

Introdução ao carrinho seguidor de linha

Um robô seguidor de linha é composto de duas partes básicas: uma correspondente à percepção da linha e outra correspondente aos motores e seu acionamento.

Em síntese, o funcionamento é simples: enquanto a linha for percebida pelos sensores os motores recebem energia e fazem com que o robô se desloque.

Na prática, há complicadores envolvidos. Por exemplo, para seguir em linha reta, todas rodas devem estar na mesma velocidade e com a mesma tração (a mesma força contra o solo). Se uma das rodas estiver em condições diferentes de velocidade, ou se ‘derrapar’ devido à falta de aderência com o traçado (perder a tração), o avanço não será em linha reta, mas tenderá a virar para o lado de menor velocidade ou que esteja ‘travando’ (o lado de menor velocidade/ travando agirá como uma espécie de freio, enquanto o outro lado continuará a avançar normalmente, fazendo o carrinho ‘girar’).

Como todo projeto de criação, a primeira versão nem sempre será a melhor, pois partiremos de uma imaginação de que tudo é perfeito, e, quando colocarmos em prática, teremos que realizar ajustes – inclusive devidos à tolerância dos componentes e imperfeições de nossa ‘construção’. Isto é normal, faz parte do processo; deve ser encarado como desafio, como aprendizagem, e não como motivo de desânimo.

Para todo o problema, deve existir uma solução. Portanto, uma vez identificado o problema, poderemos nos debruçar sobre uma solução. Identificar o que não está de acordo com o esperado pode não ser tão simples: exige atenção, análise, noção do projeto e do que ele deveria fazer, observação do que não está correto, noção do funcionamento das partes. Portanto, vamos avaliar as duas partes (sensores e motores) que comporão nosso projeto separadamente, depois vamos juntá-las.

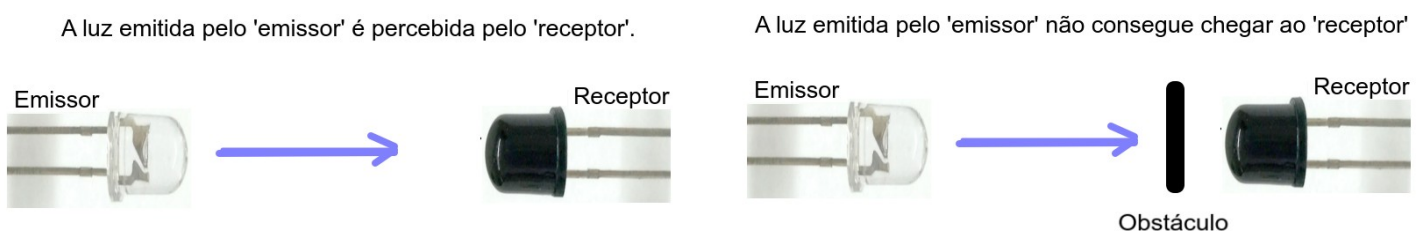
Sensor de linha reflexivo

O tipo de sensor a ser adotado depende do tipo de linha. Por exemplo, podemos ter um traçado magnético ou eletromagnético, o que nos obrigará a utilizar um sensor capaz de perceber campos magnéticos. Podemos ter uma linha física em 3D, caso em que teremos que usar um sensor capaz de ‘sentir’ a linha (que poderia, por exemplo, ser uma parede, e o sensor um interruptor). Podemos ter um traçado em forma de labirinto, no qual um sensor de ultrassom poderia ser adequado à percepção de obstáculos. Obviamente, podemos ter que combinar diferentes sensores em função de finalidades diferentes de ‘percepção’ do ambiente.

O mais comum em torneios de robótica é um traçado em forma de linha, por exemplo com uma linha preta sobre superfície branca. Vamos iniciar nossos estudos com este traçado como meta; e, portanto, usando um sensor capaz de perceber esta linha.

Um sensor comumente utilizado em um carrinho seguidor de linha preta é o sensor reflexivo. Por ser o mais comum, veremos como ele funciona.

O sensor é composto por dois componentes, um capaz de emitir radiação infravermelha (uma ‘luz’ que não podemos ver), e outro capaz de percebê-la. Um uso comum destes dispositivos é em sistemas capazes de perceber a interrupção de um fluxo luminoso (como a passagem de um objeto ou uma pessoa), que poderia, por exemplo, ser utilizado em um sistema de alarmes:



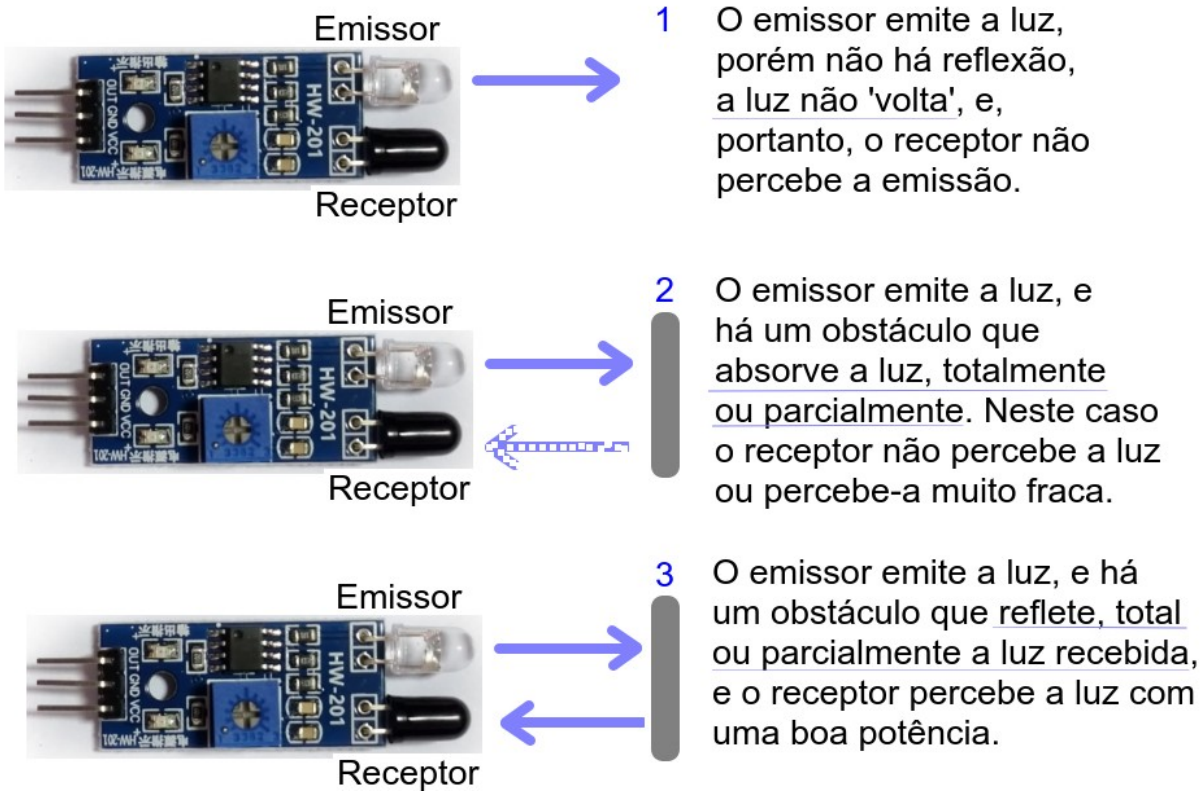
A figura representa esta situação: perceber que a luz foi interrompida cria a necessidade de alguma ação (devemos notar que um dos componentes é ‘escuro’: ele é opaco para a luz visível, porém transparente para a luz de frequência infravermelha).

Os sensores reflexivos podem ser elaborados a partir de componentes discretos (‘sozinhos’), como exibido na figura anterior, ou podem vir ‘prontos’ em pequenos módulos, como os exibidos a seguir (imagens ‘Google’):

				
1 - Sensor Infravermelho Reflexivo E18-D80NK	2 - Sensor de Obstáculos Reflexivo Infravermelho	3 - Sensor Óptico Reflexivo TCRT-5000	4 - Sensor Óptico Reflexivo Tcrt5000	5 - Sensor Óptico Reflexivo IR QRE1113 - SMD

A escolha do sensor será realizada em função da sensibilidade, confiabilidade, disponibilidade, limitação financeira – e também, claro, em função de sua disponibilidade. Quando temos somente o conjunto sensor (emissor/ receptor), como no caso dos exemplos ‘4’ e ‘5’ da figura, teremos uma sensibilidade relativamente menor do que quanto utilizados estes sensores em um módulo que já trás também um amplificador do sinal (ver ‘2’ e ‘3’): na figura ‘2’, os dois componentes discretos ‘emissor’ e ‘receptor’ vem soldados em uma placa que possui um amplificador de sinal; no caso da figura ‘3’ também temos o amplificador, porém com um sensor diferente (o da figura ‘5’) soldado nela.

Seja qual for o conjunto adotado, o funcionamento será o mesmo: o emissor emite uma luz infravermelha (IR, de *InfraRed* em Inglês), e o receptor é capaz de percebê-la. A figura que segue apresenta três situações possíveis em relação à emissão e recepção desta luz:

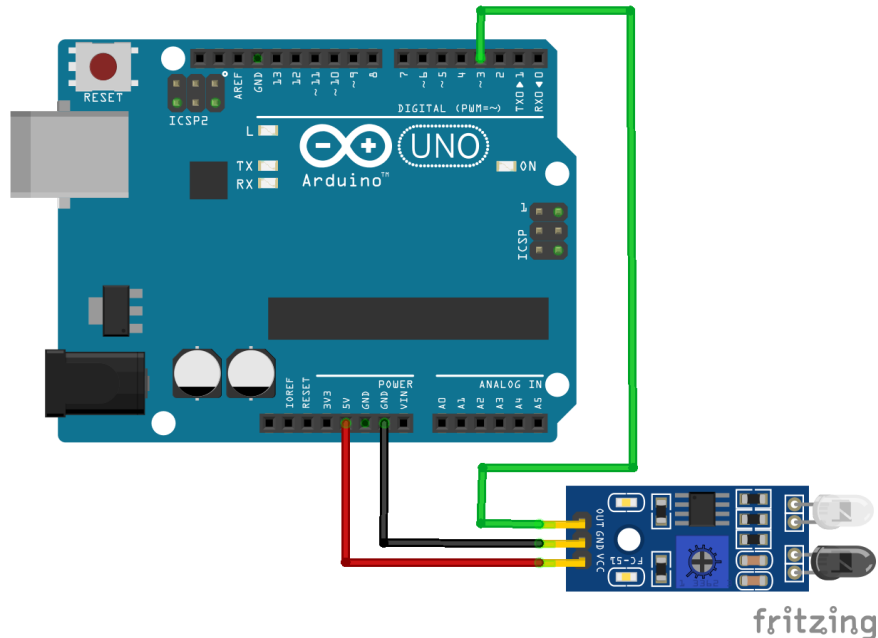


Na situação ‘1’ a luz foi emitida, porém não houve reflexão (por exemplo, o objeto refletor está ‘longe demais’ para a sensibilidade do conjunto infravermelho utilizado). Neste caso, embora tenha sido emitida a luz corretamente, não houve retorno e nenhum tipo de circuito elétrico/ lógico poderá ser construído a partir desta situação, pois o ‘receptor’ nunca terá a resposta adequada. Na situação ‘2’ houve um retorno parcial da luz

emitida. Para este caso, poderíamos tentar: aumentar a potência de emissão; amplificar o sinal recebido pelo receptor; ou, deixar o refletor mais próximo ou com maior poder reflexivo. Finalmente, no exemplo '3' temos a situação ideal: o sinal foi emitido e corretamente percebido pelo receptor.

Para o caso dos sensores utilizados na foto de exemplo há na plaquinha um amplificador de sinal, e um 'trimpot' para ajuste de amplificação/ sensibilidade. Receber um bom sinal é nosso primeiro desafio.

Vamos treinar:

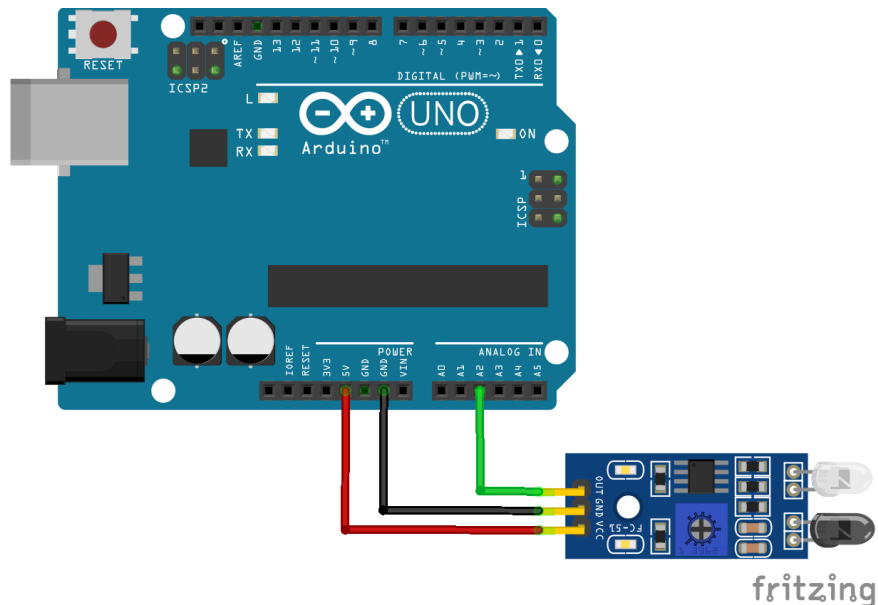


No exemplo da figura acima, o sensor reflexivo foi conectado na porta digital 3 do Arduino. Neste caso, teremos um nível lógico ALTO (HIGH) ou BAIXO (LOW) percebido naquela porta, conforme o sensor possua ou não reflexão detectada (a sensibilidade pode ser ajustada pelo 'trimpot' na plaquinha). Vejamos o código de controle:

```
reflexivo1robo $
1 #define sensorReflexivo 3 // saída do sensor no pino 3
2
3 void setup(){
4   pinMode(sensorReflexivo, INPUT); // a porta 3 será uma entrada de sinal
5   Serial.begin(9600);
6 }
7
8 void loop(){
9   if (digitalRead(sensorReflexivo) == LOW){ // se for baixo
10     Serial.println("Detectado");
11   } else {
12     Serial.println("Não detectado");
13   }
14   delay(2000);
15 }
```

O código é bem simples: a porta 3 (escolhida na linha 1, para acompanhar o desenho do *hardware*) é definida como entrada (INPUT, na linha 4) e seu valor é lido na linha 9 e o nível do sinal testado; se for nível baixo (LOW) é exibida uma mensagem (linha 10), se não for é exibida outra mensagem (linha 12). Após 2 segundos (linha 14), o ciclo é reiniciado. As mensagens são exibidas na interface serial (definida na linha 5).

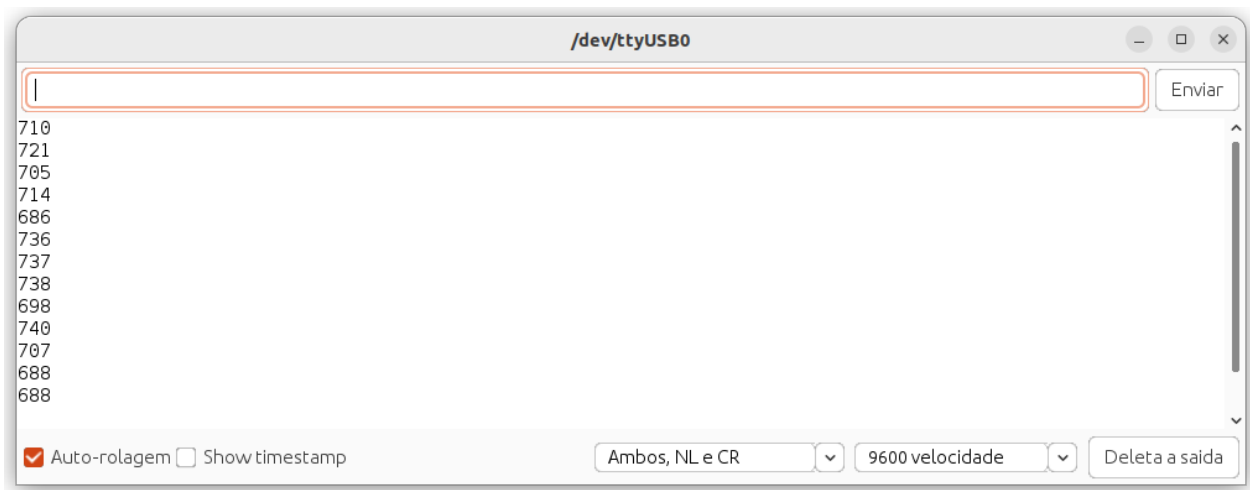
Este circuito e seu código são úteis para quando se quer detectar a presença ou ausência de um objeto (sua reflexão, na verdade): um alarme, um contador de passagens, de proximidade de uma parede ou outro obstáculo, etc. Mas, também pode ser usado em montagens de seguidores de linha, percebendo, ou não, uma linha que deve ser seguida. A desvantagem é que ele é no estilo binário, ou seja, ‘tudo ou nada’: ou há ou não há reflexão. Muitas vezes precisamos saber o ‘quanto’ foi refletido; para estes casos, usamos uma configuração diferente, mostrada a seguir.



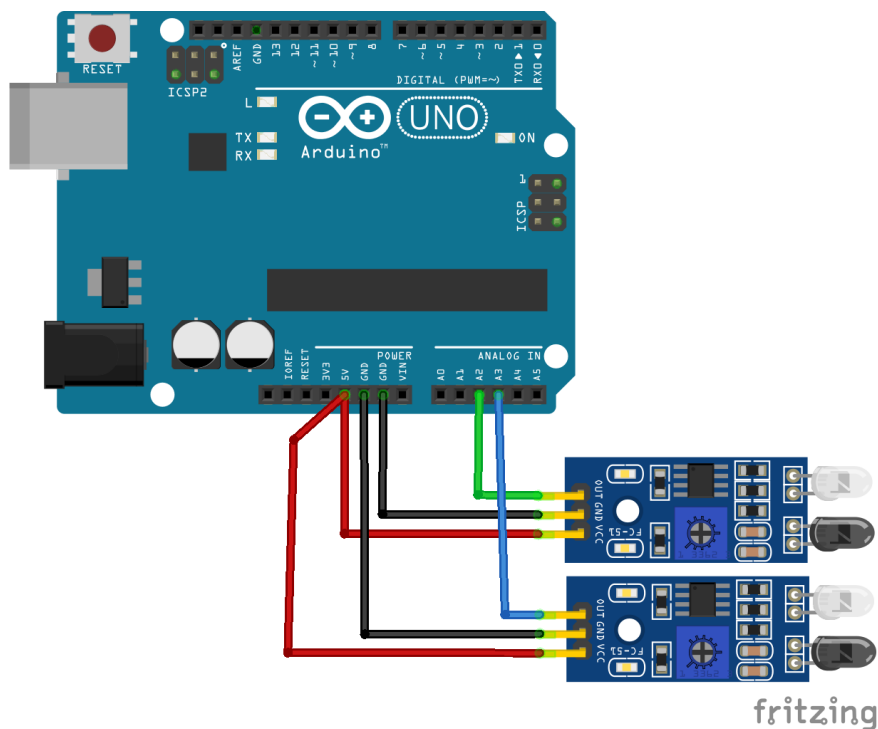
Para o exemplo de *hardware* acima fizemos a conexão de um módulo sensor, o qual teve sua saída (amplificada) conectada na porta analógica A2 do Arduino. O código é o que segue:

```
reflexivo2robo
1 #define sensorReflexivo A2 // saída do sensor no pino A2
2
3 void setup(){
4   pinMode(sensorReflexivo, INPUT); // a porta A2 será uma entrada de sinal
5   Serial.begin(9600);
6 }
7
8 void loop(){
9   Serial.println(analogRead(sensorReflexivo));
10  delay(2000);
11 }
```

Agora temos a conexão da saída do sensor realizada na porta analógica A2 e definida no *software* na linha 1; na linha 4 é definido que a porta será de entrada e na linha 5 que usaremos a interface serial. Na linha 10 instruímos o Arduino a exibir (**Serial.println**) o resultado da leitura do valor do sensor (**analogRead(sensorReflexivo)**) na interface serial, e, na linha 10, aguardamos 2 segundos antes de nova leitura. Monte o circuito, execute o código e veja os valores por meio da interface serial. Experimente aproximar e afastar objetos de diversas cores do sensor e veja quais são as leituras realizadas. Ajuste a sensibilidade. Teste (para referência, os números que obtive com meus sensores seguem na cópia de tela da próxima figura - os seus poderão ser diferentes):



Os carrinhos ou robôs seguidores de linha podem ter vários sensores, por exemplo uma ‘régua’ com 5 sensores, garantindo sensibilidade e percepção de linhas e de suas mudanças para os lados. Os mais comuns utilizam dois sensores, para perceber o que está à esquerda e/ ou à direita da linha, bem como a mudança de direção da linha. O hardware poderia (poderia pois você pode montar de outra forma; se for este o caso, não esqueça de ajustar o código para informar as conexões realizadas) ser assim:

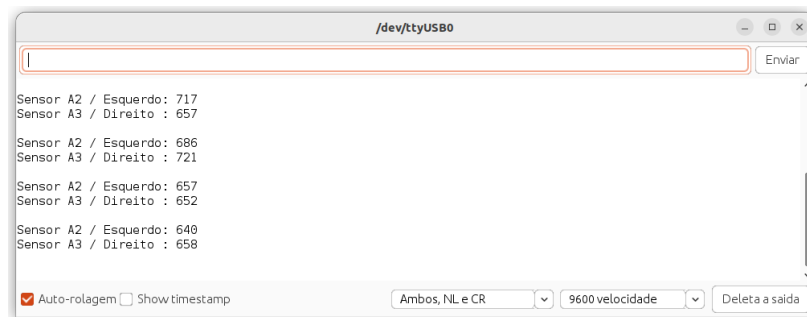


Obviamente, teríamos que ter um novo código para efetuar a leitura do segundo sensor. Poderia ser assim:

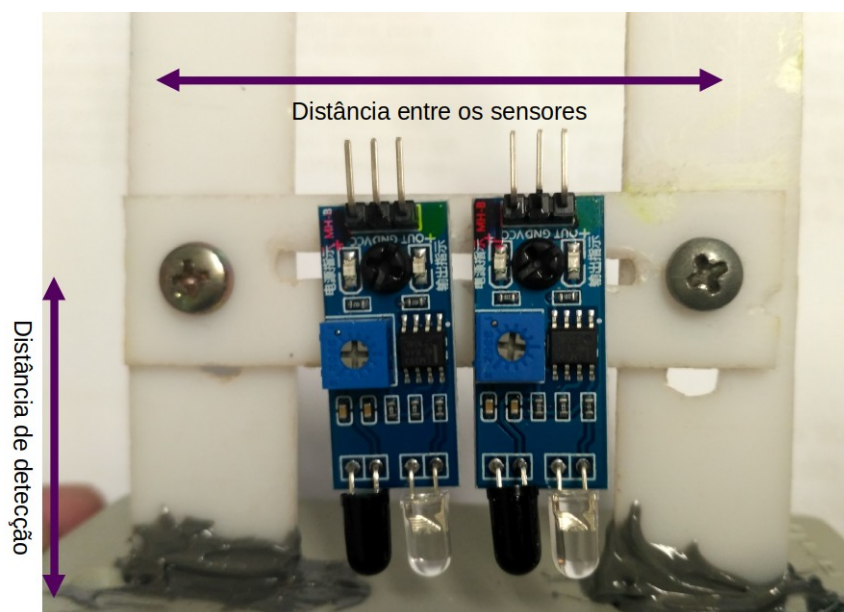
reflexivo3robo

```
1 #define sensorReflexivoEsquerdo A2 // saída do sensor no pino A2
2 #define sensorReflexivoDireito A3 // saída do sensor no pino A3
3
4 void setup(){
5   pinMode(sensorReflexivoEsquerdo, INPUT); // a porta A2 será uma entrada de sinal
6   pinMode(sensorReflexivoDireito, INPUT); // a porta A3 será uma entrada de sinal
7   Serial.begin(9600);
8 }
9
10 void loop(){
11   Serial.print("Sensor A2 / Esquerdo: ");
12   Serial.println(analogRead(sensorReflexivoEsquerdo));
13   Serial.print("Sensor A3 / Direito : ");
14   Serial.println(analogRead(sensorReflexivoDireito));
15   Serial.println(" ");
16   delay(2000);
17 }
```

Repetindo o código anterior, foi incluído um segundo sensor e modificada a mensagem, de forma a sabermos os valores por sensor. Teste (os números que obtive seguem na cópia de tela que segue, os seus poderão ser diferentes):



Pode ser útil testar e ajustar as distâncias dos sensores entre si e do solo antes de prosseguir com o controle dos motores. Para isso você pode ajustar os sensores já em seu carrinho ou calibrá-los antes e depois usar as mesmas medidas. Veja uma montagem para isso (você pode usar sua impressora 3D para criar um suporte semelhante):

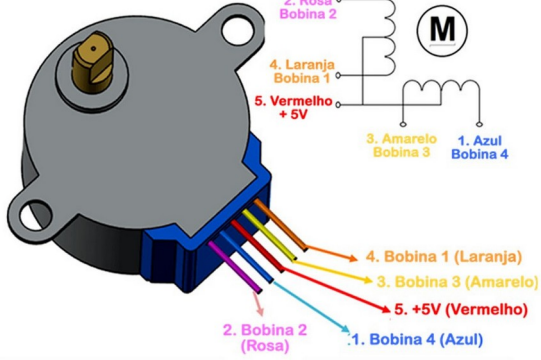


A luz de um sensor poderá interferir no outro, portanto é necessário testar e ajustar cada caso em função do posicionamento dos sensores (veja exemplos de ajustes aqui: <https://tiaplicada.ufpr.br/wp-content/uploads/2024/08/sensoresreflexivos.pdf>).

Os motores

Agora, os motores. Temos dois tipos básicos de motores a serem utilizados: motores de corrente contínua, e, motores de passo (que também são de corrente contínua mas tem um controle diferente).

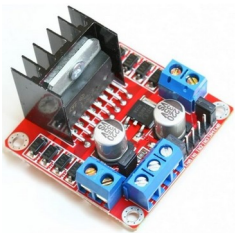
Os aspectos destes motores são exibidos a seguir:

	
Motor de corrente contínua (DC, <i>Direct Current</i>)	Motor de passo (<i>stepper motor</i>)

Os de corrente contínua, uma vez que recebam energia, irão girar indefinidamente. Se a energia for fornecida de forma inversa (trocando o negativo com o positivo), eles irão girar em sentido oposto e permanecerão girando enquanto houver energia.

Os motores de passo tem um funcionamento diferente: o sentido de rotação (e a própria rotação) depende não de um, mas do fornecimento de um conjunto de valores (em função de seu conjunto de bobinas). Os mais comuns tem um conjunto de 5 fios, sendo duas bobinas com pontos comuns. Para o controle do motor de passo usamos um controlador específico, capaz de alimentar com energia suas bobinas de forma a criar um campo eletromagnético sequencial (e, com isso, um ‘passo’): a sequência de passos provoca um giro; se a mudança for rápida o motor parecerá estar girando continuamente. Este tipo de motor é usado quando queremos maior precisão (por exemplo em um mecanismo de posicionamento para impressão).

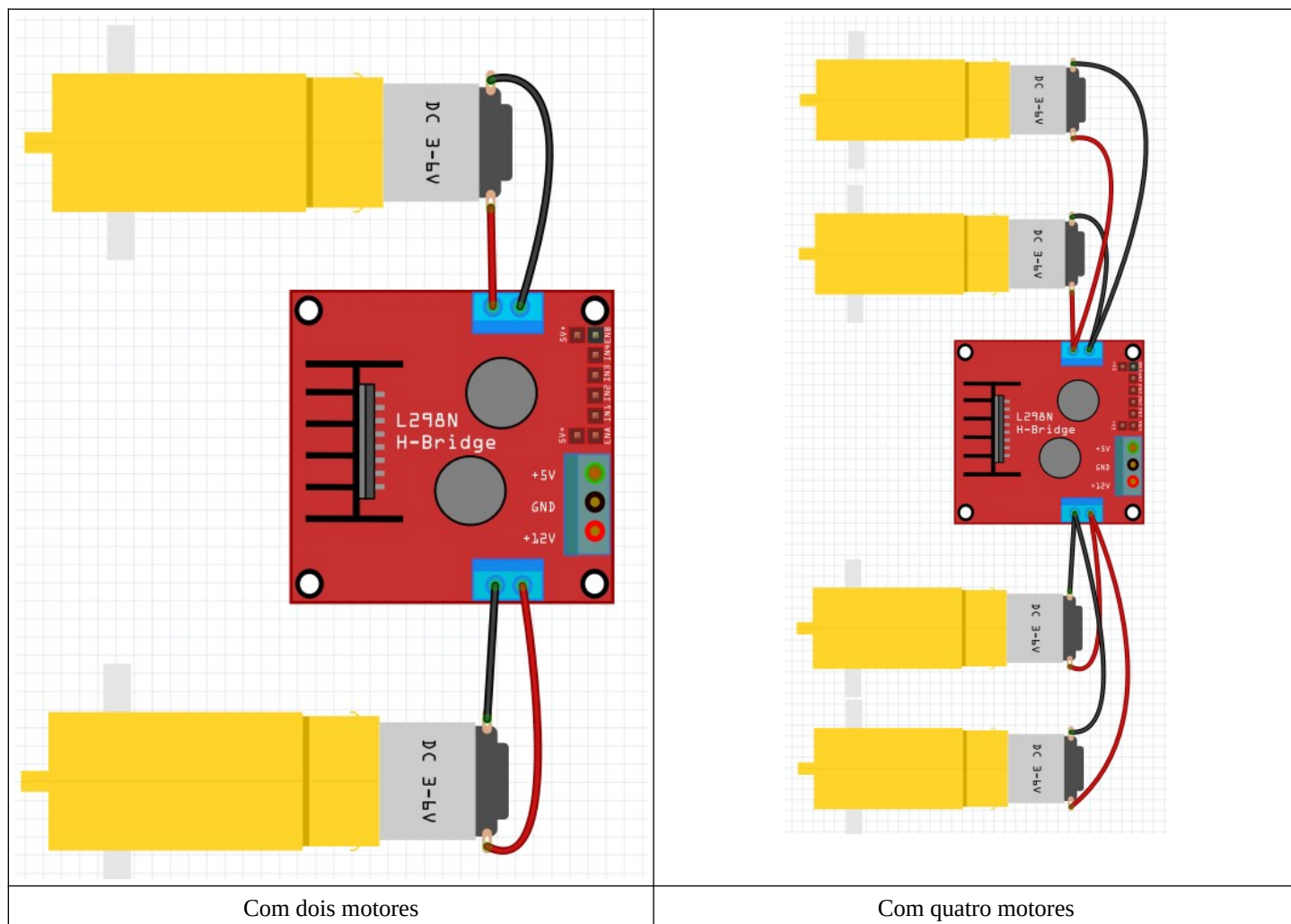
Utilizaremos em nosso primeiros exemplos motores de corrente contínua. Novamente, temos grande variedade de formas de controlar os motores/ módulos disponíveis. Alguns exemplos:

			
1 - Módulo <i>Driver</i> Ponte H ou Motor de Passo – L298	2 - Módulo Ponte H Dupla HG7881 (L9110S)	3 - Módulo <i>Driver</i> Pwm	2 - Controlador <i>Shield</i> L293D De 4 Motores DC

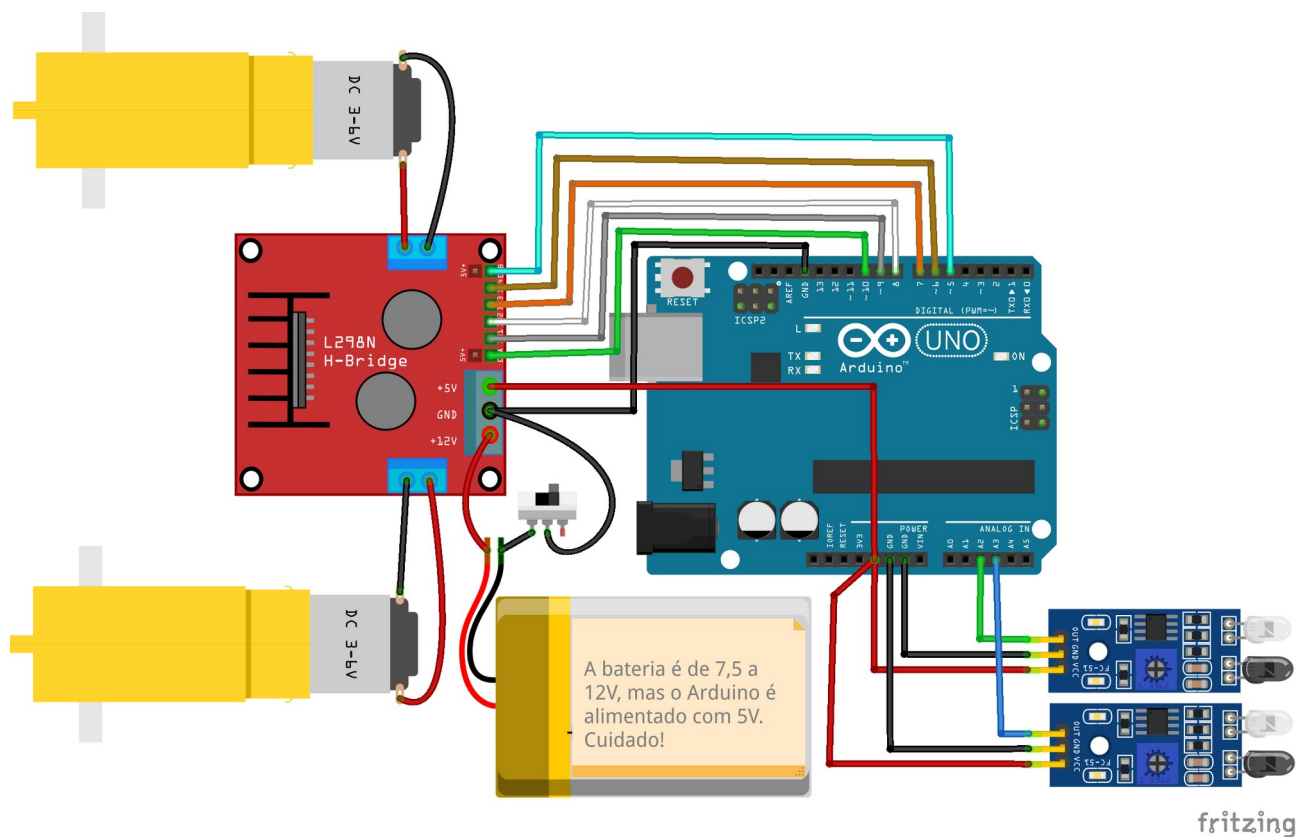
A seleção do módulo, como no caso dos sensores, dependerá das características necessárias, disponibilidade do módulo e de recursos para adquiri-lo, dos controles que desejamos implementar, do número de motores a controlar, ...

No exemplo que segue, vamos iniciar com o primeiro tipo (controlador L298).

Nosso *hardware* ficará assim:



SE utilizarmos 4 motores, podemos ligá-los em paralelo dois a dois, como exibido acima, ou podemos ter dois controladores, um para cada conjunto de dois motores. De qualquer forma, é importante que os motores tenham as características o mais próximas possíveis e girem em velocidades o mais iguais possíveis, pois caso contrário será muito difícil seguir uma linha reta. O conjunto de *hardware* para motores + sensores poderia ficar assim:



Note que foi inserido um pequeno interruptor, para podermos desligar o circuito. E, principalmente, que, embora a alimentação dos motores esteja sendo realizada por uma fonte de tensão entre 7,5 ou 7,4 e 12V, **a alimentação do Arduino será sempre de 5V!**

Vamos ver uma primeira versão de código (e, aqui, poderemos ter uma infinidade de variações / ajustes):

(os sensores deverão ser ajustados para valores altos em linha 'branca' e baixos em linha 'preta')

```

1  /*
2  ver: https://circuitbest.com/line-follower-robot-car-with-arduino-uno-l298n-driver-ir-sensor/
3
4  Arduino Uno   L298N Motor Driver
5  Pin 10        ENA
6  Pin 9         IN1
7  Pin 8         IN2
8  Pin 7         IN3
9  Pin 6         IN4
10 Pin 5         ENB
11
12 obs.: os motores poderiam estar habilitados previamente por jumpers na placa
13
14 */
15
16 #define in1 9
17 #define in2 8
18 #define in3 7
19 #define in4 6
20 #define Motor1 10
21 #define Motor2 5
22
23
24 int M1_velocidade = 80;
25 int M2_velocidade = 80;
26 int VelocidadeGiroEsquerdo = 250;
27 int VelocidadeGiroDireito = 250;
28
29
30 void setup() {
31   pinMode(in1,OUTPUT);
32   pinMode(in2,OUTPUT);
33   pinMode(in3,OUTPUT);
34   pinMode(in4,OUTPUT);
35   pinMode(Motor1,OUTPUT);
36   pinMode(Motor2,OUTPUT);
37   pinMode(A2, INPUT); // sensor esquerdo
38   pinMode(A3, INPUT); // sensor direito
39 }
40
41 void loop() {
42   int SensorEsquerdo = digitalRead(A2);
43   int SensorDireito = digitalRead(A3);
44   if(SensorDireito==0 && SensorEsquerdo==0) {
45     emFrente();
46   }
47   else if(SensorDireito==0 && SensorEsquerdo==1) {
48     moverDireita();
49   }
50   else if(SensorDireito==1 && SensorEsquerdo==0) {
51     moverEsquerda();
52   }
53   else if(SensorDireito==1 && SensorEsquerdo==1) {
54     parar();
55   }
56 }

```

Até a linha 27 temos a definição de variáveis e de constantes.

Entre as linhas 31 e 38 a definição das entradas e saídas.

O controle do carrinho robô é realizado por meio da interação de testes que ocorre entre as linhas da função `loop()`, entre as linhas 41 e 56. Primeiramente, os sensores são lidos e seus valores armazenados nas variáveis `SensorEsquerdo` e `SensorDireito`. Em função da leitura de seus valores, são comparados os resultados e chamadas funções que comandarão o conjunto de motores.

Por exemplo, se as leituras forem iguais (teste da linha 44), será chamada a função `emFrente()`. Compreender a lógica de funcionamento (e testá-la) é o primeiro passo para programa seguidores de linha com Arduino.

Devemos notar aqui que o controle/ teste está sendo realizado de maneira simples. A evolução é testar valores de leitura dos sensores dentro de uma certa amplitude de variação, inclusive somando ou subtraindo fatores de ajuste. Faremos isso depois. Por enquanto teste o código. Eis as funções:

<pre>58 void emFrente() 59 { 60 digitalWrite(in1, HIGH); 61 digitalWrite(in2, LOW); 62 digitalWrite(in3, HIGH); 63 digitalWrite(in4, LOW); 64 analogWrite(Motor1, M1_velocidade); 65 analogWrite(Motor2, M2_velocidade); 66 } 67 </pre>	<pre>68 void paraTras() 69 { 70 digitalWrite(in1, LOW); 71 digitalWrite(in2, HIGH); 72 digitalWrite(in3, LOW); 73 digitalWrite(in4, HIGH); 74 analogWrite(Motor1, M1_velocidade); 75 analogWrite(Motor2, M2_velocidade); 76 } 77 </pre>
<pre>78 void moverDireita() 79 { 80 digitalWrite(in1, LOW); 81 digitalWrite(in2, HIGH); 82 digitalWrite(in3, HIGH); 83 digitalWrite(in4, LOW); 84 analogWrite(Motor1, VelocidadeGiroEsquerdo); 85 analogWrite(Motor2, VelocidadeGiroDireito); 86 } 87 </pre>	<pre>88 void moverEsquerda() 89 { 90 digitalWrite(in1, HIGH); 91 digitalWrite(in2, LOW); 92 digitalWrite(in3, LOW); 93 digitalWrite(in4, HIGH); 94 analogWrite(Motor1, VelocidadeGiroEsquerdo); 95 analogWrite(Motor2, VelocidadeGiroDireito); 96 } 97 </pre>
<pre>98 void parar() 99 { 100 digitalWrite(in1, LOW); 101 digitalWrite(in2, LOW); 102 digitalWrite(in3, LOW); 103 digitalWrite(in4, LOW); 104 }</pre>	O controle do sentido de rotação dos motores, para o módulo utilizado, é realizado por meio de níveis lógicos em suas entradas IN1, IN2, IN3 e IN4. A função <code>analogWrite()</code> do Arduino é usada para controlar o giro com uma velocidade maior do que a de rotação. Teste variações!

Lembre-se de que a diferença de leituras entre os sensores (calibragem distinta) e a diferença de desempenho ou de tração entre os motores vai fazer com que o comportamento saia fora do desejado e fique instável (ou impossível) seguir uma linha reta! O código pode ser melhorado para contornar algumas situações-problema.

Este conteúdo pode ser copiado, editado e distribuído livremente. PINTO, José Simão de Paula. **Introdução ao carrinho seguidor de linha**. Curitiba : Edição própria, 2024.