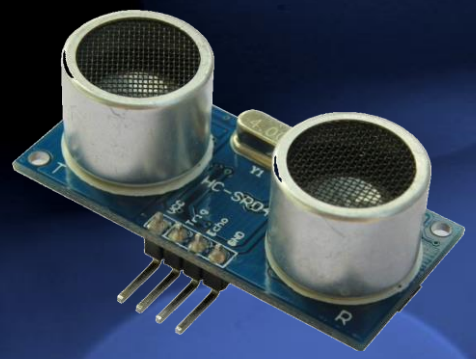
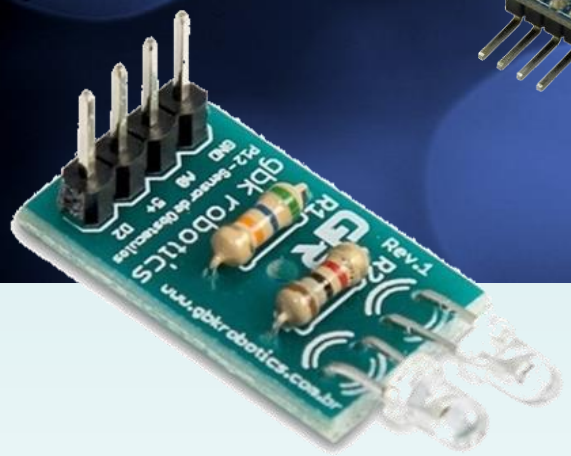
PMR 3100 – Introdução à Engenharia Mecatrônica

Módulo 04 – Meu Primeiro Robô

Aula 05 – Atuadores e Sensores

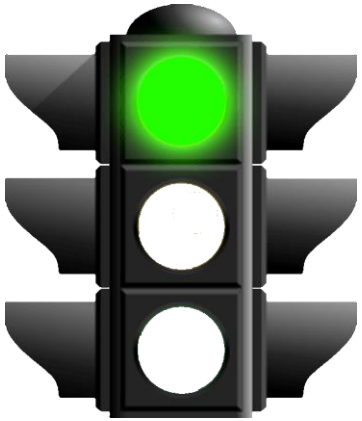


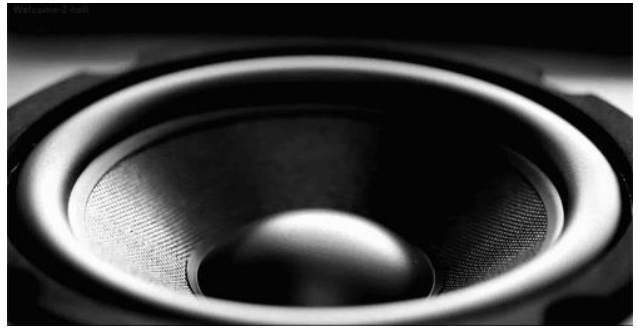


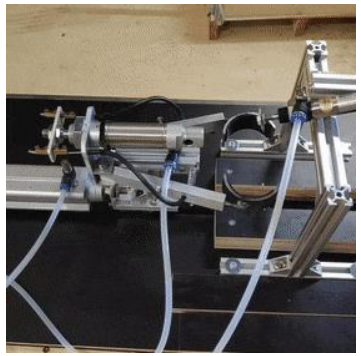
Prof. Dr. Rafael Traldi Moura

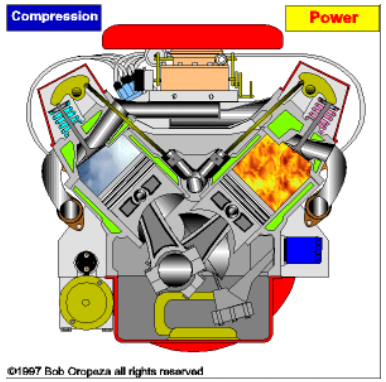
Atuadores

- Os atuadores podem ser de diversos tipos, como:
  - Luminosos;
  - Sonoros;
  - Térmicos;
  - Movimento;
- Os de movimento podem ser:
  - Pneumático;
  - Hidráulico;
  - Combustão;
  - Elétrico;
  - Etc.



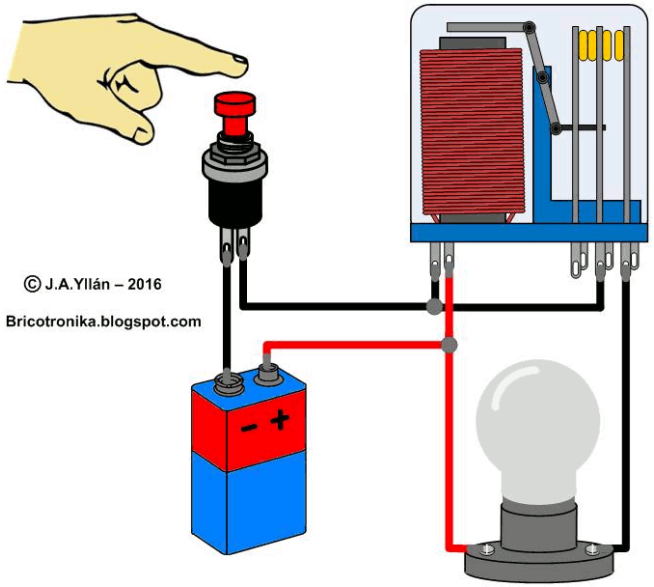


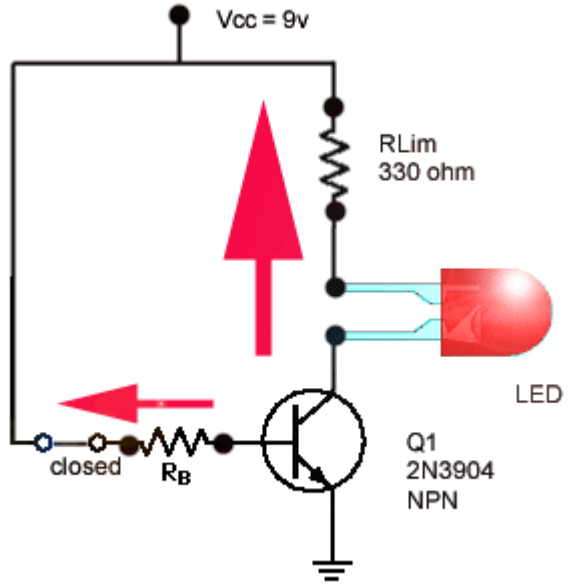




Fornecendo energia elétrica a atuadores

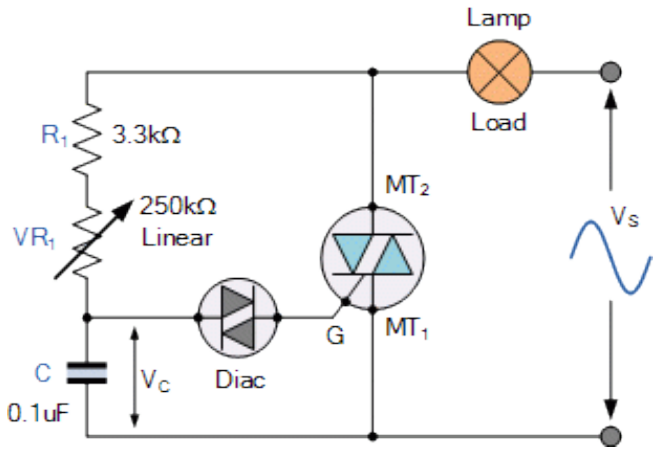
- Existem dois tipos de energia elétrica que podem ser fornecidas a atuadores:
  - Contínua;
  - Alternada.
- Em ambos os casos, podemos ativar ou desativar o atuador controlando o fornecimento ou não de energia através de um interruptor;
- Relês e transistores podem funcionar como interruptores que pode ser acionado remotamente!

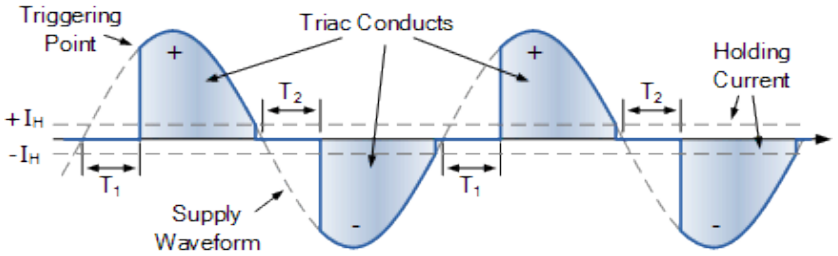


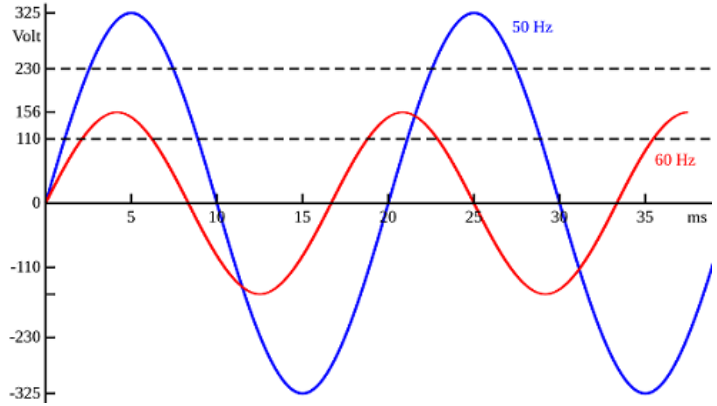


Variando o fornecimento em AC

- Existem dois modos principais de controlar o fornecimento de energia AC: i) Circuitos com Diacs e Triacs e ii) Inversores de frequência;
- No primeiro caso, somente parte da onda senoidal é aplicada a carga;
- No segundo caso, variamos a frequência e/ou amplitude da onda senoidal aplicada à carga;

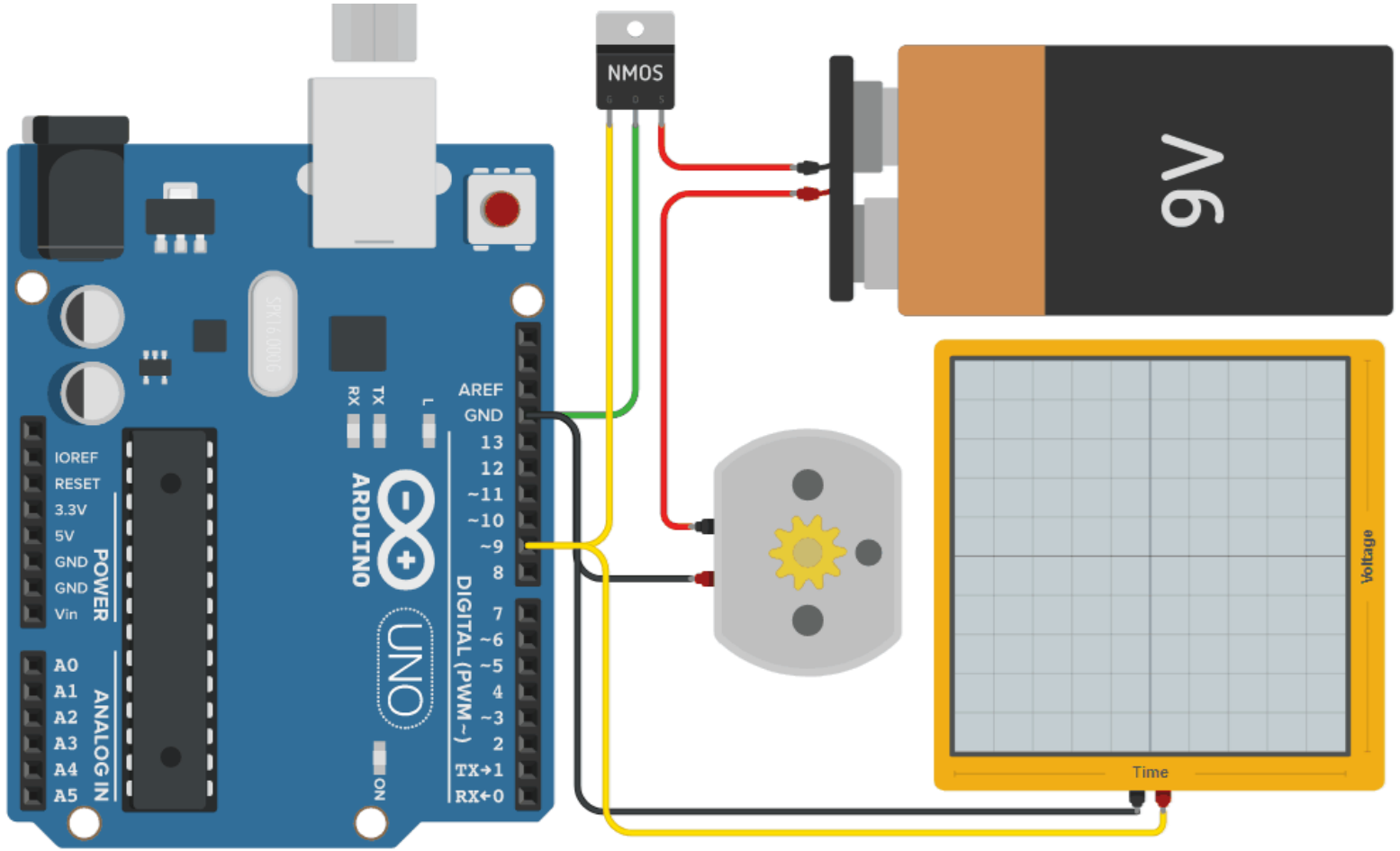






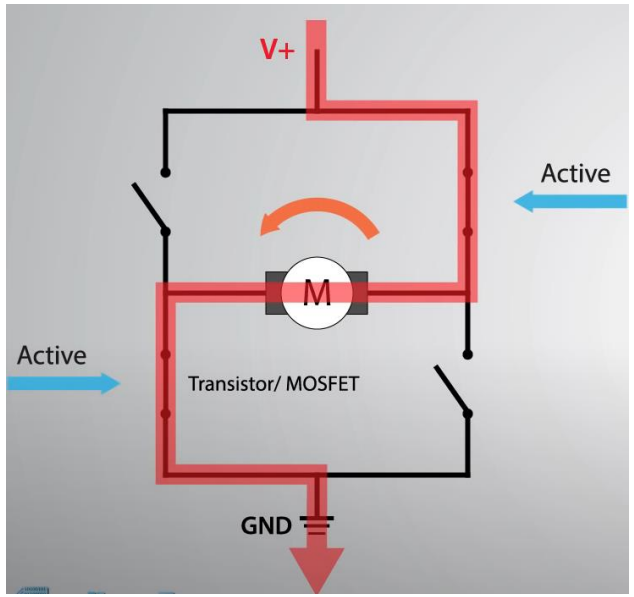
Variando o fornecimento em DC

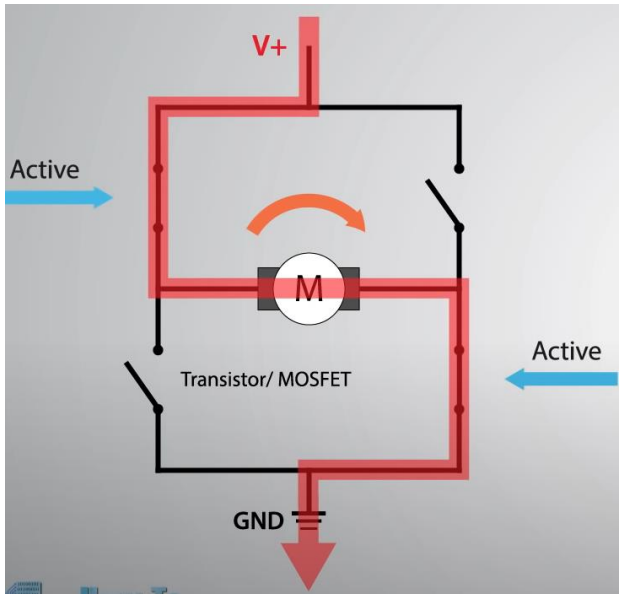
- Já no caso DC, utilizamos circuitos específicos ou microcontroladores que geram PWM;
- Como a saída destes circuitos é de baixa potência, precisamos de um driver para fornecer energia.
- Como exemplos de driver podemos ter um único transistor.

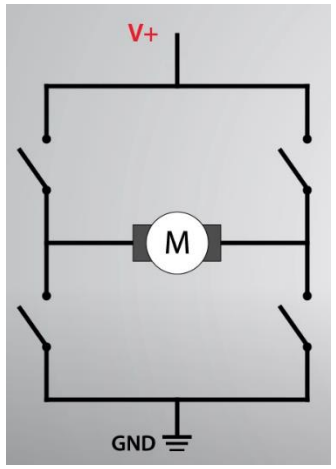


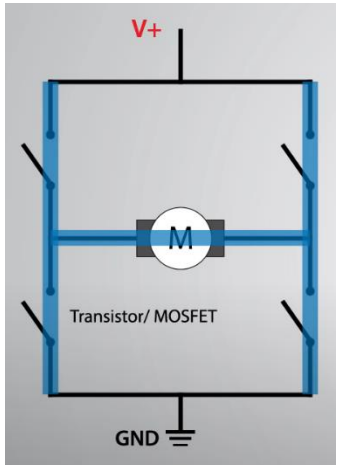
Para um motor DC

- Mas podemos querer ter um circuito que fornecem a energia desejada ao sistema, mas é capaz de inverter o sentido de rotação do motor.
- Este circuito pode ser um driver chamado ponte H.



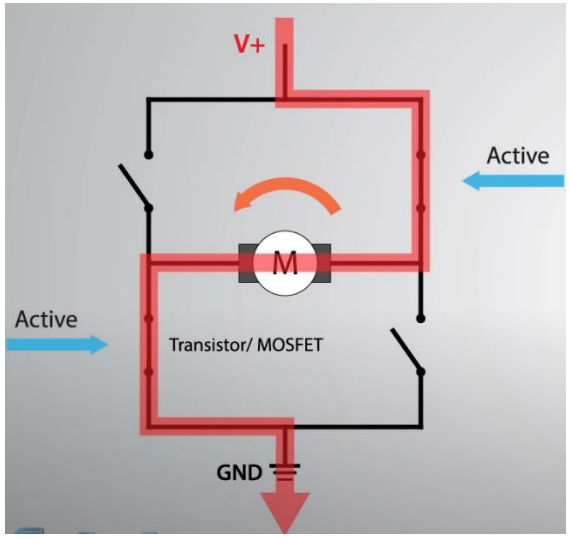


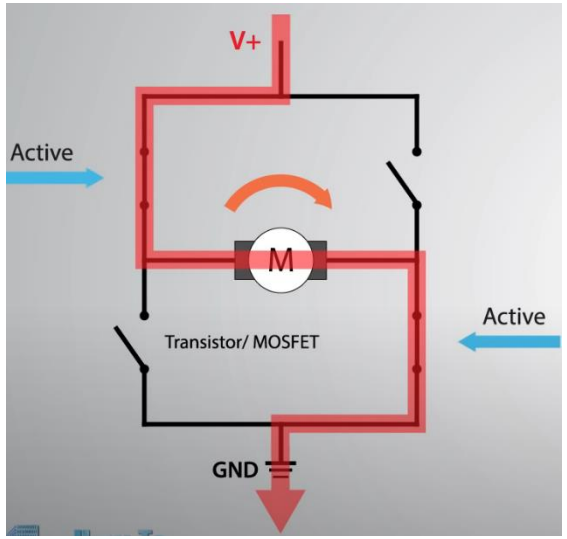


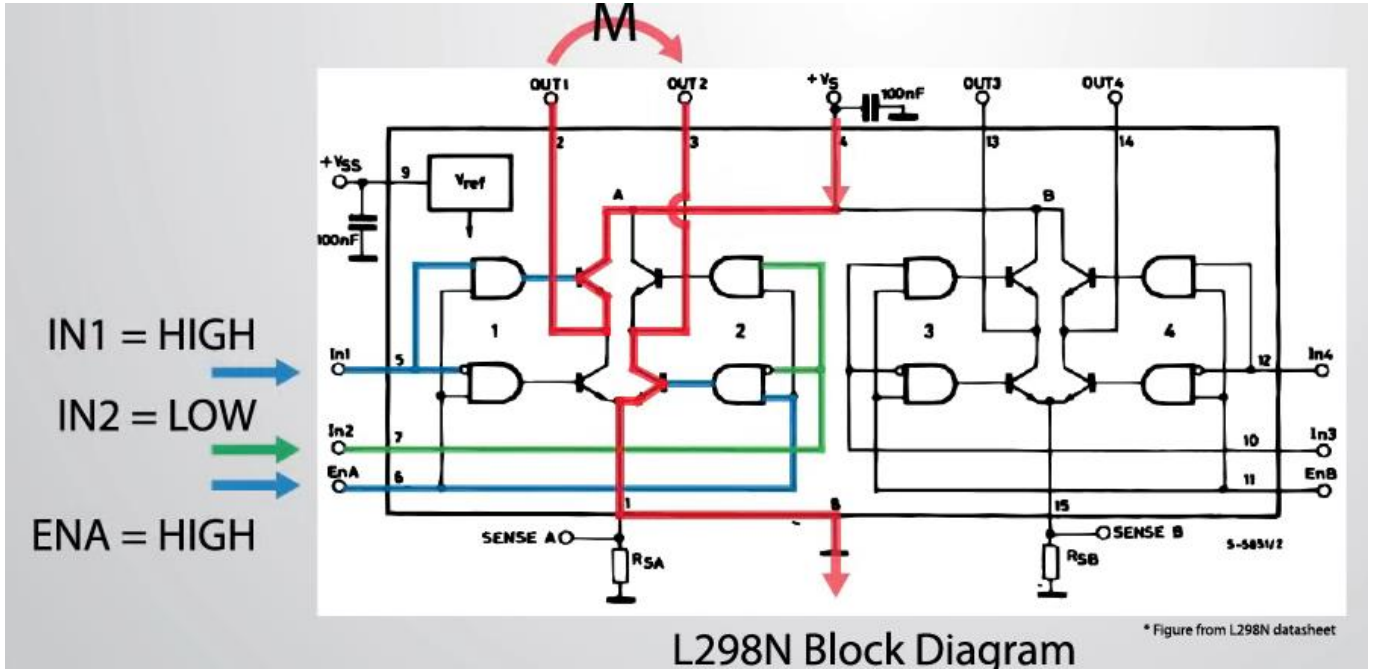


Para um motor DC

- Abaixo está o diagrama de blocos do L298N, que possui duas pontes-H.
- A velocidade do motor é controlada por um sinal PWM no pino ENABLE.
- Se colocarmos HIGH no pino IN1 e LOW no pino IN2, o motor gira para um lado.
- Se colocarmos LOW no pino IN1 e HIGH no pino IN2, o motor gira para o outro lado.

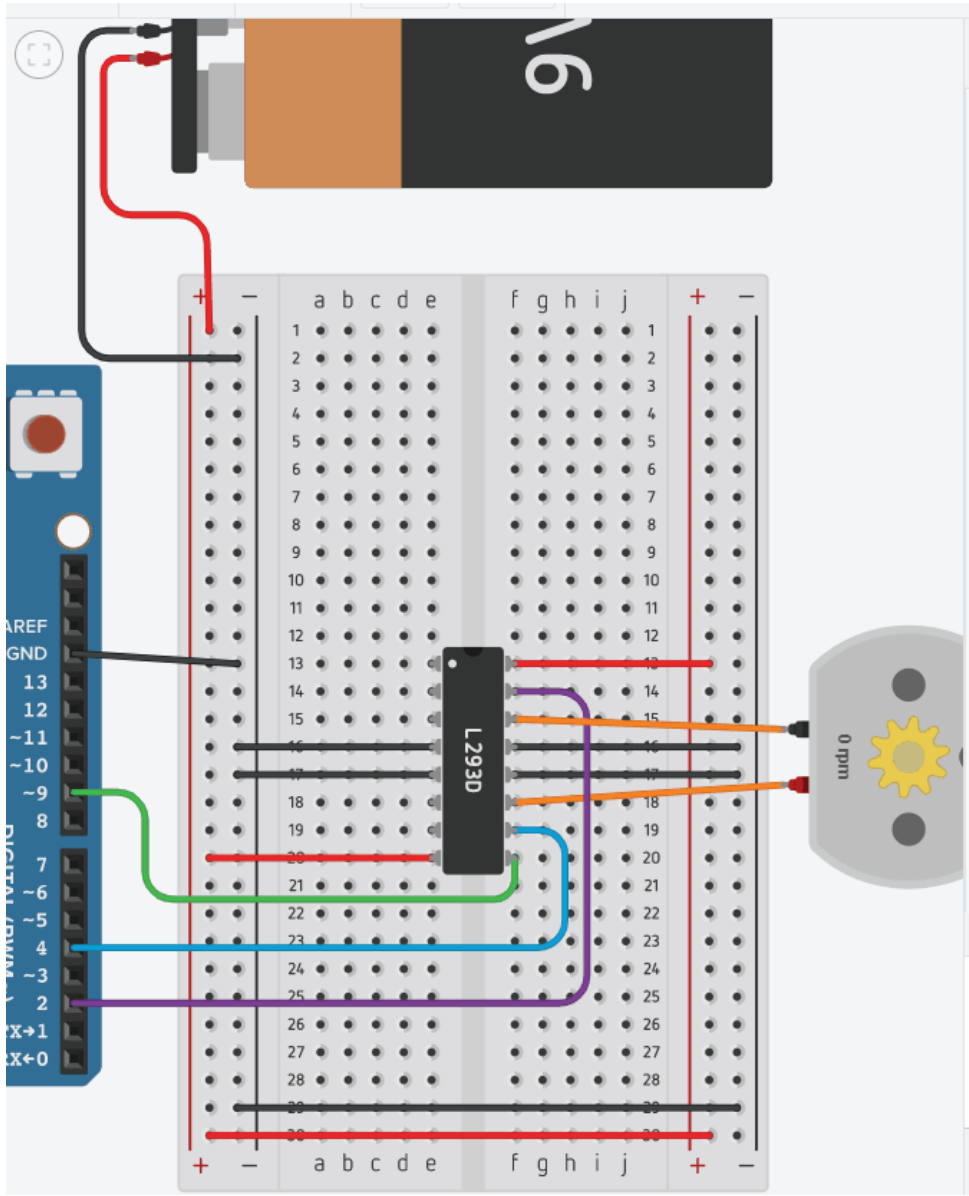


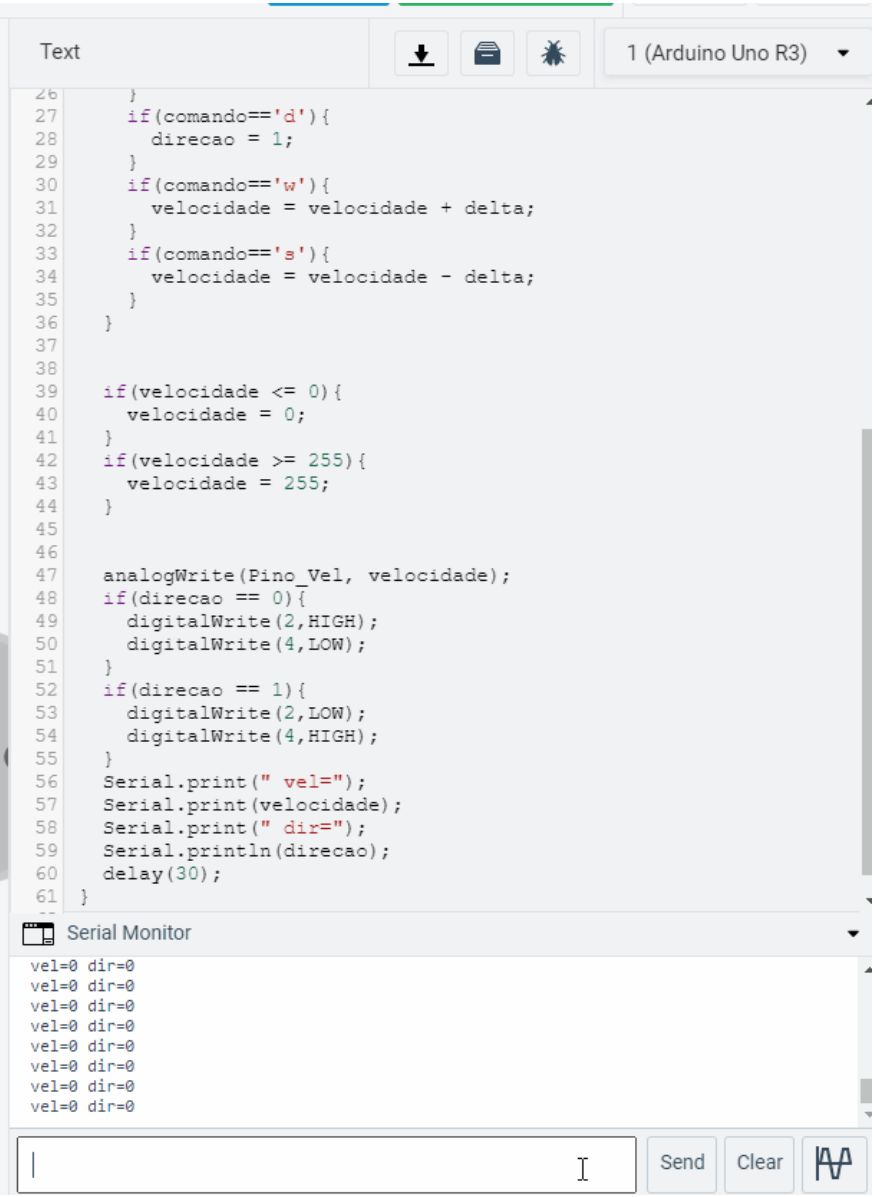




Para um motor DC - circuito

- O circuito ao lado usa a ponte H L293D, conforme explicado anteriormente;
- A fonte de energia é uma bateria que o driver passa ao motor é uma bateria;
- Comandos:
  - `w` acelera,
  - `s` desacelera,
  - `a` gira em um sentido,
  - `d` gira em outro sentido.





Para um motor DC – código parte 1

- A primeira parte do código contém:
  - define
  - declarações de variáveis globais
  - função `setup()` com:
  1. inicialização da comunicação serial em 9600kbps;
  2. definição de 3 pinos como saída (enable, IN1 e IN2);
  3. atribuição inicial de valores para as variáveis globais

```
#define Pino_Vel 9

int velocidade;
int direcao;
int delta;
char comando;

void setup() {
  Serial.begin(9600);
  pinMode(Pino_Vel, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(4, OUTPUT);
  velocidade = 0;
  direcao = 0;
  delta = 5;
  comando = 'a';
}
```

Para um motor DC – código parte 2

- A função `loop()` começa verificando se há algo no buffer da porta serial;
- Caso exista:
  - Coloca-se o valor do buffer na variável `comando`.
  - Altera-se o valor da variável `direcao` caso o comando seja `a` ou `d`;
  - Altera-se o valor da variável `velocidade` caso o comando seja `w` ou `s`.

```
void loop() {
  if(Serial.available()){
       comando = Serial.read();
    Serial.print(comando);
    Serial.print("   ");
    if(comando=='a'){
      direcao = 0;
      }
    if(comando=='d'){
      direcao = 1;
      }
    if(comando=='w'){
      velocidade = velocidade + delta;
      }
    if(comando=='s'){
      velocidade = velocidade - delta;
      }
}
```

Para um motor DC – código parte 3

- Ainda na função `loop()` verificamos se os valores para o atuador estão saturados. Caso estejam, corrigimos os valores.

```
if(velocidade <= 0){
  velocidade = 0;
}
if(velocidade >= 255){
  velocidade = 255;
}
```

Para um motor DC – código parte 1

- Por fim atualizamos as saídas, colocando o valor de PWM desejado no pino 9 (enable do driver) e ajustando os valores de IN1 e IN2 do driver de acordo com a variável `direcao`;
- Neste caso, também colocamos prints das velocidade e direção para conferência pelo usuário;

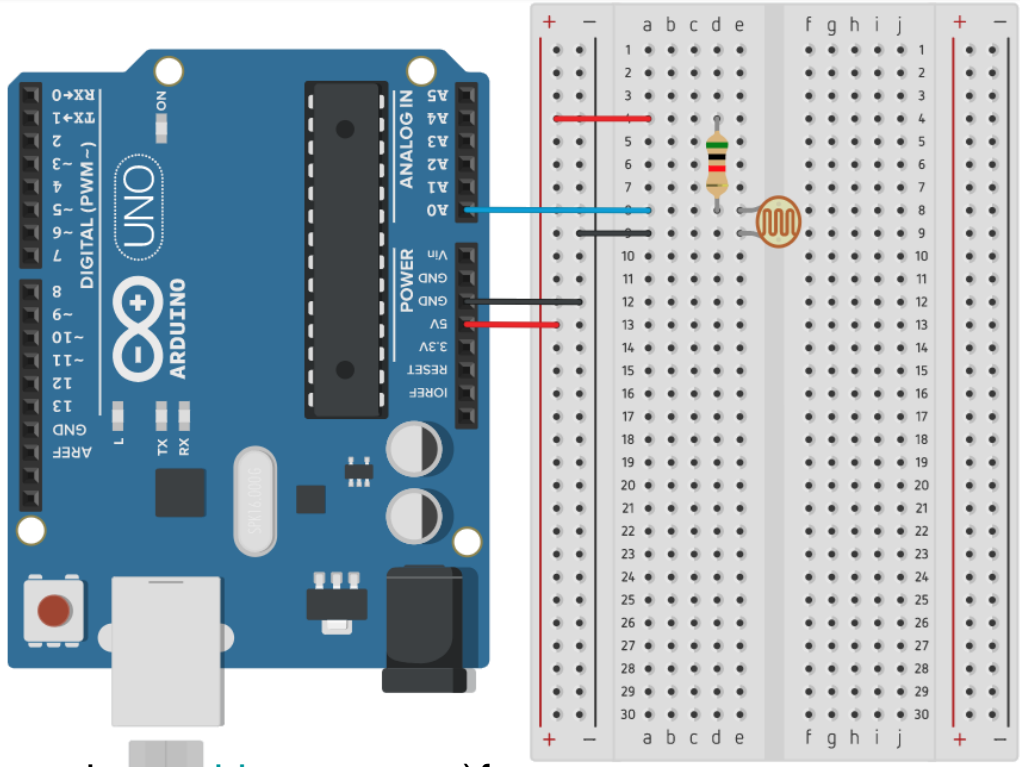
```
  analogWrite(Pino_Vel, velocidade);
  if(direcao == 0){
    digitalWrite(2,HIGH);
    digitalWrite(4,LOW);
  }
  if(direcao == 1){
    digitalWrite(2,LOW);
    digitalWrite(4,HIGH);
  }
  Serial.print(" vel=");
  Serial.print(velocidade);
  Serial.print(" dir=");
  Serial.println(direcao);
  delay(30);
}
```

Sensor para seguidor de linha

- A figura ao lado é o módulo sensor reflexivo infravermelho. Este é o sensor principal para desenvolver robôs seguidores de linha!
- O sensor conta com um transmissor e um receptor de infravermelho. O mesmo deve estar posicionado próximo ao chão e deve-se evitar que haja luz externa sendo captada pelo receptor.
- Assim como a grande maioria dos módulos, sensores ou atuadores que iremos estudar na mecatrônica, este sensor possui uma entrada de terra (gnd) e uma de alimentação +Vcc, nesse caso 5V. O pino denominado D2 liga ou desliga o transmissor. A quantidade de luz captada pelo receptor é traduzida em voltagem de 0 a 5V, ou seja, deve ser colocado em um ADC do Arduino.

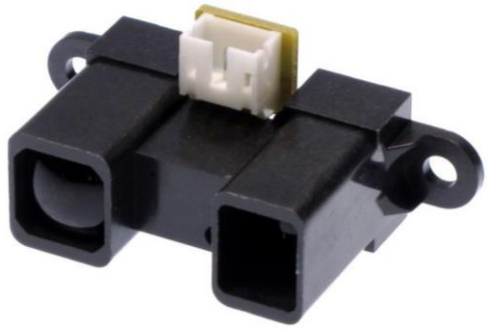
Simulando o sensor de linha com photoresistor R=5KΩ

```
int sensorPin = A0;
int sensorValue = 0;
double Voltagem = 0.0;
 void setup() {
 Serial.begin(9600);
}
void loop() {
  sensorValue = analogRead(sensorPin);
  Voltagem = interpolacao((double)sensorValue,0.0,1023.0,0.0,5.0);
 Serial.print("Nivel Logico ADC: ");
 Serial.print(sensorValue);
 Serial.print("  Voltagem: ");
 Serial.print(Voltagem);
 Serial.println("V");
}
double interpolacao(double valor, double old_min, double old_max, double new_min, double new_max){
  double derivada;
  double delta_old;
  double valor_new;
  derivada = (new_max-new_min)/(old_max-old_min);
  delta_old = valor - old_min;
  valor_new = new_min + derivada*delta_old;
  return(valor_new);
}
```



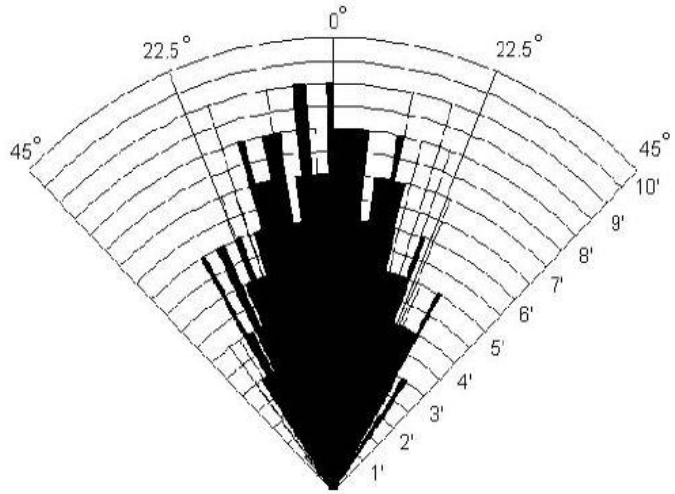
Sensor para seguidor de linha

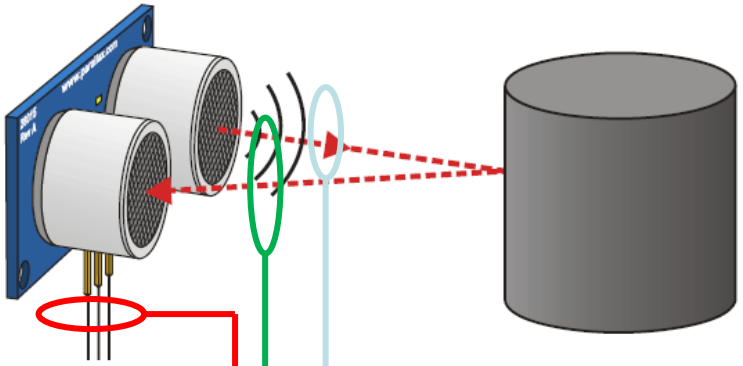
- Se estivéssemos presentes, eu pediria para vocês:
  - Verificarem a leitura dos sensores em superfícies de cores diferentes;
  - Verificarem a leitura do sensor para distancias da superfície (não ultrapassando 20cm);
  - Muitos sensores utilizam um encapsulamento que isola o emissor do receptor. Simule um encapsulamento com fita isolante e repita os procedimentos anteriores.

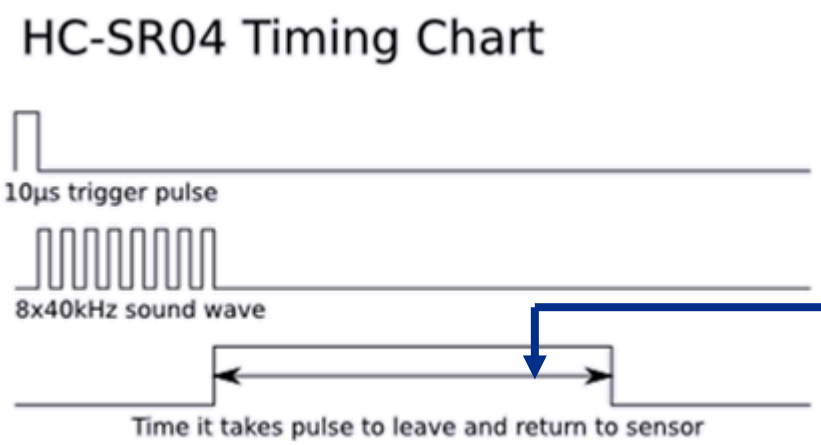


Sensor de distância ultrassônico

- Sensor para medir distância pelo uso de ultrassom;
- Medidas entre 2cm e 400cm com precisão de 0,3cm;
- A leitura deve ser a cada 50ms ou mais (valor a ser programado no delay).





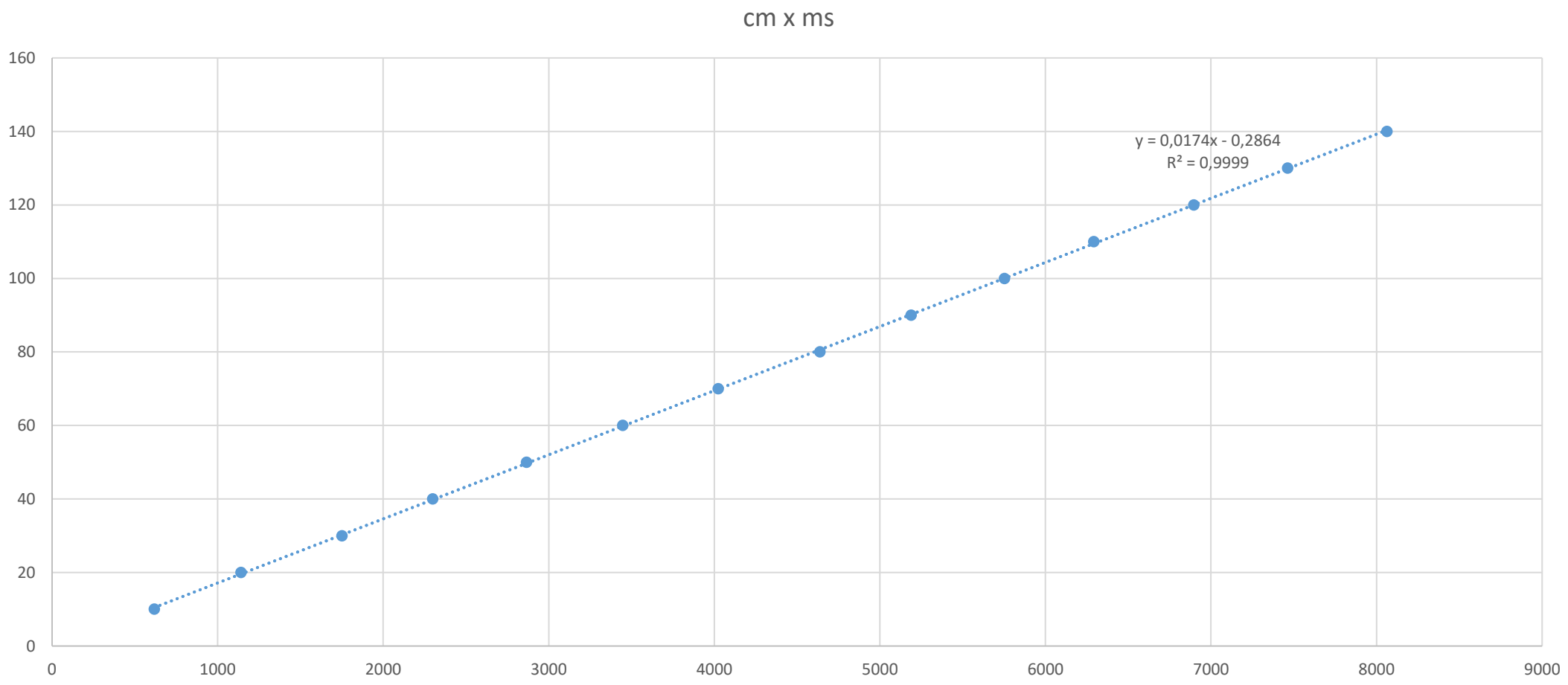


Medir com a função `pulseIn()`

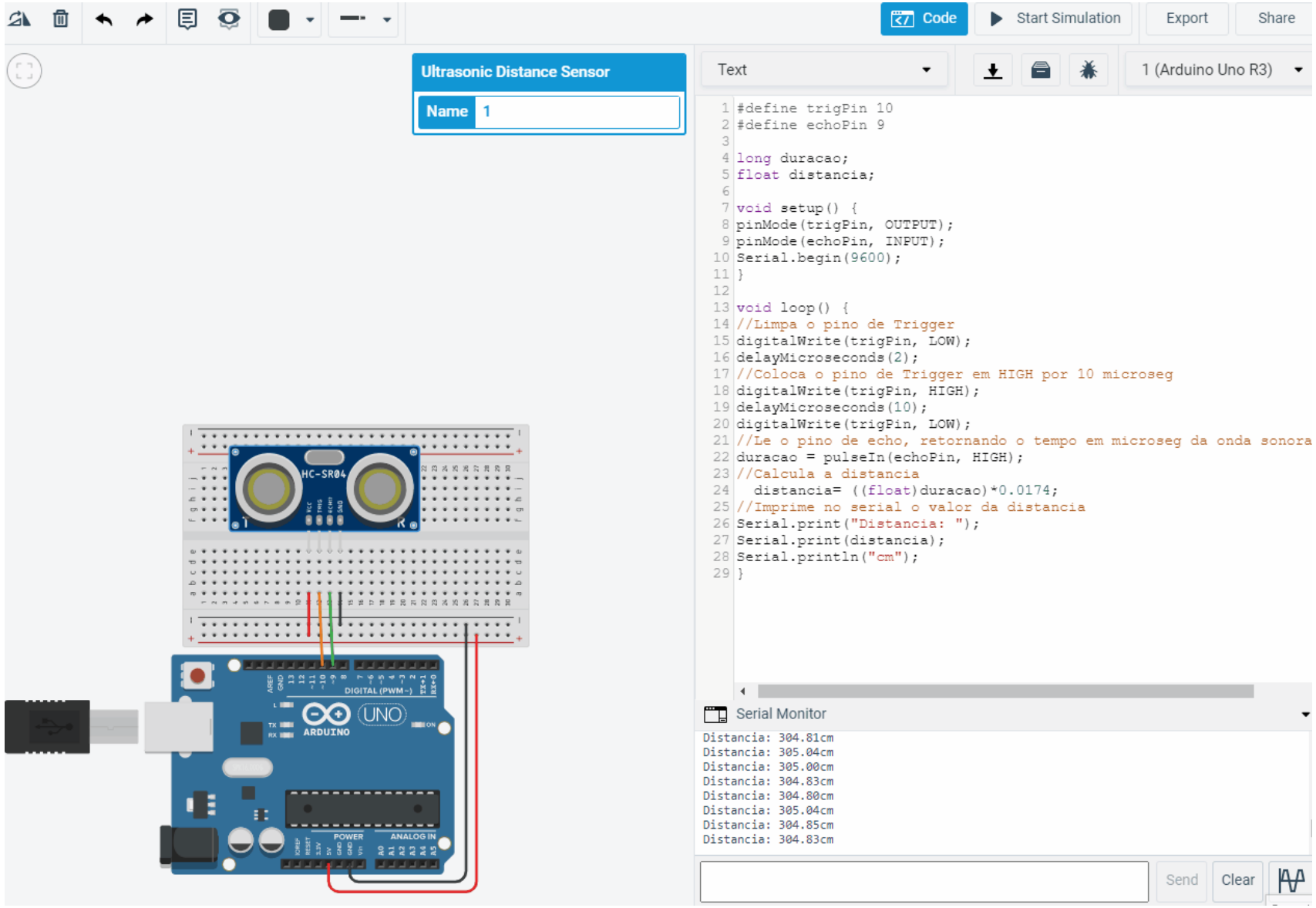
Sensor de distância ultrassônico

ms	cm
617	10
1141	20
1750	30
2299	40
2866	50
3446	60
4024	70
4637	80
5188	90
5753	100
6292	110
6896	120
7462	130
8063	140

Distância (cm) = 0,0174 * Tempo (ms)



Sensor para seguidor de linha - circuito



Sensor de distância – código parte 1

- A primeira parte do código contém:
  - define
  - declarações de variáveis globais
  - função `setup()` com:
  1. inicialização da comunicação serial em 9600kbps;
  2. definição do pino de trigger como saída;
  3. definição do pino de echo como entrada.

```
#define trigPin 10
#define echoPin 9

long duracao;
float distancia;

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
```

Sensor de distância – código parte 2

- Na função `loop()` temos:
  - Limpamos o pino de trigger colocando em LOW por 2 microssegundos;
  - Com o pino limpo, colocamos o pino de trigger em HIGH por 10 microssegundos. Neste momento são enviados 8 pulsos ultrassônicos;
  - Assim que os pulsos são enviados, o pino echo fica em HIGH até o momento quem que recebe um dos pulsos refletidos, indo para LOW. Medimos esse tempo com a função `pulseIN()`.

```
void loop() {
//Limpa o pino de Trigger
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
//Coloca o pino de Trigger em HIGH
// por 10 microseg
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
//Le o pino de echo, retornando o tempo
//em microseg da onda sonora ir e vir
  duracao = pulseIn(echoPin, HIGH);
```

Sensor de distância – código parte 3

- Na função `loop()` temos ainda:
  - Cálculo da distância baseado na tebela anterior;
  - Envio da distância via comunicação serial;

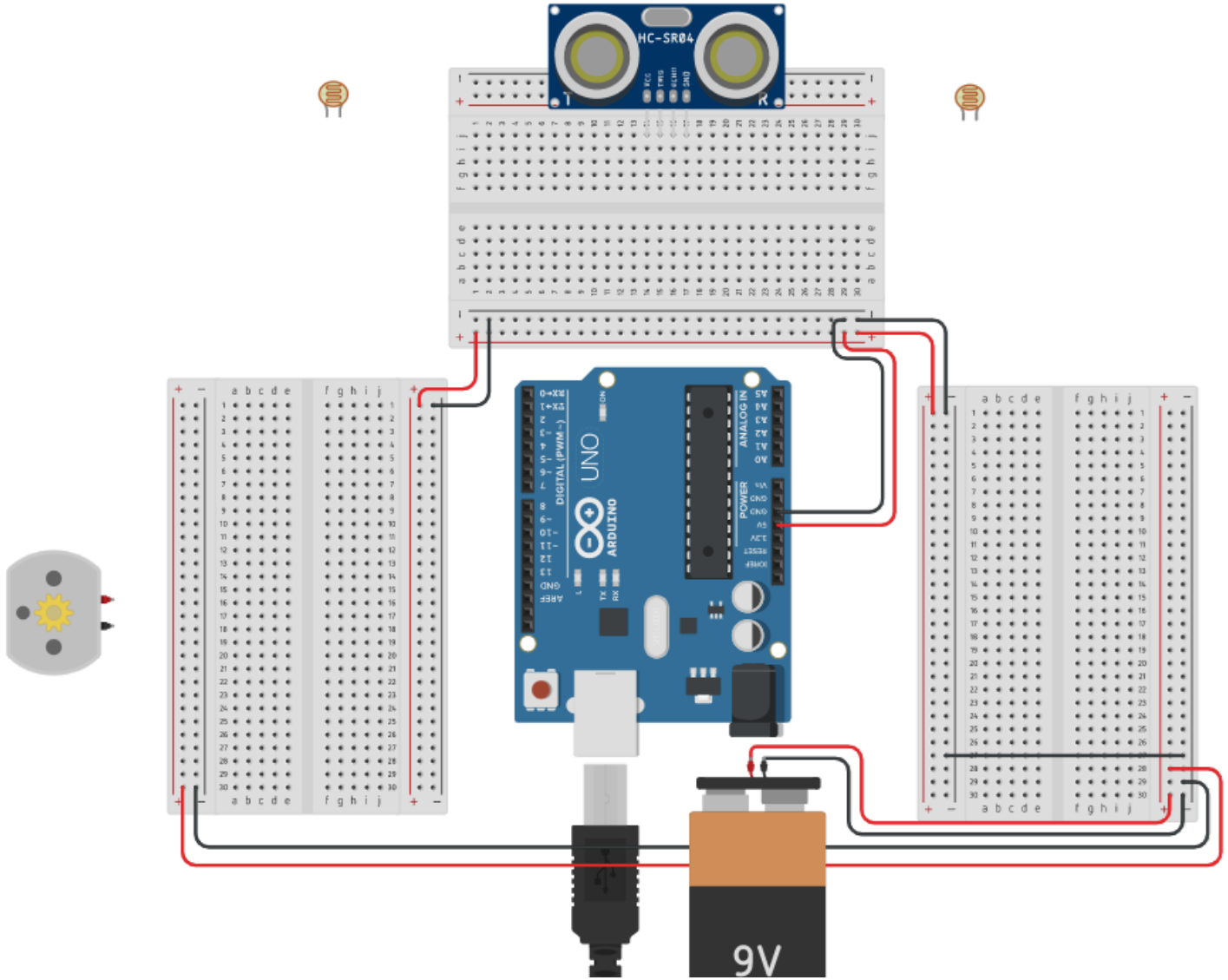
```
//Calcula a distancia
  distancia= ((float)duracao)*0.0174;
//Imprime no serial o valor da
//distancia
  Serial.print("Distancia: ");
  Serial.print(distancia);
  Serial.println("cm");
} //fim da função loop
```

Exercício para entrega

Implemente e complete o circuito ao lado.

No sensoriamento:

- Para os sensores de linha, não esqueça de colocar os resistores;
- O código deve imprimir na serial:
  - Valor do PWM do motor esquerdo;
  - Valor do PWM do motor direito;
  - Valor do sensor de linha esquerdo;
  - Valor do sensor de linha direito;
  - Valor da distância no sensor ultrassônico.



Exercício para entrega

Implemente e complete o circuito ao lado.

Na atuação:

- Os motores podem girar apenas em um sentido, mas você deve ser capaz de controlar a velocidade de ambos independentemente, através dos seguintes comandos:
  - `r` acelera motor direito,
  - `f` desacelera motor direito,
  - `t` acelera motor esquerdo,
  - `g` desacelera motor esquerdo.