal é calculada com base no tempo entre o envio e leitura de retorno.

"Sinais Ultrassônicos são ondas mecânicas com frequência acima de 40 KHz"

Como ouvido humano só consegue identificar ondas mecânicas até a frequência de 20KHz, os sinais emitidos pelo sensor Ultrassônico não podem ser escutados por nós.

O sensor HC-SR04 é composto de três partes principais:

- Transmissor Ultrassônico – Emite as ondas ultrassônicas que serão refletidas pelos obstáculos;
- Um receptor – Identifica o eco do sinal emitido pelo transmissor;
- Circuito de controle – Controla o conjunto transmissor/receptor, calcula o tempo entre a emissão e recepção do sinal;

O sensor é ilustrado na imagem abaixo. Repare como os invólucros do transmissor e receptor são bem proeminentes.

Sensor HC-SR04

O funcionamento consiste em três etapas:

1. O circuito externo (Arduino) envia um pulso de trigger de pelo menos 10us em nível alto;
2. Ao receber o sinal de trigger, o sensor envía 8 pulsos de 40khz e detecta se há algum sinal de retorno ou não;
3. Se algum sinal de retorno for identificado pelo receptor, o sensor gera um sinal de nível alto no pino de saída cujo tempo de duração é igual ao tempo calculado entre o envio e o retorno do sinal ultrassônico;

Por meio desse pulso de saída cujo tempo é igual a duração entre emissão e recepção, nós calculamos a distância entre o sensor e o objeto/obstáculo, por meio da seguinte equação:

$$Distancia = \frac{(Tempodeduraçãodosinaldesaídaxvelocidadedosom)}{2}$$

Em que o Tempo de duração do sinal de saída é o tempo no qual o sinal de output permanece em nível alto e a velocidade do som = 340 m/s;

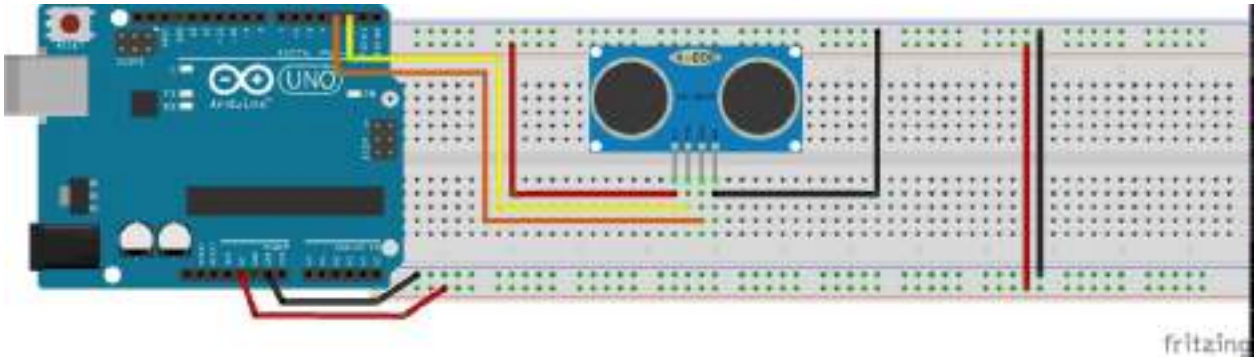
Repare que as unidades devem estar coerentes para o resultado da conta ser correto. Assim, se a velocidade do som é dada em metros/segundo, o tempo de duração do sinal de saída deve estar em segundos para que possamos encontrar a distância em metros.

Lista de Materiais:

Para este exemplo você vai usar os seguintes componentes:

- Protoboard;
- Jumpers de ligação;
- Sensor ultrassônico HCSR-04

Diagrama de circuito:



A montagem é bem direta, basta interligar o sensor com o Arduino. Não há necessidade de conversão de níveis lógicos ou ligação de componentes externos.

Código :

As variáveis declaradas são para determinar os pinos de trigger (pino 2) e de leitura do sensor (pino 3). Temos três variáveis do tipo float utilizadas para medir o tempo do pulso no output do sensor e calcular a distância. Na função void setup(), inicializamos o pino 2 como saída e o 3 com entrada. Além disso, configuramos a comunicação serial para 9600 de baud rate.

Na função void loop(), onde o programa será executado continuamente, executa-se três passos básicos:

1. Enviar pulso de 10us para o pino de trigger do sensor. Isto é feito com a função DisparaPulsoUltrassonico();

2. Ler o pino de output do sensor e medir o tempo de duração do pulso utilizando a função PulseIn. Esta função retorna o tempo de duração do pulso em microsegundos para que o mesmo seja armazenado em uma variável;

3. Por fim, chamamos a função CalculaDistancia (tempo) passando como parâmetro o tempo lido com a função PulseIn. Esta função calcula a distância entre o sensor e o obstáculo. Nós chamamos esta função de dentro da função Serial.println(), de forma que o resultado já é impresso diretamente no terminal serial;

```
// Exemplo 8 - Sensor de Distância Ultrassônico HC-SR04
// Apostila Eletrogate - KIT ROBOTICA

int PinTrigger = 2;                          // pino usado para disparar os pulsos do
sensor
int PinEcho = 3;                             // pino usado para ler a saida do sensor
float TempoEcho = 0;                         // variável tempo do eco
const float velocidadeSom_mps = 340;         // em metros por segundo
const float velocidadeSom_mpus = 0.000340;   // em metros por microsegundo

void setup()
{
  pinMode(PinTrigger, OUTPUT);               // configura pino Trigger como saída
  digitalWrite(PinTrigger, LOW);             // pino trigger - nível baixo
  pinMode(PinEcho, INPUT);                   // configura pino ECHO como entrada
  Serial.begin(9600);                        // inicializa monitor serial 9600 Bps
  delay(100);                                // atraso de 100 milisegundos
}

void loop()
{
  DisparaPulsoUltrassonico();                    // dispara pulso ultrassonico
  TempoEcho = pulseIn(PinEcho, HIGH);            // mede duração do pulso HIGH de eco em
microsegundos
  Serial.print("Distancia em metros: ");         // mostra no monitor serial

  Serial.println(CalculaDistancia(TempoEcho));   // mostra o calculo de distancia em
metros
  Serial.print("Distancia em centimentros: ");   // mostra no monitor serial
  Serial.println(CalculaDistancia(TempoEcho) * 100); // mostra o calculo de distancia em cm
  delay(2000);                                   // atraso de 2 segundos
}
```

//CONTINUA NA PRÓXIMA PÁGINA

```
void DisparaPulsoUltrassonico()
{
  digitalWrite(PinTrigger, HIGH);                    // pulso alto de Trigger
  delayMicroseconds(10);                             // atraso de 10 milisegundos
  digitalWrite(PinTrigger, LOW);                     // pulso baixo de Trigger
}

float CalculaDistancia(float tempo_us)
{
  return ((tempo_us * velocidadeSom_mpus) / 2 );     // calcula distancia em metros
}
```

A função **DisparaPulsoUltrassonico()** apenas ativa o pino 2 em nível alto, espera 10 microsegundos e volta a setar o pino para nível baixo. Lembre que 10 us é o tempo mínimo que o pulso deve perdurar para disparar o sensor HC-SR04.

A função **CalculaDistancia()** recebe como parâmetro o tempo do pulso no pino Echo do sensor. Com esse tempo nós usamos a equação, para calcular a distância entre o sensor e o obstáculo.

Referências:

- https://create.arduino.cc/projecthub/dusnoki/arduino-ultrasonic-sensor-hc-sr04-full-build-and-code-73614d
- http://blog.eletrogate.com/sensor-ultrassonico-hc-sr04-com-arduino/

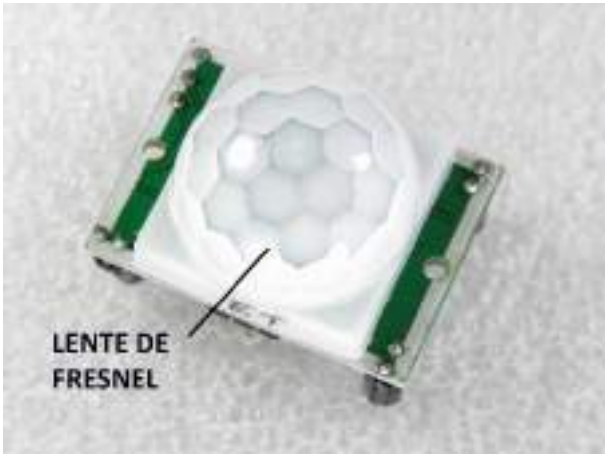
Exemplo 9 – Módulo Sensor de presença PIR + Buzzer

O Módulo Sensor de presença usa um componente eletrônico piroelétrico passivo, sensível às variações de luz de raios infravermelhos – PIR (passive infrared) . Sabemos que todos seres vivos emitem radiações de infravermelho, imperceptíveis ao olho humano. Dessa forma, esse sensor é capaz de detectar a presença de pessoas ou animais dentro de um ambiente.

https://en.wikipedia.org/wiki/Passive_infrared_sensor

Sob o sensor PIR existe uma lente de Fresnel que amplia a luz infravermelha recebida. Quando o ser vivo monitorado está em movimento, algumas variações na intensidade da luz são percebidas pelo circuito do módulo. Ao atingirem um determinado nível pré-configurado, o circuito dispara o sensor.

https://pt.wikipedia.org/wiki/Lente_de_Fresnel



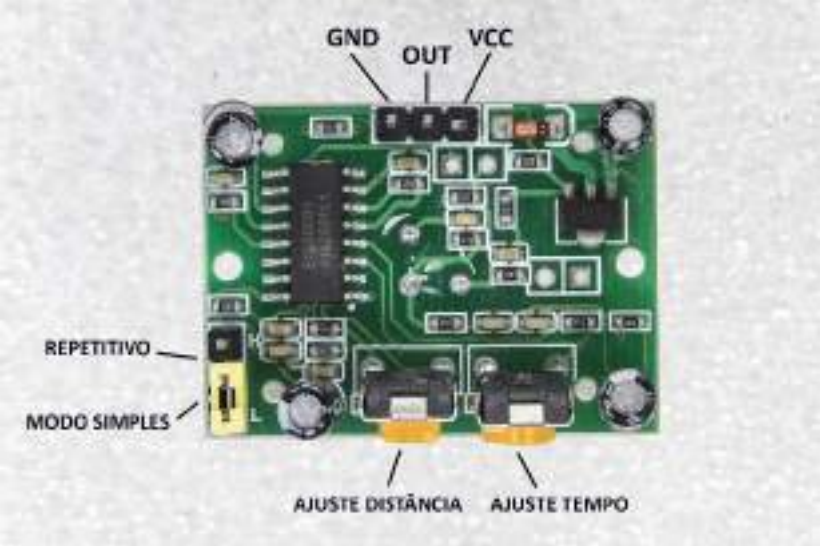
Especificações do Módulo PIR DYP-ME003:

- Sensibilidade e tempo ajustáveis
- Ângulo máximo de detecção = 110 °
- Tensão de Operação **VCC** : 4,5-20V
- Corrente estática: 50 uA
- Tensão de dados **OUT**: 3,3V (Alto) e 0V (Baixo)
- Distância detectável: 3-7m (Ajustável)
- Tempo de Atraso: 5 a 300seg (Default: 5seg)
- Tempo de Bloqueio: 2,5seg (Default)
- Dimensões: 3,2 x 2,4 x 1,8cm

https://www.elecfreaks.com/wiki/index.php?title=PIR_Motion_Sensor_Module:DYP-ME003

Aplicações sugeridas para o Módulo PIR :

- Sistemas de alarme de presença ou movimento,
- Disparo automáticos de câmera fotográfica – Camera Trap,
- Automação residencial – acionamento de lâmpadas,



Modulo PIR DYP-ME003

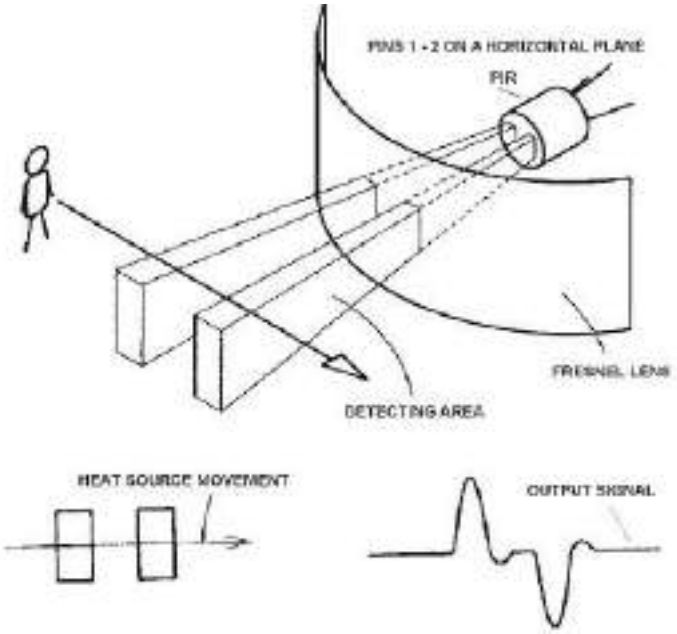
OBS : alguns fabricantes podem inverter os pinos VCC com GND. Veja a identificação na placa.

- Potenciômetro para ajuste do Tempo de atraso do disparo
  - Tempo de atraso mínimo de 5 segundos e máximo de 300 segundos
- Potenciômetro para ajuste de sensibilidade (distância)
  - A distância poderá ser ajustada entre 3 e 7 metros
- Jumper modo de disparo :
  - modo de disparo simples (posição L)
  - modo de disparo repetitivo (posição H = default)

Como funciona o Módulo PIR :

O sensor Piroelétrico possui duas áreas de “visão”, sensíveis às radiações infravermelhas. As lentes de Fresnel focalizam as ondas de radiações para sobre o sensor. Quando o

sensor está ocioso, ambas áreas sensíveis detectam a mesma quantidade de radiação. Quando um corpo quente como um ser humano ou animal passa na frente do sensor, variações nas quantidades de energia são percebidas pelo mesmo. Essas variações fazem o circuito disparar a saída OUT com um nível de tensão de 3,3V. A duração do pulso positivo nessa saída pode ser ajustada no potenciômetro TEMPO, entre 5 e 300 segundos. No potenciômetro de DISTÂNCIA ou sensibilidade, pode-se ajustar a distância entre o sensor e a pessoa, entre 3 e 7 metros. No modo simples, a partir de um disparo a saída se mantem alta durante o tempo ajustado. No modo repetitivo, os disparos se sobrepõem e assim a duração do pulso de saída será maior.

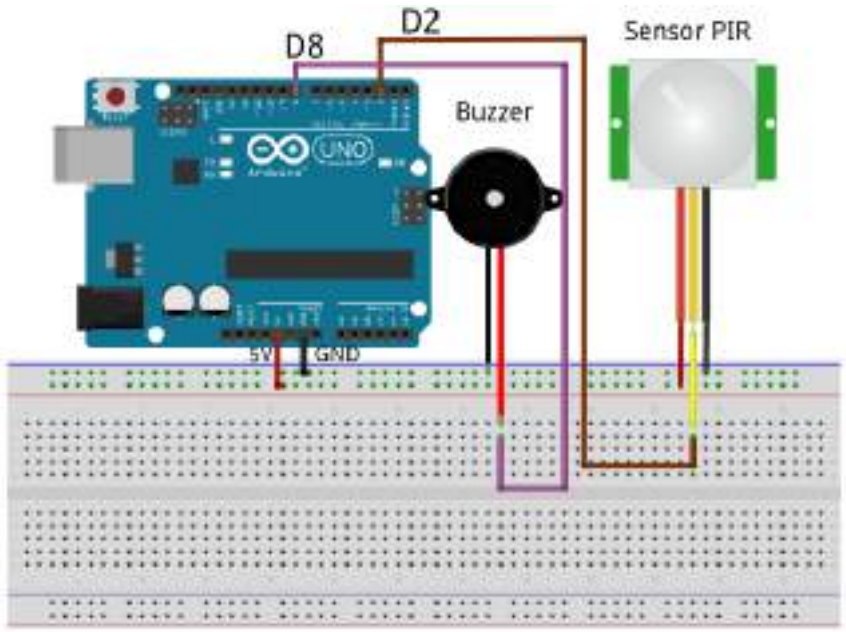


Lista de Materiais:

Para este exemplo você vai usar os seguintes componentes:

- Protoboard;
- Jumpers de ligação;
- Buzzer Ativo 5V
- Sensor PIR.
- Arduino UNO

Montagem do Módulo PIR com Arduino:



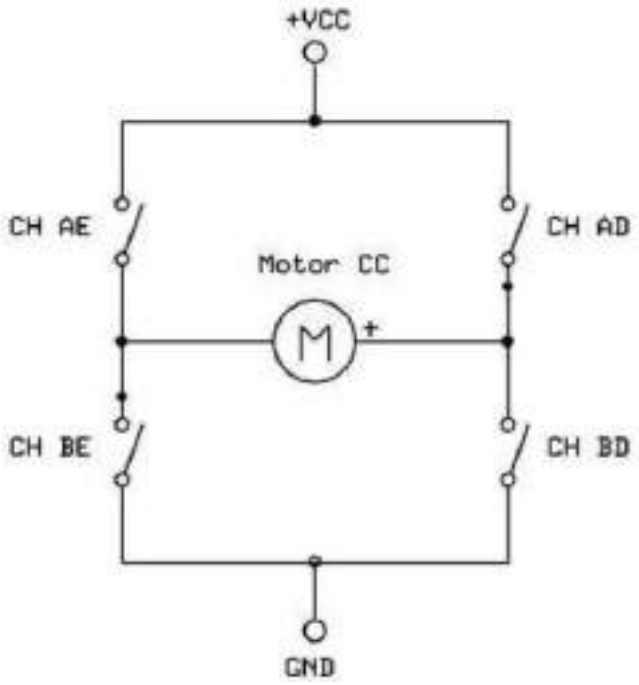

```
// Exemplo 9 – Modulo Sensor de Presença PIR
// Apostila Eletrogate - KIT ROBOTICA


int sensorPIR = 2;           // Pino ligado ao sensor PIR
int buzzer = 8;              // Pino ligado a sirene Buzzer
int disparo;                 // Variavel de disparo do sensor
int frequencia = 4000;       // Frequencia do apito em Hertz
int duracao = 500;           // Duração do apito em milisegundos


void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);        // Define LED embutido D13 como saída
  pinMode(sensorPIR, INPUT);           // Define pino do sensor PIR D2 como entrada
  pinMode(buzzer, OUTPUT);             // define pino da sirene buzzer D8 como saída
  delay(5000);                         // tempo necessário para o módulo PIR se inicializar 5 segundos
}

void loop()
{
  disparo = digitalRead(sensorPIR);          // Le o valor da saída do sensor PIR
  if (disparo == LOW)                        // Se saída tem nível baixo
  {
    digitalWrite(LED_BUILTIN, LOW);          // LED embutido permanece apagado
  }
  else                                       // Se a saída tem nível alto
  {
    digitalWrite(LED_BUILTIN, HIGH);         // LED embutido acende
    tone(buzzer, frequencia, duracao);       // Buzzer apita
  }
}
```

Para o funcionamento do circuito, ajuste os potenciômetros de Tempo e Distancia para a posição mínima (gire tudo para o sentido anti-horário) . Dessa forma o tempo de duração será de aproximadamente 5 segundos e a distância de percepção será de 3 metros. Insira o jumper de modo para a posição H, para que o sensor fique mais perceptível. Despois dos testes, faça as suas modificações.

Quando o sensor perceber a presença de alguém, o LED embutido no Arduino (D13) acenderá e a sirene Buzzer apitará na frequência de 4000 Hz.
Com essa montagem, usando o relé do exemplo 7 poderá implementar um sistema de alarme ou automação residencial.

Exemplo 10 – Ponte H Dupla L298N

O uso de Ponte H é interessante para diversas implementações com motores, porque estes componentes permitem uma série de facilidades. Então, o primeiro ponto a citar é que a ponte H nos permite girar o motor para ambos os lados, com um circuito que funciona da seguinte forma:



Se queremos que o motor funcione no sentido 1, fechamos somente as chaves CH AD e CH BE. Se caso queremos que gire no sentido 2, abrimos as anteriores e fechamos CH AE e CH BD.

Então, por meio de comutação de chaves, a ponte H consegue fazer com que um motor gire nos dois sentidos. Como no módulo Ponte H L298N essa comutação não é feita de

forma física, existe um chip (L298N) embarcado na placa que faz essa comutação de chaves de forma eletrônica, que é mais eficiente do que uma comutação mecânica.

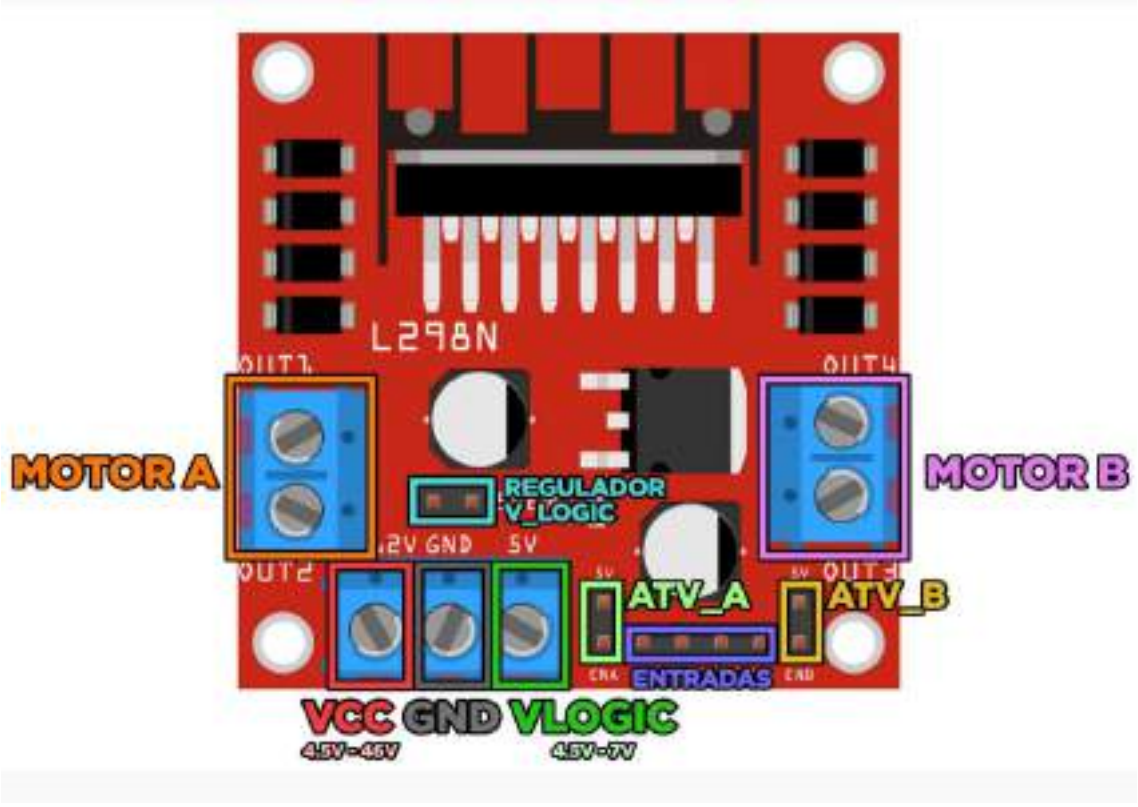
As especificações do chip L298, que faz todo o controle da ponte H, pode ser conferido clicando neste link: https://www.st.com/resource/en/datasheet/l298.pdf.

As principais especificações são:

- Tensão de Operação: 4.5 V~46 V
- Controle de 2 motores DC ou 1 motor de passo
- Corrente de Operação máxima: 2 A por canal ou 4 A total.
- Tensão lógica: 4.5 V ~7 V
- Corrente lógica: 0~36 mA
- Limites de Temperatura: -20 a +135 °C
- Potência Máxima: 25 W
- Dimensões: 43 x 43 x 27 mm.

Como funciona a Ponte H L298N :

Abaixo será explicado com maiores detalhes como funciona a placa e como usar:



Vcc:

Este borne é responsável pela alimentação dos motores, **com isso a tensão colocada neste borne de Vcc será copiada para os motores**. Por isso, muita atenção: Se você possui um motor que opera somente até 12V, por exemplo, não alimente Vcc com mais de 12V, pois danificará o motor.

- **Observação:** Esta placa possui um regulador de tensão do modelo 78m05, e a tensão máxima dele é **35V, mas a margem segura para uso é abaixo de 20V, sendo o recomendado 12V**. Por isso, se for usar tensão de **alimentação Vcc acima de 12V, não use a porta VLogic para alimentar outros circuitos**, e para tensões acima de 20V, desative o regulador de tensão (como será demonstrado mais abaixo) e faça a alimentação Vlogic através do borne, observando sua limitação de tensão.

GND:

É o GND da placa, que deverá ser o mesmo GND do Arduino ou de outro microcontrolador. Por isso, é importante interligar os GNDs sempre com fios. Serão demonstrados na seção de exemplos.

Vlogic:

Este borne é responsável pela alimentação lógica da ponte H. O chip L298N presente na placa recebe comandos do Arduino, e para ele conseguir operar os comandos e atuar na saída ele precisa de operar como um componente lógico (como o Arduino), e para isso precisa atuar em uma faixa bem restrita de tensão, que no caso é 4.5V a 7V. Como foi dito a placa possui um regulador de tensão de 5V, que a função dele é justamente fazer a regulação de Vcc para a alimentação lógica do chip em 5V. Por isso, se o regulador estiver ativo, este pino Vlogic terá a tensão de 5V, pois, é a tensão presente na saída do regulador.

É possível desativar o regulador de tensão e fazer a alimentação lógica do chip por conta própria, através do borne Vlogic. Em algumas situações é necessário desativar o regulador de tensão e fazer a alimentação lógica através de Vlogic, essas situações serão abordadas e explicadas na seção de exemplos de uso da placa.

- **Importante:** Como dito, quando o regulador de tensão está ativo, a tensão VLogic está sendo regulada em 5V automaticamente, logo, este borne terá 5V. Essa tensão é para uso interno do chip e pode ser aproveitada para alimentar algum circuito ou microcontrolador que use até 5V, com a limitação de corrente de aproximadamente 200mA. **Porém, se o regulador de tensão estiver acionado, JAMAIS alimente este pino, pois causará danos a placa.**

Regulador Vlogic:

Este jumper é responsável por ativar ou desativar o regulador de tensão da alimentação de Vlogic. Quando o jumper está encaixado entre os terminais, o regulador de tensão está ativado e a Vlogic como a ser regulado em 5V a partir da tensão de Vcc. Por isso, é muito importante saber manusear esse jumper do regulador. Caso ficou dúvidas sobre o funcionamento do regulador, releia a parte Vlogic.

Atv_A:

Este jumper é responsável por desativar ou ativar o motor A para ser controlado pelo Arduino ou outra placa.
Este jumper funciona da seguinte forma: um dos terminais possui 5V (ou a tensão de alimentação VLogic), e o outro terminal é o que vai para o chip L298N. A placa possui um resistor de pull-down no outro resistor, o que significa que quando o jumper é solto, o terminal ficará com nível lógico baixo, e o chip entenderá que é necessário desabilitar o motor A. Mas quando o jumper é colocado entre os terminais, o 5V é ligado ao terminal de sinal pelo jumper. Com isso, o sinal de nível alto é enviado ao chip L298n e ele entende que os motores devem estar habilitados para os acionamentos.

Atv_B:

Este jumper é responsável por desativar ou ativar o motor B para ser controlado.
O funcionamento é exatamente idêntico ao jumper Atv_A.

Entradas:

As entradas de controle da placa possuem os pinos IN1, IN2, IN3 e IN4, sendo que a entrada IN1 e IN2 são as entradas que controlam os bornes do motor A, e as entradas IN3 e IN4 controlam os bornes do Motor B. Para controlar os motores, acione os pinos da seguinte forma:

MOTOR A	IN1	IN2
Sentido direto	HIGH	LOW
Sentido reverso	LOW	HIGH
Freio	LOW	LOW
Freio	HIGH	HIGH

E no motor B, acione da mesma forma:

MOTOR B	IN3	IN4
Sentido direto	HIGH	LOW
Sentido reverso	LOW	HIGH
Freio	LOW	LOW
Freio	HIGH	HIGH

Motor A e Motor B:
Estes bornes são de fato onde os motores DC são conectados. A posição dos fios do motor na conexão com o borne altera o sentido de rotação do motor. Se for invertido os fios no borne, o sentido de rotação também é invertido no acionamento. Essa é uma boa dica para alterar a rotação dos motores, caso estejam girando em um sentido não desejado.
Perceba também, que como juntando o borne de Motor A e Motor B possuímos 4 conexões, podemos usar essa ponte para controlar 1 motor de passo. Se o motor for de 4 fios, podemos utilizar conectando direto. Se for de 5, ou 6, podemos conectar o tap central ao VLogic.

Montagem da Ponte H L298N com Arduino:

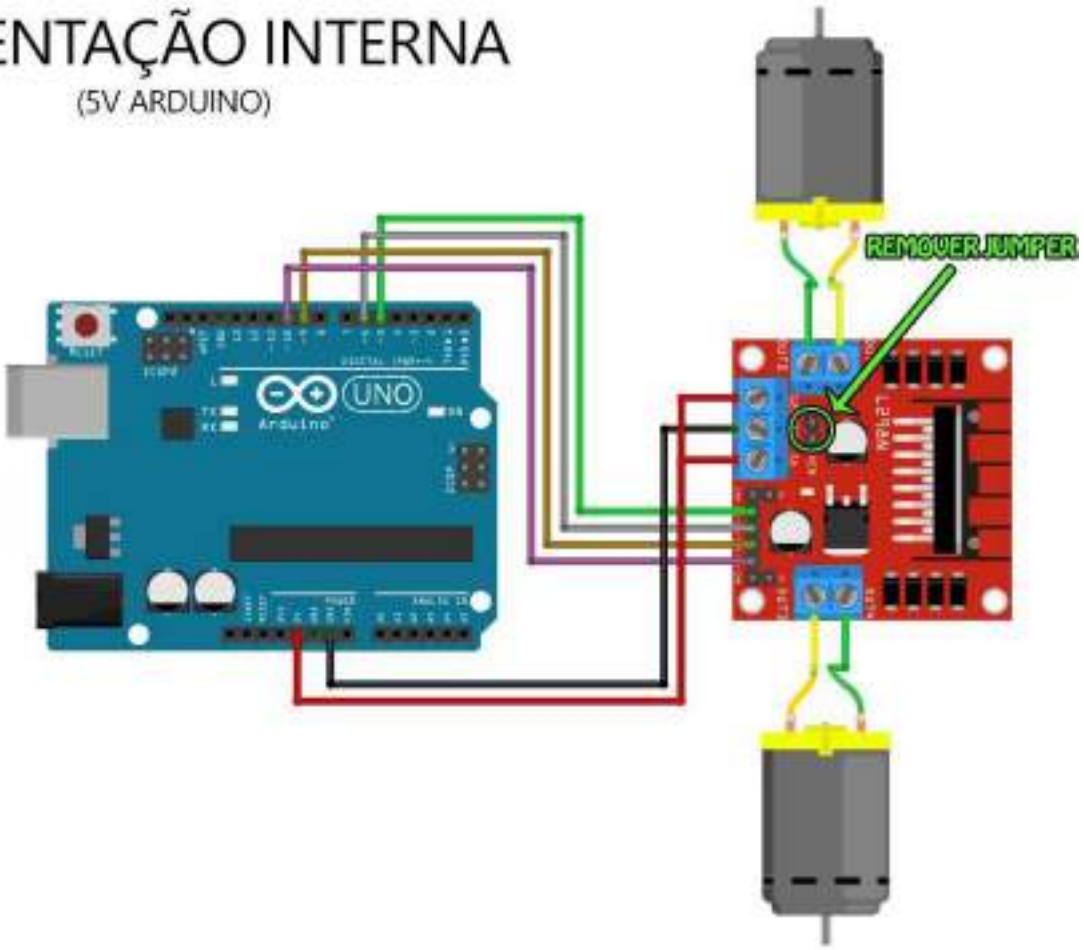
A montagem a ser feita dependerá muito da tensão disponível, por isso, aqui será abordado 4 tipos comuns de ligações que podem ser feitas:

5V diretamente pelo USB

Esta conexão é destinada para ser feita usando a própria alimentação do cabo USB, para fins de testes ou alguma aplicação onde é importante testar o sistema constantemente e precisa fazer mudanças no código. É importante que a conexão seja feita **com o motor sem carga e consumindo baixa corrente**. Para isso, deve **remover o jumper do regulador de tensão** e conectar os 5V do Arduino tanto em Vcc quanto em Vlogic. Fazendo a conexão abaixo:

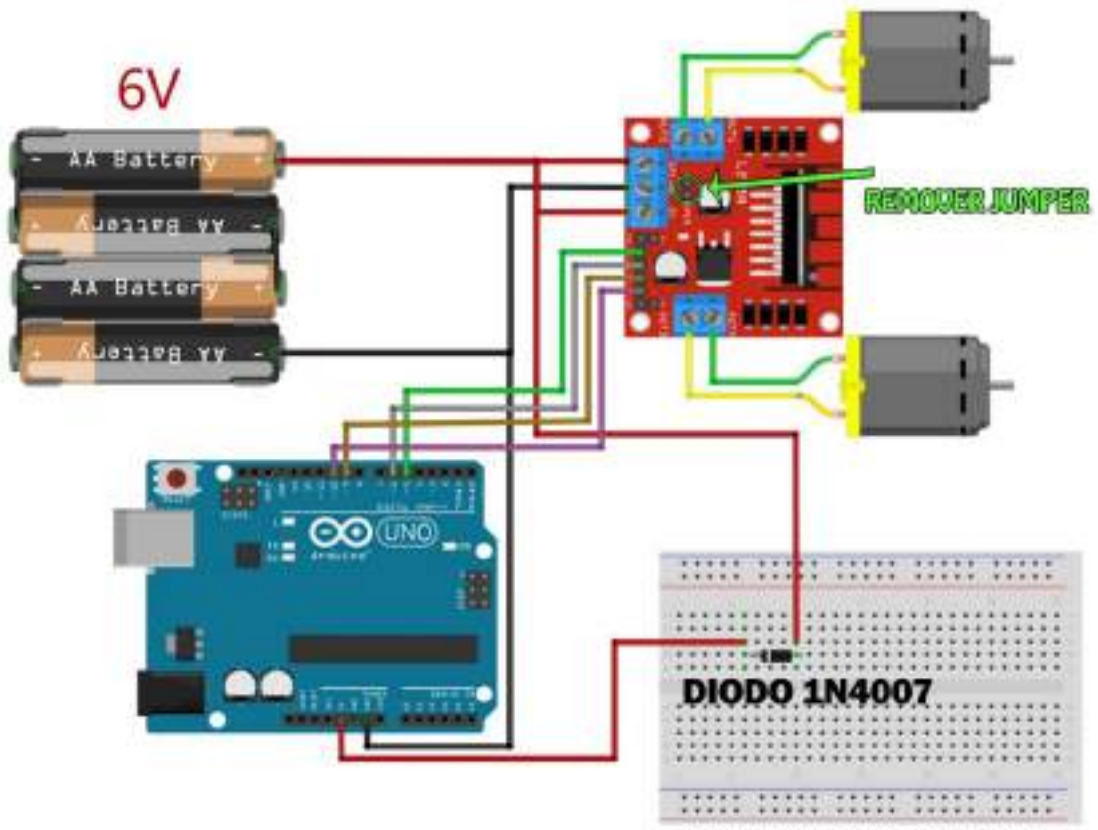
ALIMENTAÇÃO INTERNA

(5V ARDUINO)



6V de alimentação externa

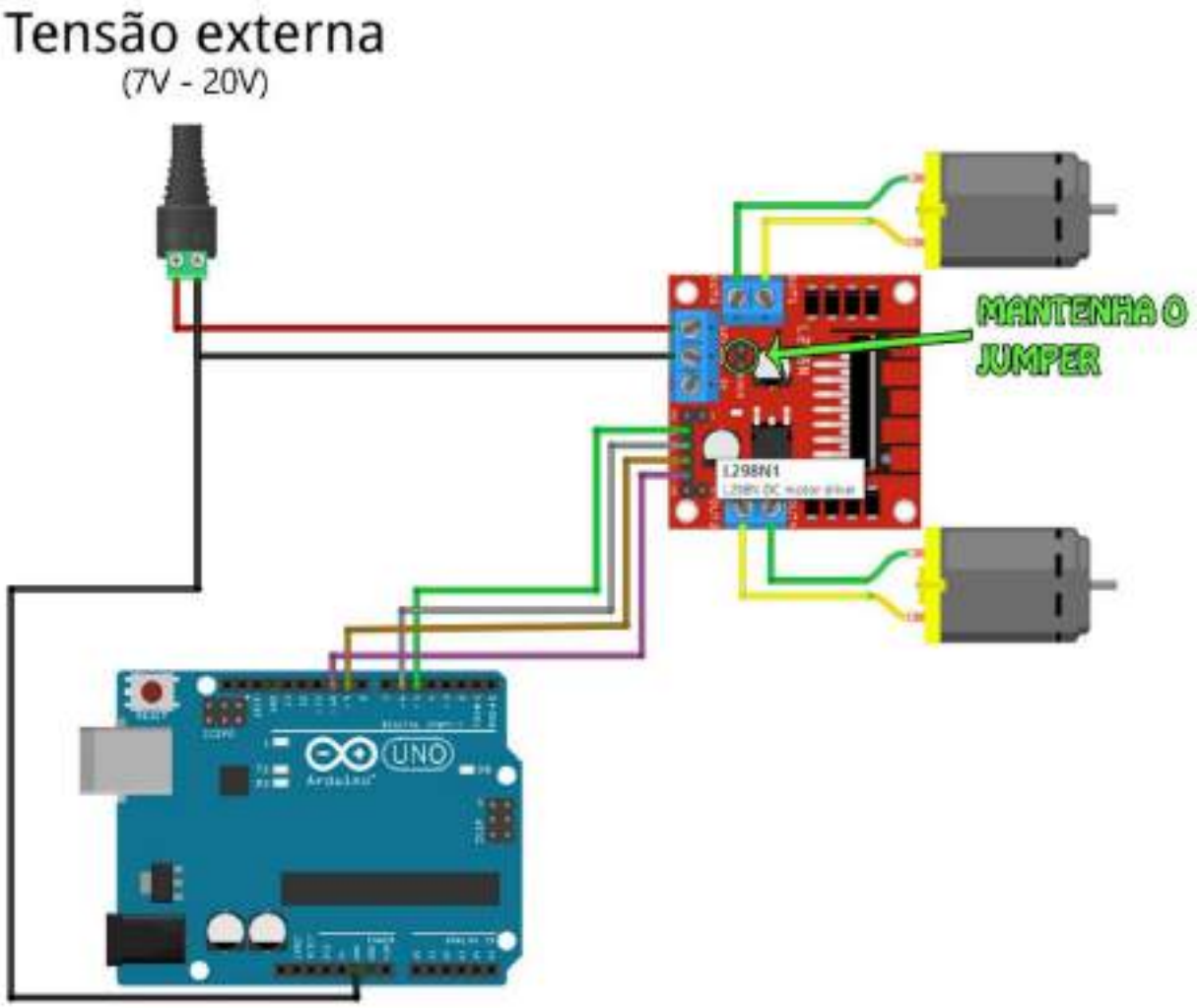
Com um suporte de 4 pilhas AA, e usando pilhas alcalinas na montagem, temos um total de 6V na alimentação da ponte H. A montagem é similar a anterior, mas com um detalhe: O fio do suporte de pilhas não pode ser ligado diretamente ao pino 5V do Arduino UNO, porque o microcontrolador Atmega 328P só suporta até 5,5V, então, para contornar isso, usaremos um diodo na montagem. Este diodo tem a função de provocar uma queda de tensão de aproximadamente 0,7V em cima dele, e com isso, a tensão no microcontrolador do Arduino seria de 5,3V, estando no intervalo de operação.
Para uma margem ainda maior de segurança, insira dois diodos em série.



É importante fazer a observação que se caso for inserido na montagem componentes adicionais. Pode ser que seja necessário fazer uma alimentação separada para o Arduino para conseguir suprir a demanda de corrente do sistema.

7V – 20V de alimentação externa

Para uma tensão externa entre 7V e 20V, é recomendado fazer uma alimentação separada entre a ponte H e o Arduino, sendo que até 12V, você pode usar o borne VLogic para alimentar alguns circuitos que necessitam de 5V e demandam baixa corrente. Nesta montagem é necessário **manter o jumper do regulador de tensão**, já que o circuito Vlogic será alimentado pela saída do regulador de tensão interno.

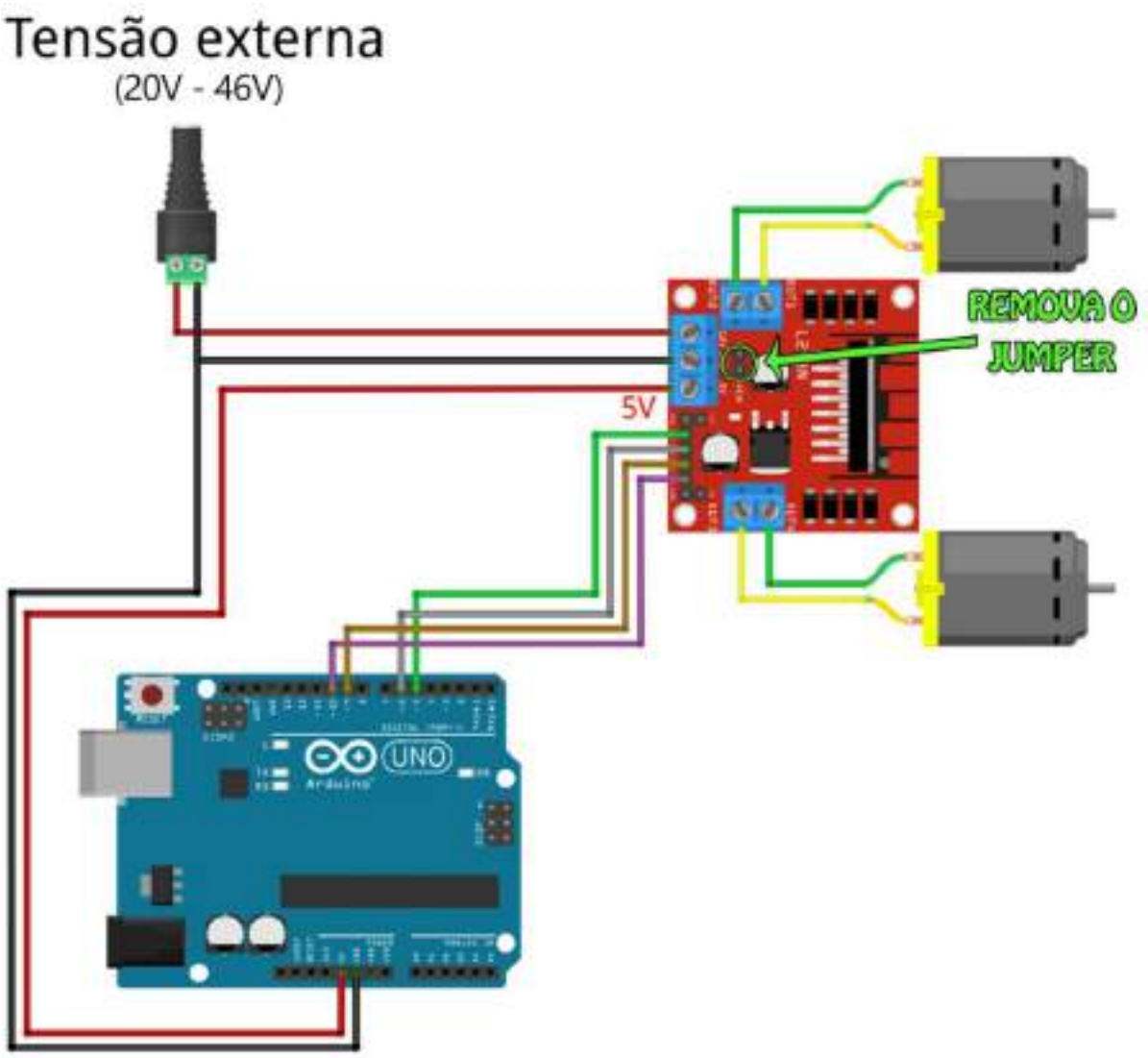


É possível, em até 12V, fazer uma alimentação única entre o Arduino e a ponte H, colocando a mesma tensão de 12V no pino de Vin do Arduino e no borne Vcc da ponte H.

20V – 46V de alimentação externa

Para uma alimentação de 20V até 46V -que é o limite máximo - é extremamente **necessário remover o jumper do regulador de tensão**. Este procedimento é necessário porque este regulador só opera em uma margem segura abaixo de 20V, e acima disso ele poderá sobreaquecer e danificar **(principalmente acima de 35V)**. Logo, é necessário remover o jumper e fazer a alimentação de Vlogic com o 5V do Arduino.

Lembrando que nesse caso a alimentação do Arduino e da ponte H é feita de maneira independente. O circuito é este:



Código para controle da Ponte H:

```
// Exemplo 10 – Ponte H Dupla L298N
// Apostila Eletrogate - KIT Robotica


// Iremos fazer uma classe para facilitar o uso da ponte H L298N na manipulação dos motores na função
Setup e Loop.

class DCMotor {
  int spd = 255, pin1, pin2;

  public:

    void Pinout(int in1, int in2){ // Pinout é o método para a declaração dos pinos que vão controlar
o objeto motor
      pin1 = in1;
      pin2 = in2;
      pinMode(pin1, OUTPUT);
      pinMode(pin2, OUTPUT);
      }
    void Speed(int in1){ // Speed é o método que irá ser responsável por regular a velocidade
      spd = in1;
      }
    void Forward(){ // Forward é o método para fazer o motor girar para frente
      analogWrite(pin1, spd);
      digitalWrite(pin2, LOW);
      }
    void Backward(){ // Backward é o método para fazer o motor girar para trás
      digitalWrite(pin1, LOW);
      analogWrite(pin2, spd);
      }
    void Stop(){ // Stop é o metodo para fazer o motor ficar parado.
      digitalWrite(pin1, LOW);
      digitalWrite(pin2, LOW);
      }
   };

   DCMotor Motor1, Motor2; // Criação de dois objetos motores, já que usaremos dois motores, e eles já
estão prontos para receber os comandos já configurados acima.

void setup() {
  Motor1.Pinout(5,6); // Seleção dos pinos que cada motor usará, como descrito na classe.
  Motor2.Pinout(9,10);
}

void loop() {
  Motor1.Speed(200); // A velocidade do motor pode variar de 0 a 255, onde 255 é a velocidade máxima.
  Motor2.Speed(200);

  Motor1.Forward(); // Comando para o motor ir para frente
  Motor2.Forward();
  delay(1000);
  Motor1.Backward(); // Comando para o motor ir para trás
  Motor2.Backward();
  delay(1000);
  Motor1.Stop(); // Comando para o motor parar
  Motor2.Stop();
  delay(500);
}
```

Exemplo 11 – Kit Chassi 2 Rodas

O kit chassi é um conjunto de peças e componentes que permitem a montagem de um robô móvel, com 2 rodas que giram em sentidos independentes, fazendo assim o robô ter uma possibilidade de movimentos e direções muito grandes.

O chassi presente neste kit tem uma série de furos onde pode ser encaixado sensores, atuadores, e que te permite incluir diversos sensores presente no kit Arduino Robótica para realizar as mais variadas tarefas.

Esta montagem do carrinho também está disponível no nosso em https://blog.eletrogate.com, onde existem diversas outras montagens e experimentos.

Lista de Materiais

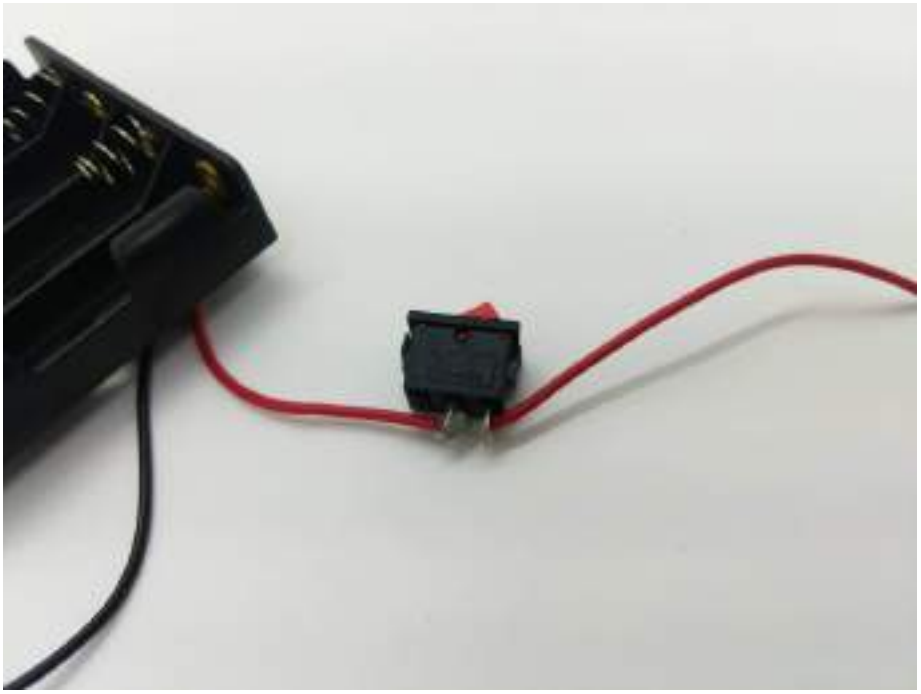
- Módulo Ponte H Dupla L298N
- Chassi 2WD em acrílico
- 2 Motores DC 3V – 6V com caixa de redução
- 2 rodas 68mm de borracha
- Roda boba (universal)
- Suporte para 4 pilhas AA
- Jogo de parafusos e acessórios
- Jumpers macho-fêmea
- Jumpers macho-macho
- Adaptador de bateria 9V
- Chave Gangorra On/Off
- 4 Pilhas Alcalinas AA
- Bateria Alcalina 9V
- Protoboard
- Arduino e cabo de conectar no computador

O primeiro passo é soldar os fios soltos aos terminais do motor (peça ajuda a um adulto, caso você seja criança). Para isso, vamos soltar o motor de sua caixa de redução, removendo a alça que une os dois

Após isso, retire-o e faça a ligação de fios aos terminais do motor com solda, e também é recomendado passar um pouco de cola quente em cima da solda para evitar algum curto ou contato indesejado.

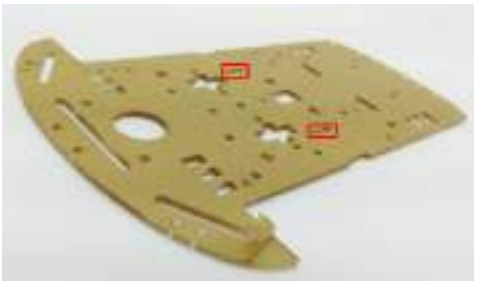
Faça a solda para os dois motores. E depois, volte com os motores para as suas respectivas caixas de redução (as caixas amarelas) e coloque a alça transparente, e assim, os motores já estão prontos para serem usados.

Com o ferro de solda ainda ligado, solde a chave gangorra no suporte de pilhas, para podermos desligar o carrinho e economizar energia. O procedimento será bem simples, basta colocar a chave on/off no final do fio vermelho do suporte e soldar em um dos dois terminais, e pegar um outro fio disponível (pode ser um jumper macho/macho que veio junto ao kit) e soldar no outro fio. Se quiser também, uma possibilidade é cortar o fio vermelho ao meio, descascar só um pouco cada ponta, e fazer como nos passos anteriores; a desvantagem fica sendo a limitação será o limite para posicionar a chave para desligar na montagem do carrinho.



Com tudo soldado, volte a montagem da estrutura do carrinho. Observe que a peça maior do carrinho parece de madeira. Se trata de uma película protetora, pois a case é de acrílico transparente. Se quiser deixar o seu carrinho mais estético, remova a película marrom de ambos os lados, só puxando, como na imagem abaixo.

Com as peças da imagem acima em mãos, pegue os motores e o conjunto de parafusos, e faça o seguinte:

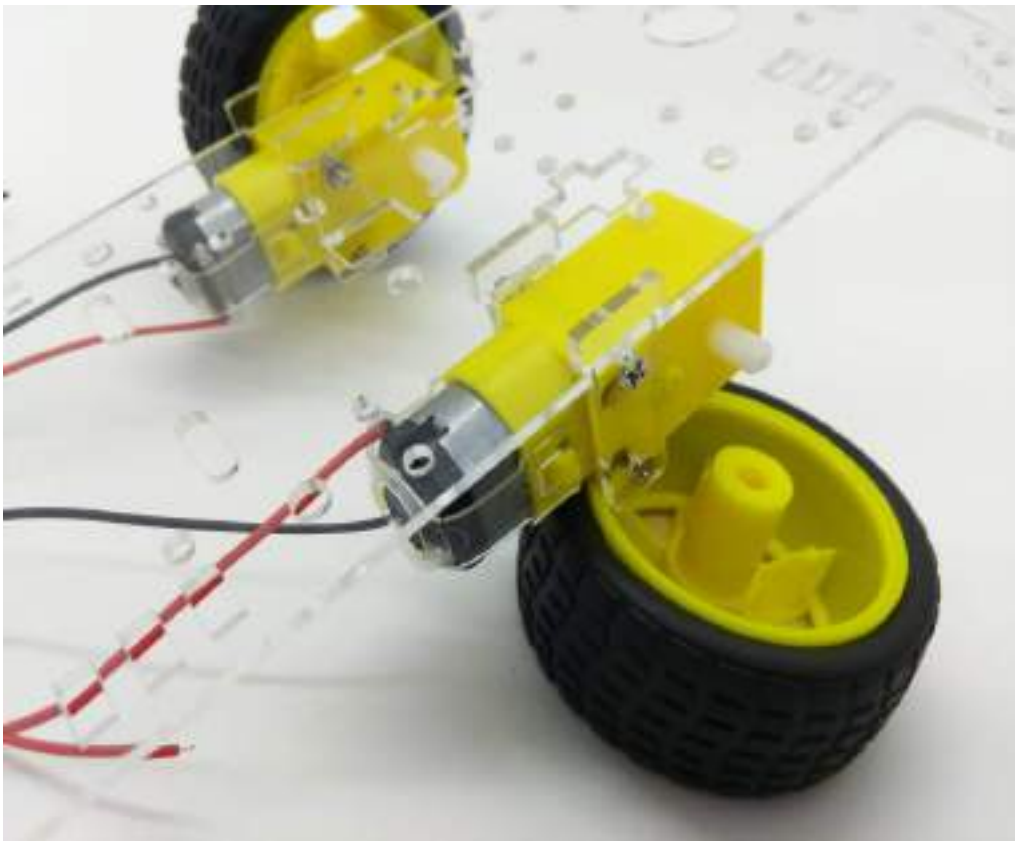


1° - Encaixe duas peças menores dessas em formato de T, nas fendas internas.

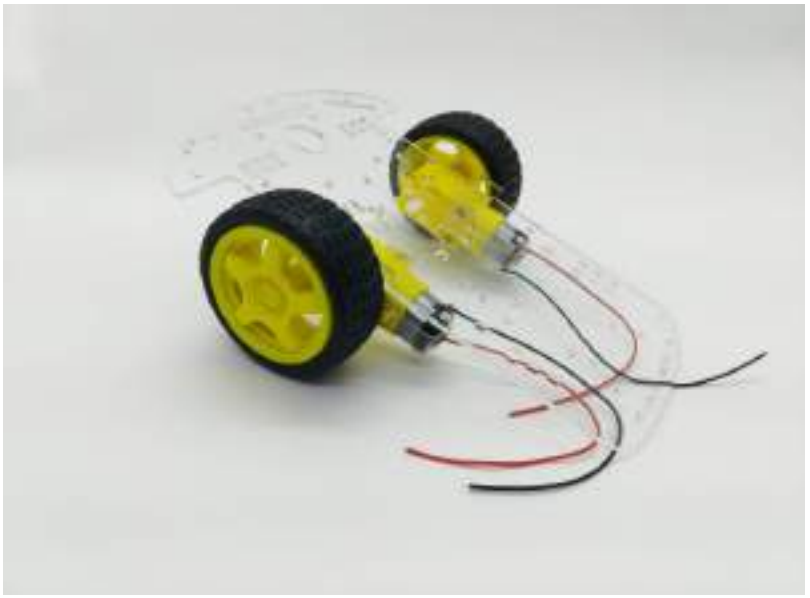
2° - Coloque o motor com seus furos laterais alinhados aos furos do suporte menor, que foi encaixado anteriormente

3° - Encaixe outros dois suportes pequenos em formato de T com os furos alinhados aos furos do motor também e parafuse usando porcas.

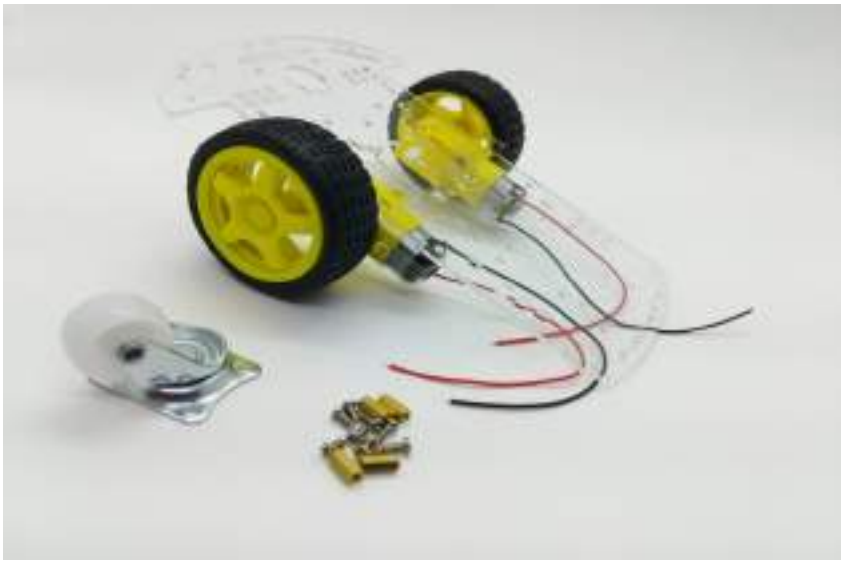
É interessante observar que devemos colocar o eixo do motor virado para a extremidade do chassi onde tem um círculo vazado no acrílico. Fazendo para os dois lados, e encaixando a roda no eixo do motor, o resultado obtido deverá ser este:



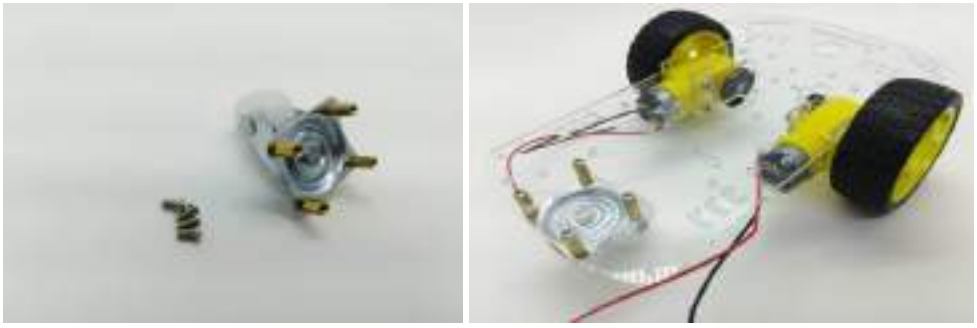
E encaixando a roda no eixo do motor, fica assim:



O próximo passo será a instalação da roda boba, e para isso precisaremos das seguintes peças da imagem abaixo:



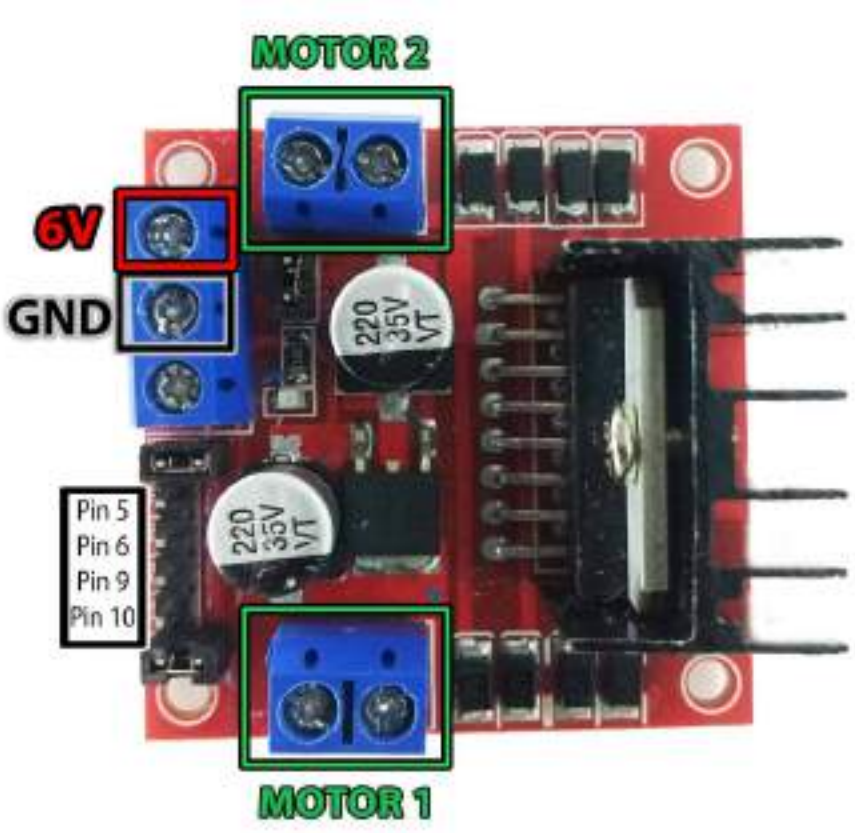
Começaremos parafusando o extensor, que é essa peça dourada no kit de parafusos à roda boba, e depois parafusando o extensor ao chassi do carrinho da seguinte forma:



E assim, está concluído a montagem do esqueleto básico do carrinho, e agora iremos montar a parte para controlar ele.

Para fazer o acionamento dos motores, precisaremos de algo que consiga fornecer energia o suficiente para isso, e que possa ser controlado. Para isso iremos usar um driver ponte H L298N. A vantagem de usar este driver é permitir a rotação nos dois sentidos, como visto no exemplo anterior, e também permitir que uma corrente bem maior que a do que o Arduino suporta passe pelo motor.

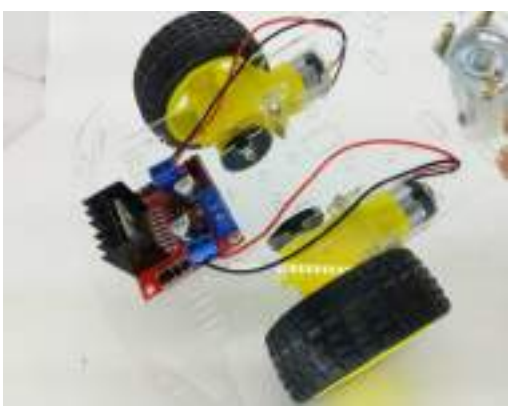
O módulo ponte H usado é desta forma:

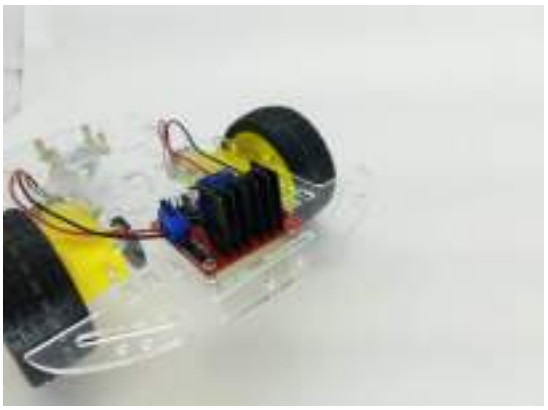


Onde iremos, inicialmente, fazer as conexões dos motores 1 e 2. Inicialmente não será necessário preocupar a posição dos fios do motor no borne. Para instalar o motor, basta

afrouxar com uma chave de fenda ou Philips o borne, encaixar o fio do motor e apertar. Faça para os dois motores.

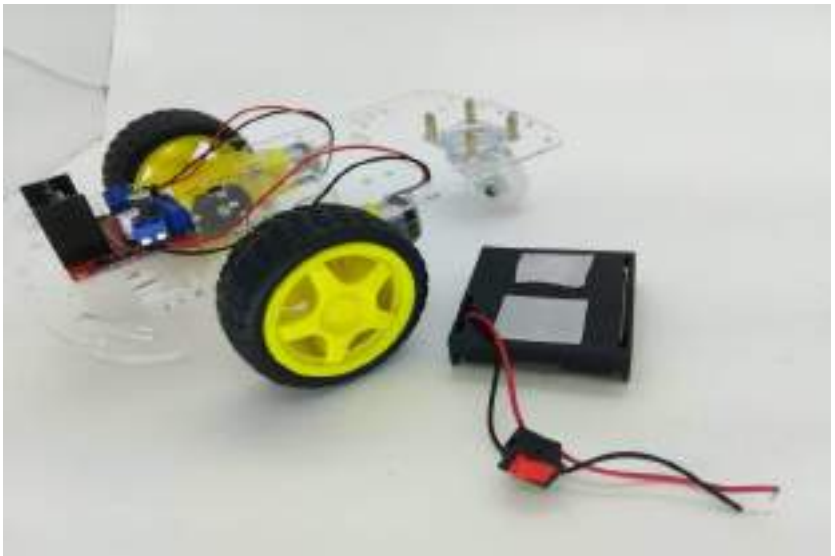
A montagem deve estar da seguinte forma:

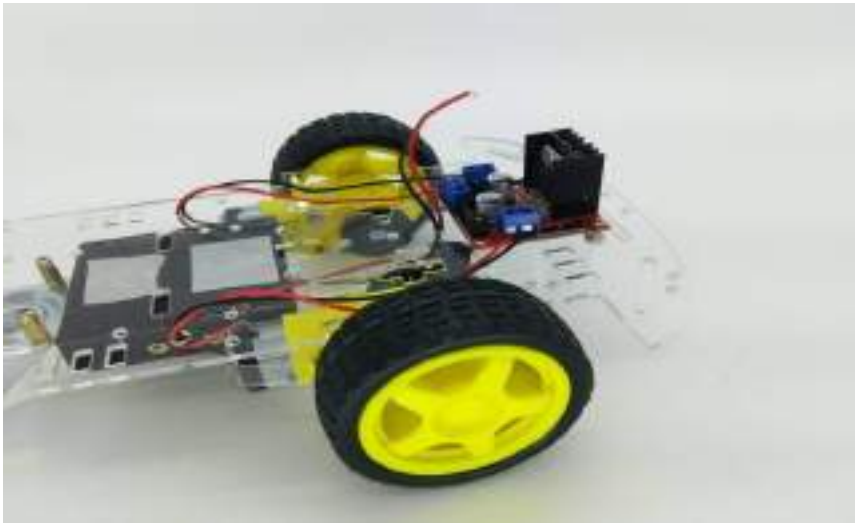




Perceba pela foto acima que há um parafuso no módulo ponte H. Esse parafuso está ligado com uma arruela ao chassi. Ela foi adicionada porque isso fará com que possamos dispensar o uso de colas e o deixar bem fixo, mas se quiser, pode usar fita dupla face para prender a ponte H ao chassi. Com a ponte H instalada e os motores conectados, faremos a instalação do suporte de pilhas.

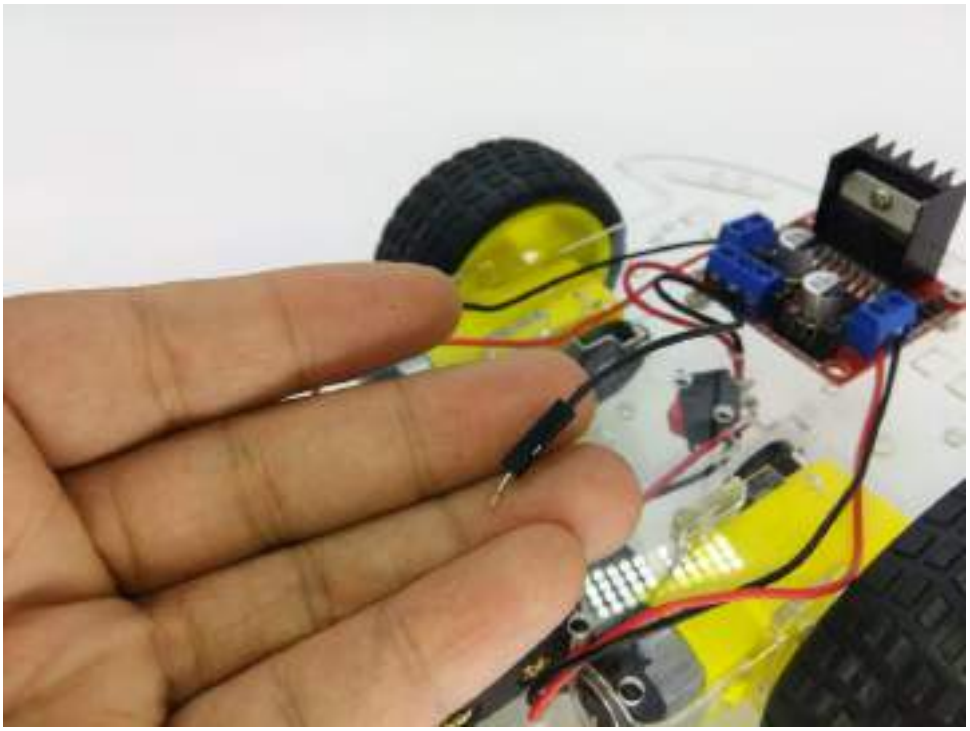
Coloque uma dupla face nas costas do suporte e então ele foi f na parte debaixo do chassi, da seguinte forma:





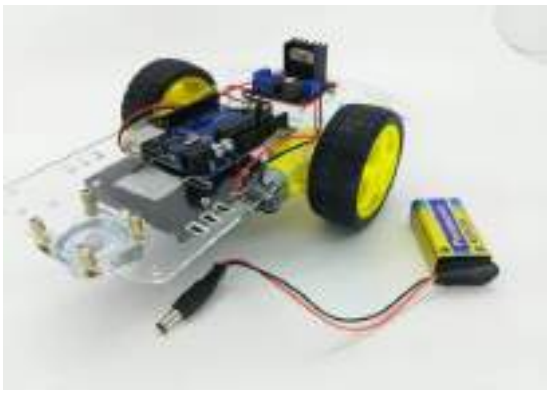
Conectando os fios de alimentação na ponte, Coloque o fio vermelho, que sai do suporte de pilhas, no borne 6V do módulo ponte H, e o preto do suporte de pilhas no borne de GND. Para consultar essas informações, basta ver a imagem da ponte H um pouco mais acima.

Coloque também um jumper macho/macho dentro do borne de GND junto com o fio preto do suporte de pilhas, porque mais tarde ele será usado. A ligação ficou assim:

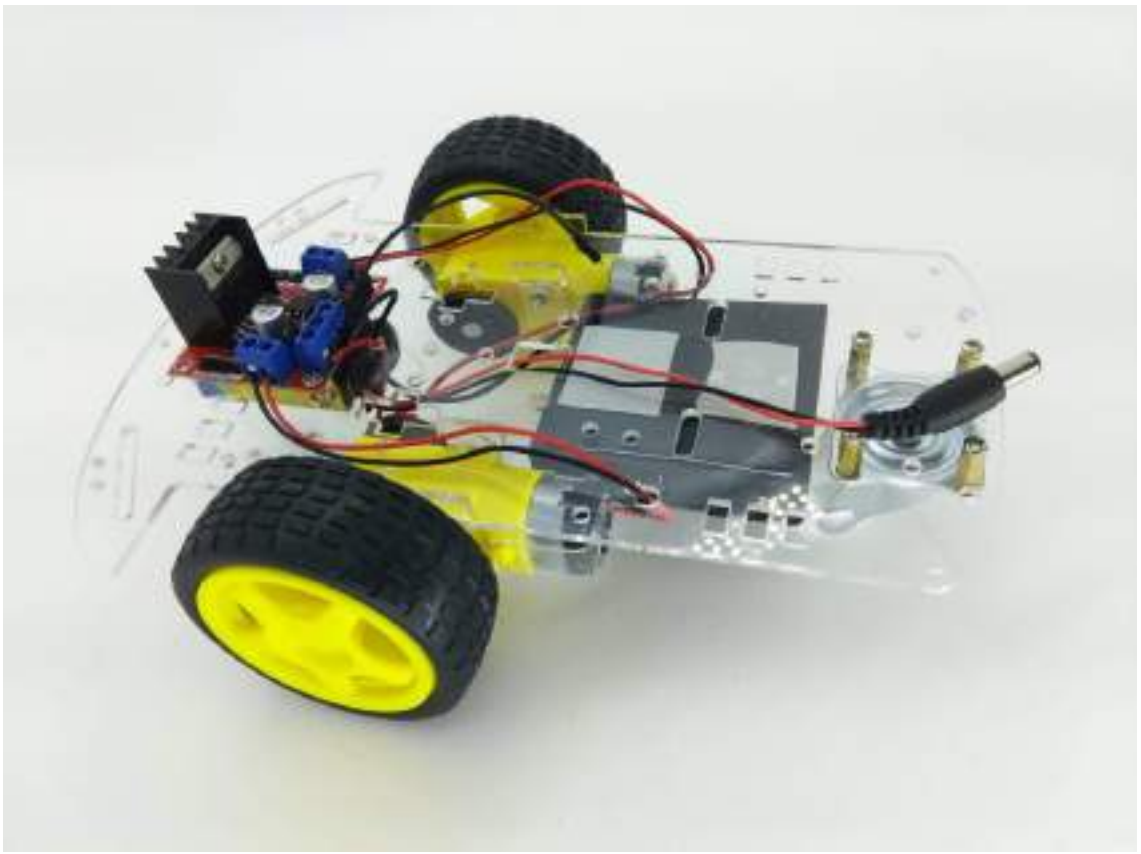


O jumper colocado é para ser usado sendo colocado na porta GND do Arduino. Agora só está faltando uma bateria de 9V, para fornecer energia ao Arduino. Essa alimentação separada foi escolhida porque as 4 pilhas possuem uma disponibilidade de energia um pouco baixa, e separando as alimentações fará que a durabilidade energética do robô aumente.

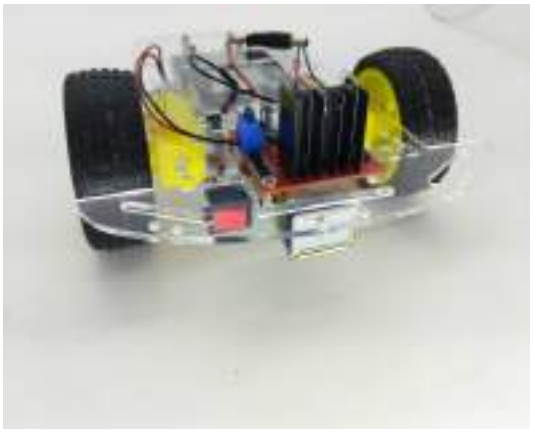
Coloque uma fita dupla face na bateria de 9V e instale-a exatamente embaixo (no chassi) do módulo ponte H. O resultado previsto é este:



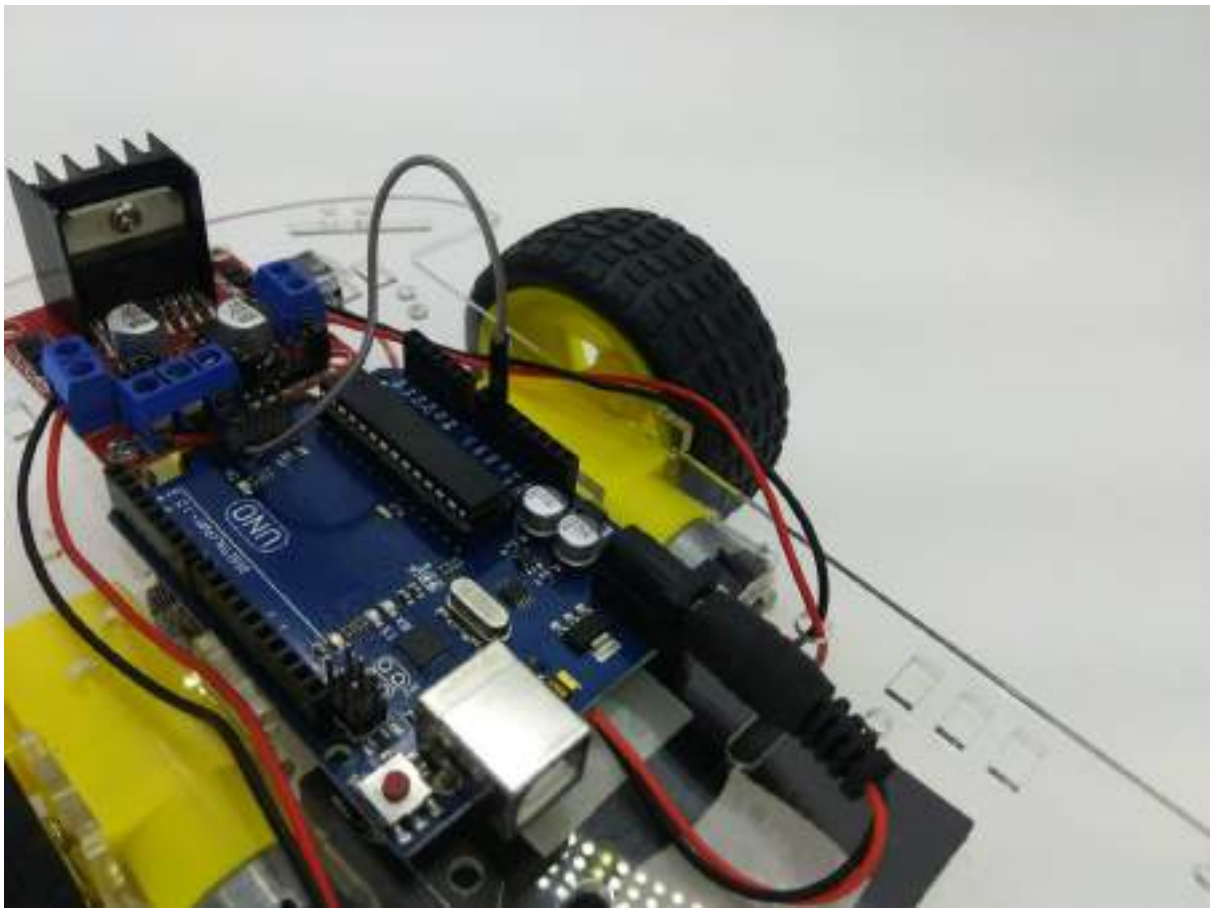
A imagem abaixo demonstra em como deve estar a montagem do robô, se caso não estiver, revise os outros passos.



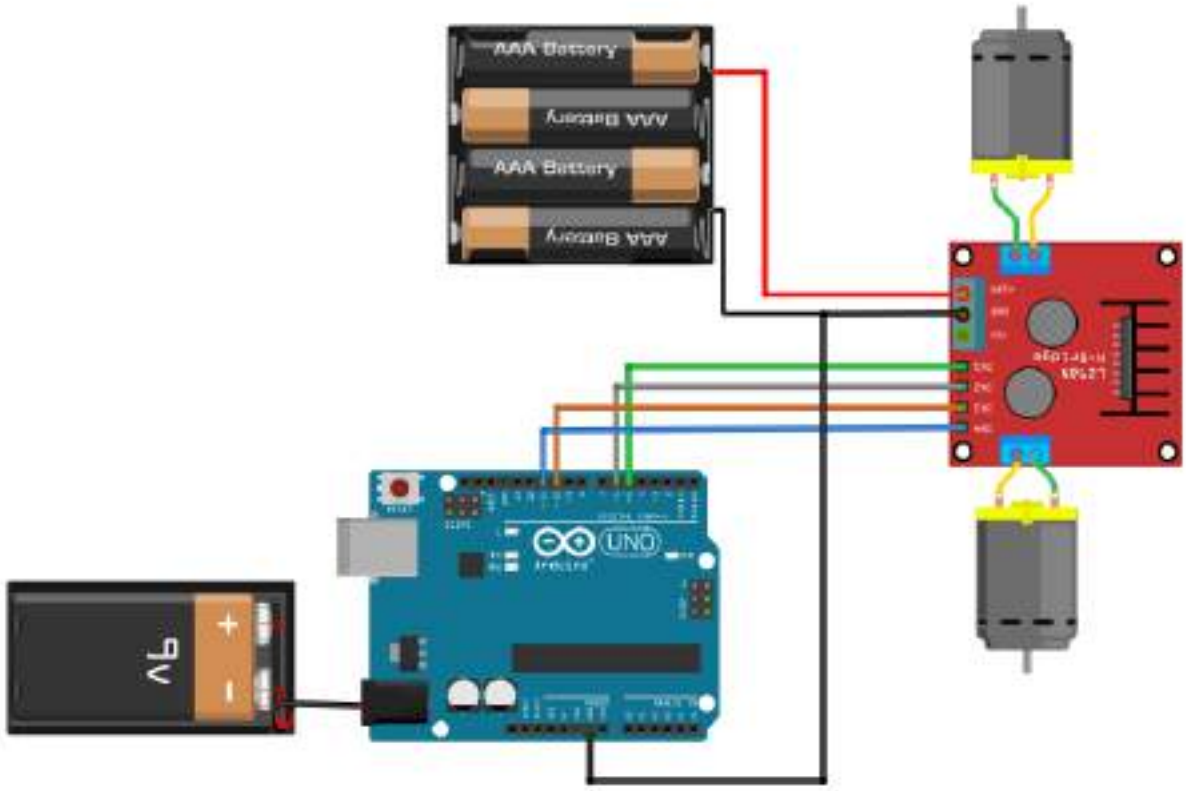
Para melhorar um pouco a estética e facilitar o uso, a chave on/off foi instalada bem na frente do carrinho, e colada com cola quente. Para isso foi preciso alongar um pouco mais o fio.



Coloque um pouco de fita dupla face na parte inferior da placa Arduino e encaixe ela na parte superior da placa, da seguinte forma:



Colocando a placa na parte central. As conexões da ponte H com o Arduino deverão ser feitas da seguinte forma:



Observações:

- É extremamente importante que os GND estejam ligados juntos, isso significa que temos que ligar o GND da ponte H com o GND do Arduino. (Estamos fazendo isso com o jumper que foi colocado junto com o GND da bateria no borne da ponte H. Esse GND foi ligado na porta GND do Arduino).
- Remova a alimentação da bateria de 9V ao colocar o cabo USB,.

Código para teste do robô:

```
// Exemplo 11 – Chassi 2 rodas
// Apostila Eletrogate - KIT Robotica


// Iremos fazer uma classe para facilitar o uso da ponte H L298N na manipulação dos motores na função
Setup e Loop.

class DCMotor {
  int spd = 255, pin1, pin2;

  public:

    void Pinout(int in1, int in2){ // Pinout é o método para a declaração dos pinos que vão controlar
o objeto motor
      pin1 = in1;
      pin2 = in2;
      pinMode(pin1, OUTPUT);
      pinMode(pin2, OUTPUT);
      }
    void Speed(int in1){ // Speed é o método que irá ser responsável por regular a velocidade
      spd = in1;
      }
    void Forward(){ // Forward é o método para fazer o motor girar para frente
      analogWrite(pin1, spd);
      digitalWrite(pin2, LOW);
      }
    void Backward(){ // Backward é o método para fazer o motor girar para trás
      digitalWrite(pin1, LOW);
      analogWrite(pin2, spd);
      }
    void Stop(){ // Stop é o metodo para fazer o motor ficar parado.
      digitalWrite(pin1, LOW);
      digitalWrite(pin2, LOW);
      }
   };

   DCMotor Motor1, Motor2; // Criação de dois objetos motores, já que usaremos dois motores, e eles já
estão prontos para receber os comandos já configurados acima.

void setup() {
  Motor1.Pinout(5,6); // Seleção dos pinos que cada motor usará, como descrito na classe.
  Motor2.Pinout(10,11);
}

void loop() {
  Motor1.Speed(200); // A velocidade do motor pode variar de 0 a 255, onde 255 é a velocidade máxima.
  Motor2.Speed(200);

  Motor1.Forward(); // Comando para o robô ir para frente
  Motor2.Forward();
  delay(1000);
  Motor1.Backward(); // Comando para o robô ir para trás
  Motor2.Backward();
  delay(1000);
  Motor1.Forward(); // Comando para o robô girar
  Motor2.Backward();
  delay(1000);
  Motor1.Stop(); // Comando para o robô parar
  Motor2.Stop();
  delay(500);
}
```

E com isso o robô deve ir para frente, para trás girar e parar, e repetir este ciclo. Altere o código para gerar mais funções.

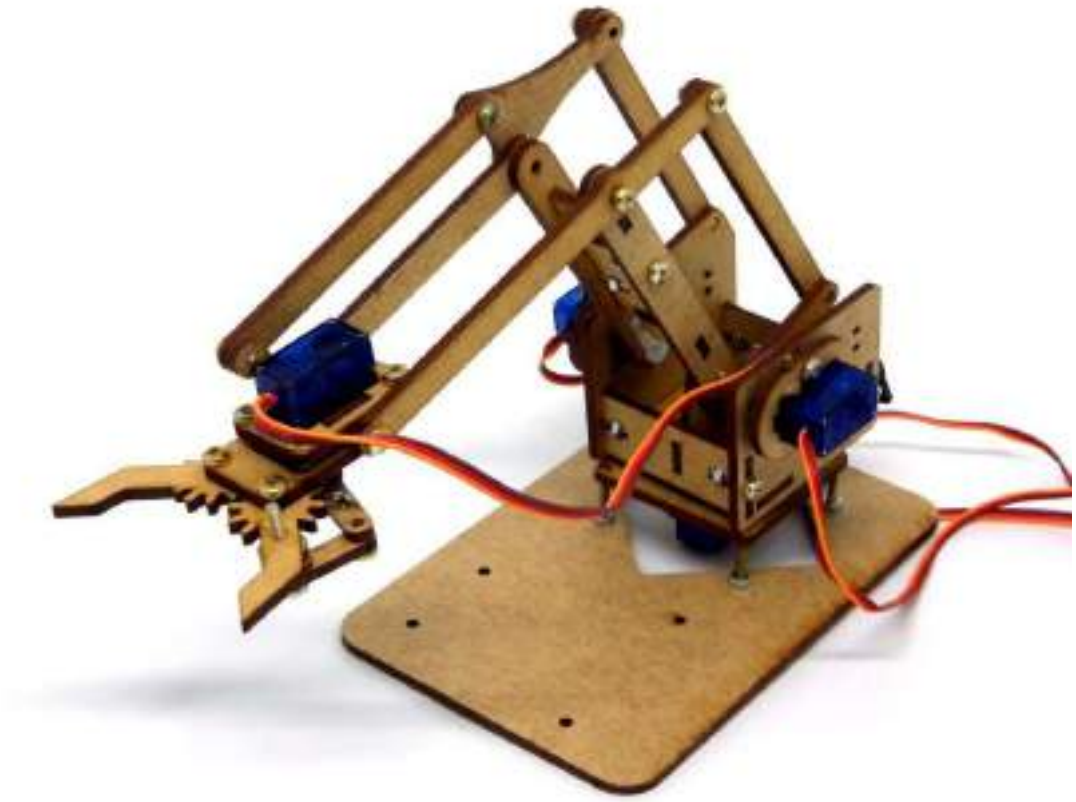
Exemplo 12 – Kit Braço Robótico

O Braço Robótico MDF é um típico braço mecânico desenvolvido para aplicações de robótica. Feito em MDF de 3mm de espessura e de forma modular e fácil de ser montado, é ideal para projetos de prototipação e validação de sistemas robóticos. Este é o equipamento ideal para ser o primeiro braço robótico de quem está começando no mundo da robótica.

As articulações do Braço Robótico são movimentadas por um conjunto com 4 servo motores TowerPro SG90. Enviando comandos a cada servo, é possível controlar a posição de seu eixo de rotação de forma a controlar as articulações do braço mecânico de forma desejada. As articulações executam movimentos de até 180°. Além de tudo, o braço também conta com uma garra de aproximadamente 60mm para segurar e soltar pequenos objetos.

As peças são todas cortadas a laser e os parafusos, porcas e demais componentes de montagem (exceto os servos) já vêm todos incluídos no kit. Pode ser montado por qualquer pessoa. Basta seguir o manual de instruções com bastante atenção. Para consultar o manual para realizar a montagem perfeita, acesse em:

http://manuais.eletrogate.com/Manual_Braco_Robotico.pdf



Tome bastante cuidado ao montar e sempre deixe as articulações não muito apertadas e nem muito frouxas, sempre com uma folga o suficiente para o braço deslizar da forma correta.

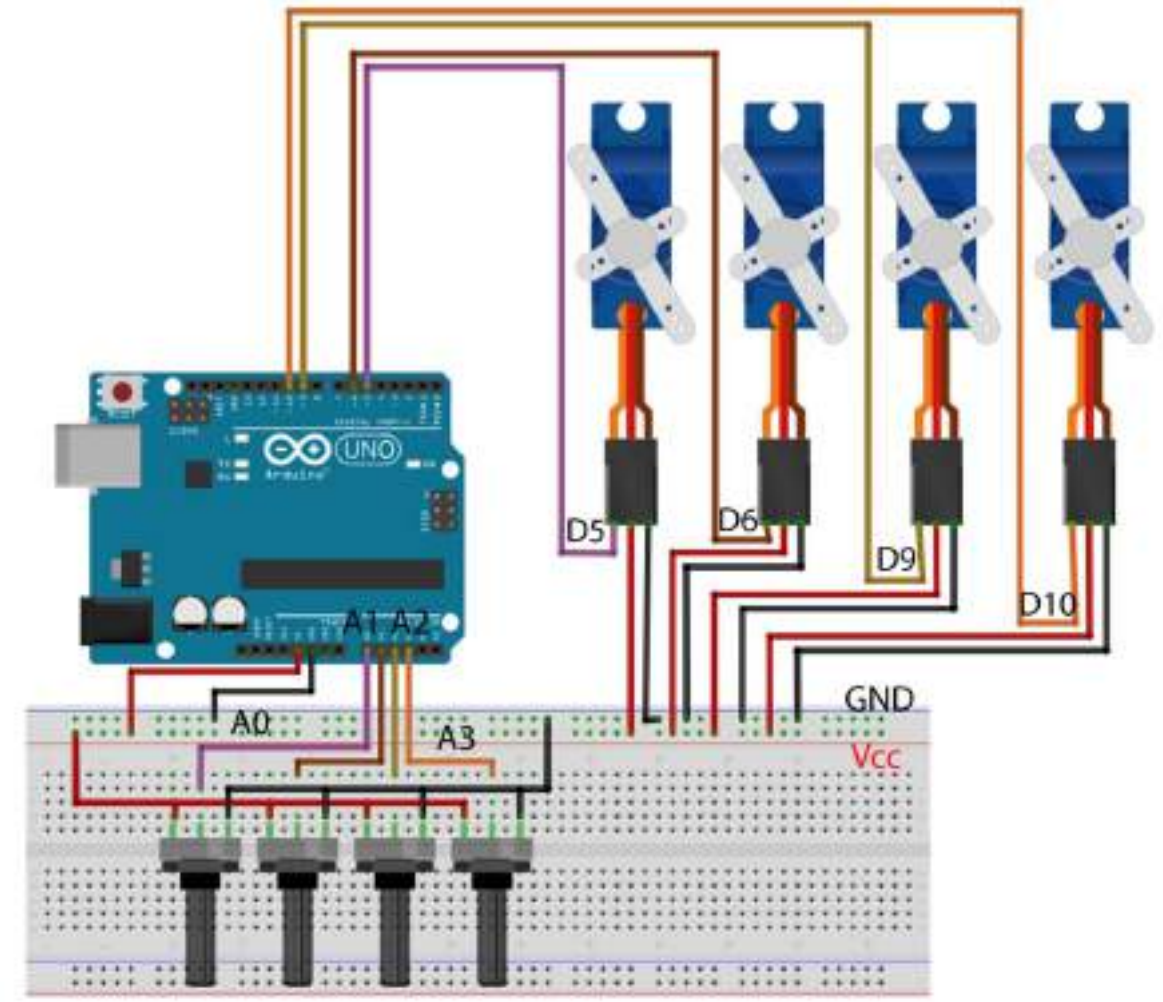
Lista de Materiais:

Para este exemplo você vai usar os seguintes componentes:

- Protoboard;
- Jumpers de ligação;
- 4 micro servos 9G
- 4 potenciômetros
- 1 Arduino Uno

Diagrama de circuito:

Depois que o braço robótico estiver montado, faça essa montagem com os servos e os potenciômetros.



Código :

Para entender o código, basta notar que primeiro definimos as portas analógicas em que são ligados os potenciômetros, que nesse exemplo é A0, A1, A2 e A3.

Após isso são criados os objetos servos, para controlar cada parte do braço robótico, e uma variável auxiliar para receber o valor dos pinos analógicos.

No loop, a gente lê o valor dos pinos analógicos e já converte esse valor para um ângulo de 0 até 179°, e escreve esse valor no objeto servo, que irá mover o servo. E faz isso para cada servo motor. Assim quando cada potenciômetro for girado, as articulações do braço robótico irão mexer.

```
// Exemplo 12 – Braço Robótico MDF
// Apostila Eletrogate - KIT Robotica

#define potpin1  0
#define potpin2  1
#define potpin3  2
#define potpin4  3

#include <Servo.h>

Servo myservoBase;  // Objeto servo para controlar a base
Servo myservoGarra;  //Objeto servo para controlar a garra
Servo myservoAltura; //Objeto servo para controlar a altura do braço
Servo myservoProfundidade; //Objeto servo para profundidade a altura do braço

int val;    // variable to read the value from the analog pin

void setup()
{
  myservoBase.attach(5); //Associa cada objeto a um pino pwm
  myservoGarra.attach(6);
  myservoAltura.attach(9);
  myservoProfundidade.attach(10);
}

void loop()
{
  val = map(analogRead(potpin1), 0, 1023, 0, 179);
  myservoBase.write(val);

  val = map(analogRead(potpin2), 0, 1023, 0, 179);
  myservoGarra.write(val);

  val = map(analogRead(potpin3), 0, 1023, 0, 179);
  myservoAltura.write(val);

  val = map(analogRead(potpin4), 0, 1023, 0, 179);
  myservoProfundidade.write(val);

  delay(100);
}
```

Considerações finais

Essa apostila tem por objetivo apresentar alguns exemplos básicos sobre como utilizar os componentes do Kit Arduino Robótica, a partir dos quais você pode combinar e fazer projetos mais elaborados por sua própria conta.

Nas seções de referências de cada exemplo e nas referências finais, também tentamos indicar boas fontes de conteúdo objetivo e com projetos interessantes. Sobre isso ponto, que consideramos fundamental, gostaríamos de destacar algumas fontes de conhecimento que se destacam por sua qualidade.

O fórum oficial Arduino possui muitas discussões e exemplos muito bons. A comunidade de desenvolvedores é bastante ativa e certamente pode te ajudar em seus projetos. No Project Hub poderá encontrar milhares de projetos com Arduino.

- Fórum oficial Arduino: https://forum.arduino.cc/
- Project Hub Arduino: https://create.arduino.cc/projecthub

O Instructables é a ótima referência do mundo maker atual. Pessoas que buscam construir suas próprias coisas e projetos encontram referências e compartilham suas experiências no site.

- Instructables: https://www.instructables.com/

O Maker pro é outro site referência no mundo em relação aos projetos com Arduino. Há uma infinidade de projetos, todos bem explicados e com bom conteúdo.

- Maker pro: https://maker.pro/projects/arduino

Em relação à eletrônica, teoria de circuitos e componentes eletrônicos em geral, deixamos alguns livros essenciais na seção finais de referências. O leitor que quer se aprofundar no mundo da eletrônica certamente precisará de um livro basilar e de bons conhecimentos em circuitos eletroeletrônicos.
No mais, esperamos que essa apostila seja apenas o início de vários outros projetos e, quem sabe, a adoção de Kits mais avançados, como o **Advanced** e **Arduino Big Jack**. Qualquer dúvida, sugestão, correção ou crítica a esse material, fique à vontade para relatar em nosso blog oficial: http://blog.eletrogate.com/

Referências gerais

1. Fundamentos de Circuitos Elétricos. Charles K. Alexander; Matthew N. O. Sadiku. Editora McGraw-Hill.
2. Circuitos elétricos. James W. Nilsson, Susan A. Riedel. Editora: Pearson; Edição: 8.
3. Microeletrônica - 5ª Ed. - Volume Único (Cód: 1970232). Sedra,Adel S. Editora Pearson.

4. Fundamentals of Microelectronics. Behzad Razavi. Editora John Wiley & Sons; Edição: 2nd Edition (24 de dezembro de 2012).

Abraços e até a próxima!

Vitor Vidal
Engenheiro eletricista, mestrando em eng. elétrica e apaixonado por eletrônica, literatura, tecnologia e ciência. Divide o tempo entre pesquisas na área de sistemas de controle, desenvolvimento de projetos eletrônicos e sua estante de livros.

Partes editadas por Vitor Vidal:
Parte I - Revisão de circuitos elétricos e componentes básicos
Parte II - Seção de Exemplos Práticos - Exemplos 1 ao 8 e 12

Gustavo Murta
Técnico em eletrônica, formado em Curso superior de TPD, com pós-graduado em Marketing. Trabalhou por muitos anos na IBM na área de manutenção de computadores de grande porte. Aposentou-se, podendo curtir o que mais gosta: estudar e ensinar Tecnologia. Hobbista em eletrônica desde 1976. Gosta muito de Fotografia e Observação de aves.

Partes editadas por Gustavo Murta:
Revisão de todas as partes editadas por Vitor Vidal.
Exemplo 9 – Módulo Sensor de presença PIR + Buzzer

Gustavo Nery
Cursando Engenharia de Controle e Automação pela UFMG. Apaixonado por eletrônica, computação e tecnologias na área de sistemas embarcados. Nos tempos livres me divido entre desenvolver pesquisa na universidade, adquirir novos conhecimentos e estar com a família.

Partes editadas por Gustavo Nery:
Revisão geral da apostila;
Exemplo 10 – Ponte H Dupla
Exemplo 11 – Kit Chassi 2 Rodas
Exemplo 12 – Kit Braço Robótico

Versão 1.0 – Maio de 2020

APOSTILA KIT

ARDUINO ROBÓTICA

Esta apostila acompanha o **Kit ARDUINO ROBÓTICA** da Eletrogate, e contém conteúdos relacionados a todos os componentes do Kit.

WWW.ELETROGATE.COM