

Robótica III

Professores

Antonio Fernando Traina – Professor da FATEC – Franca
Doutor em Física Aplicada Computacional - IFSC-USP,
aftraina@gmail.com



Roseli Aparecida Romero – Coordenadora do Curso
Professora ICMC-USP,
rafrance@icmc.usp.br



Março -2019

Agenda do curso

Introdução

- Conceitos Iniciais
- A Olimpíada Brasileira de Robotica - OBR
- Conceitos de Arduino
- Conceitos de Sensores e atuadores
- Apresentação do Kit básico
- Plataforma e simulação

Programação para arduino

- Apresentação das Regras da Olimpíada Brasileira de Robotica
- Estrutura da Linguagem do Arduino - linguagem C
- As portas de E/S do Arduino e suas funções em C
- Programando os sensores e atuadores

Desenvolvendo Programação robô seguidor de linha

- Motores
- Sensores claro/escuro
- Sensores de cor
- Sensores de distância
- Aprimoramentos e melhorias

Competição entre equipes – Organizar.

Data		Tópicos	
16/03	Introdução	Conceitos Iniciais A Olimpíada Brasileira de Robótica - OBR Conceitos de Arduino Conceitos de Sensores e atuadores Apresentação do Kit básico Plataforma e simulação	Teórica
23/03	Programação para Arduino	Estrutura da Linguagem do Arduino (linguagem C) As portas de E/S do Arduino e suas funções em C Programando os sensores e atuadores no simulador tinkercad	Prática
30/03	Desenvolvendo Programação robô seguidor de linha	Ligando os componentes fisicamente: Motores Ponte H Seguidor de linha	Prática
06/04	Desenvolvendo Programação robô seguidor de linha	Ligando os componentes fisicamente: Sensores de distância – Sonar Micro Servo Motor Sensores de cor	Prática
13/04	Desenvolvendo Programação robô seguidor de linha	Apresentação das Regras da Olimpíada Brasileira de Robótica Aprimoramentos e melhoria	Prática



Competição entre as equipes formadas

27/04

Robótica III

Aula IV

abril - 06/2019



**Montagem de um exemplo
de carrinho segue linha**

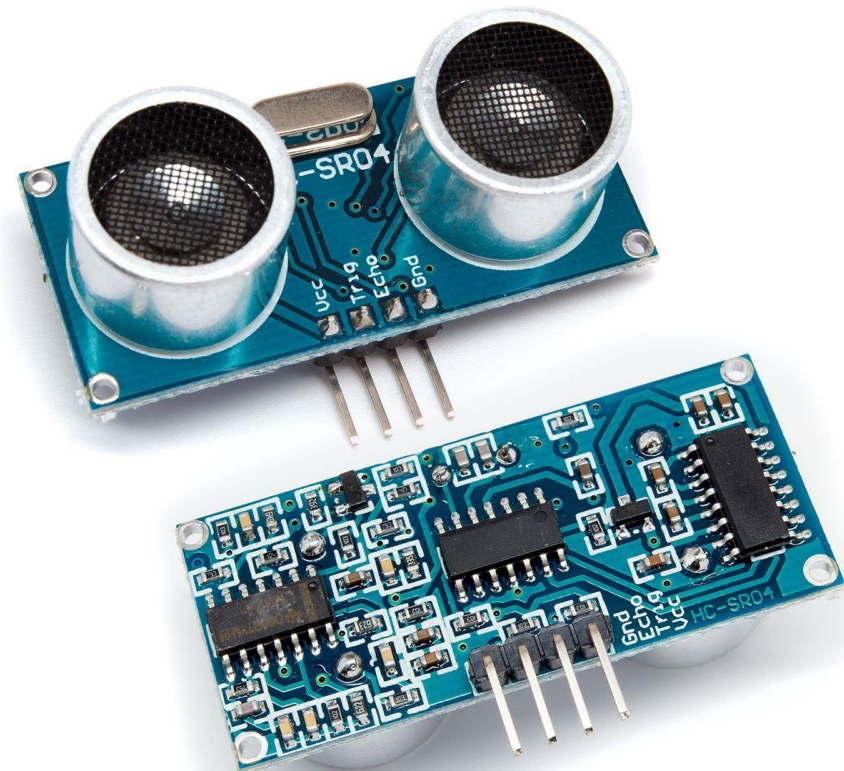
Sensores de distância

Sonar

Micro Servo Motor

Sensor Ultrassônico HC-SR04

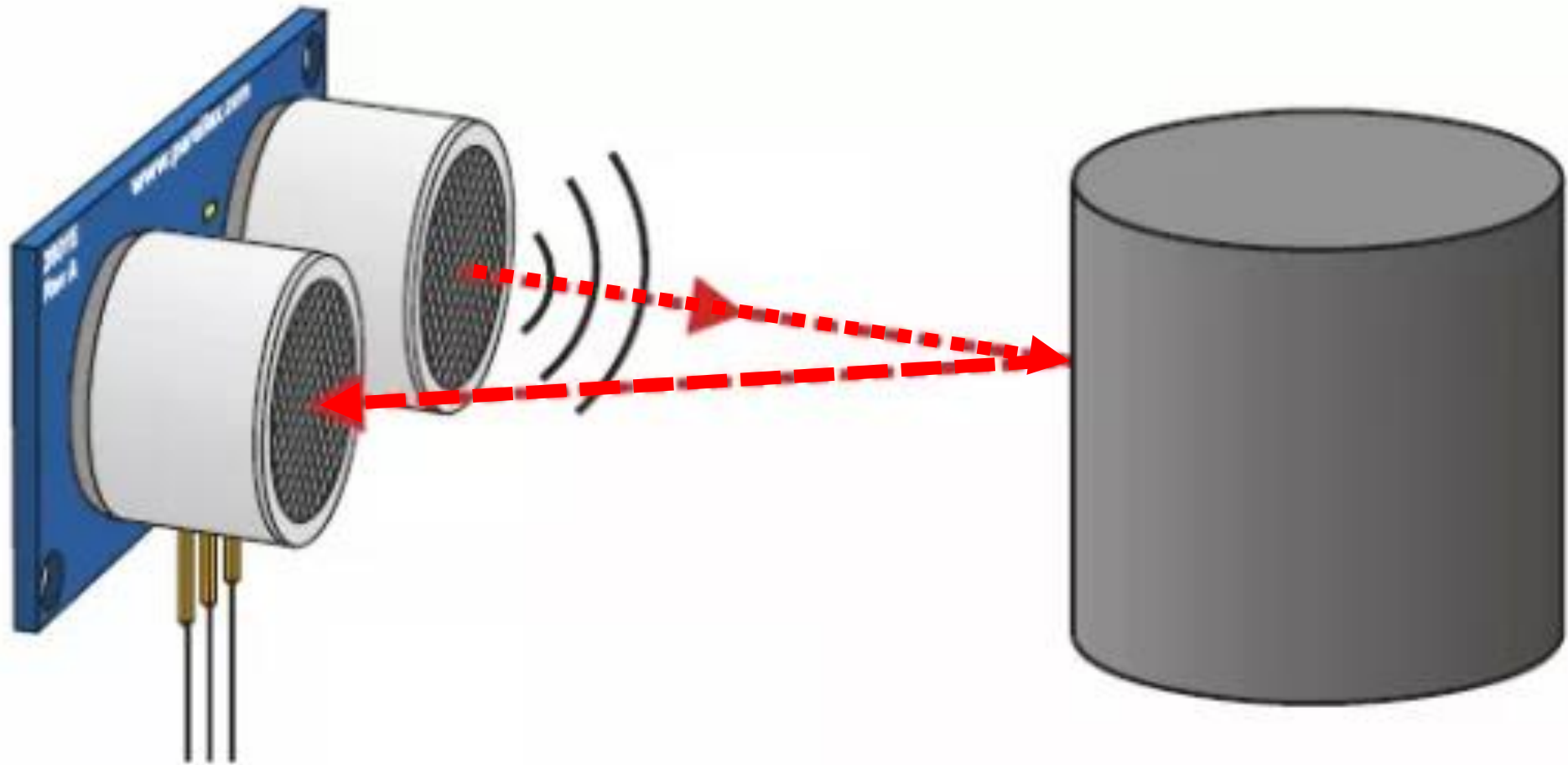
Ele é capaz de medir distâncias de 2cm a 4m com ótima precisão. Este módulo possui um circuito pronto com emissor e receptor acoplados



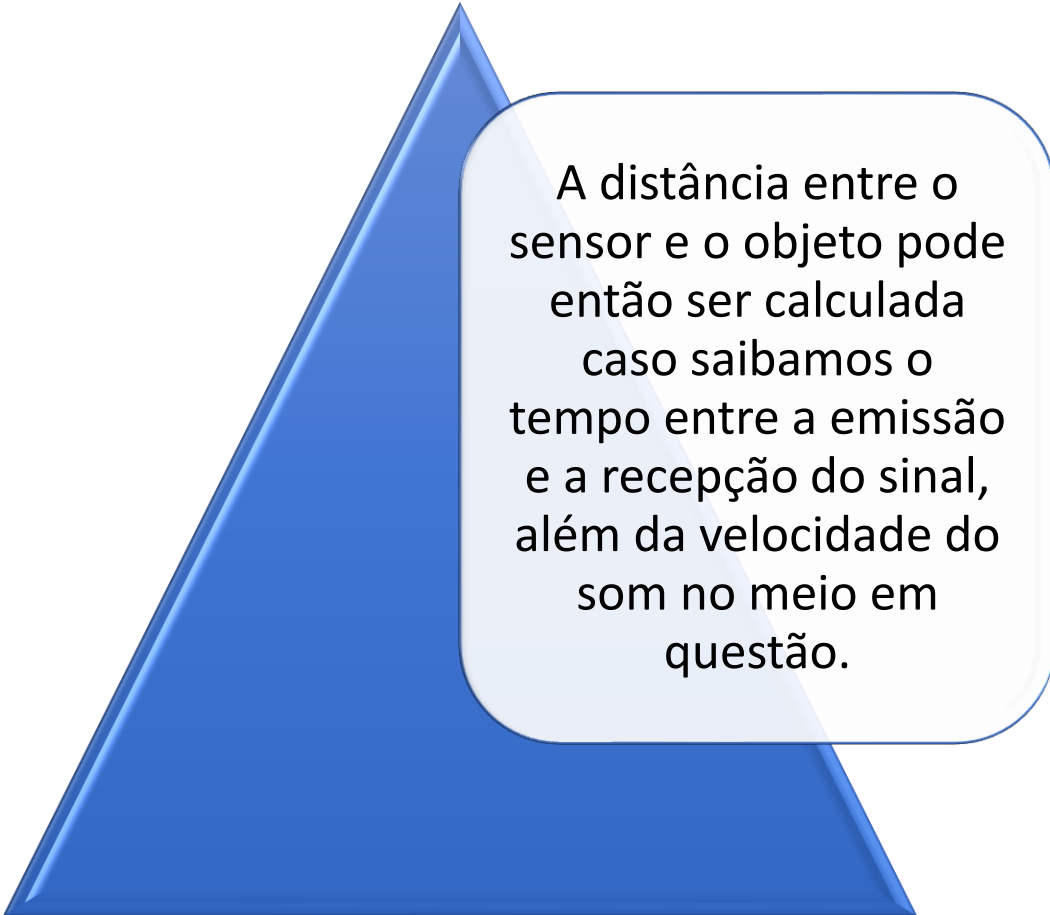
Funcionamento

Tudo começa pela emissão de um pequeno pulso sonoro de alta frequência que se propagará na velocidade do som no meio em questão.

Quando este pulso atingir um objeto, um sinal de eco será refletido para o sensor.



Cálculo da distância



A distância entre o sensor e o objeto pode então ser calculada caso saibamos o tempo entre a emissão e a recepção do sinal, além da velocidade do som no meio em questão.

$$V_{som} = 340,29 \frac{m}{s}$$

$$V_{som} = \frac{\Delta S}{\Delta T} = 340,29 \frac{m}{s}$$

$$\Delta T = T/2$$

$$\frac{\Delta S}{T/2} = 340,29$$

$$\Delta S = \frac{340,29 * T}{2}$$

Montando o exemplo

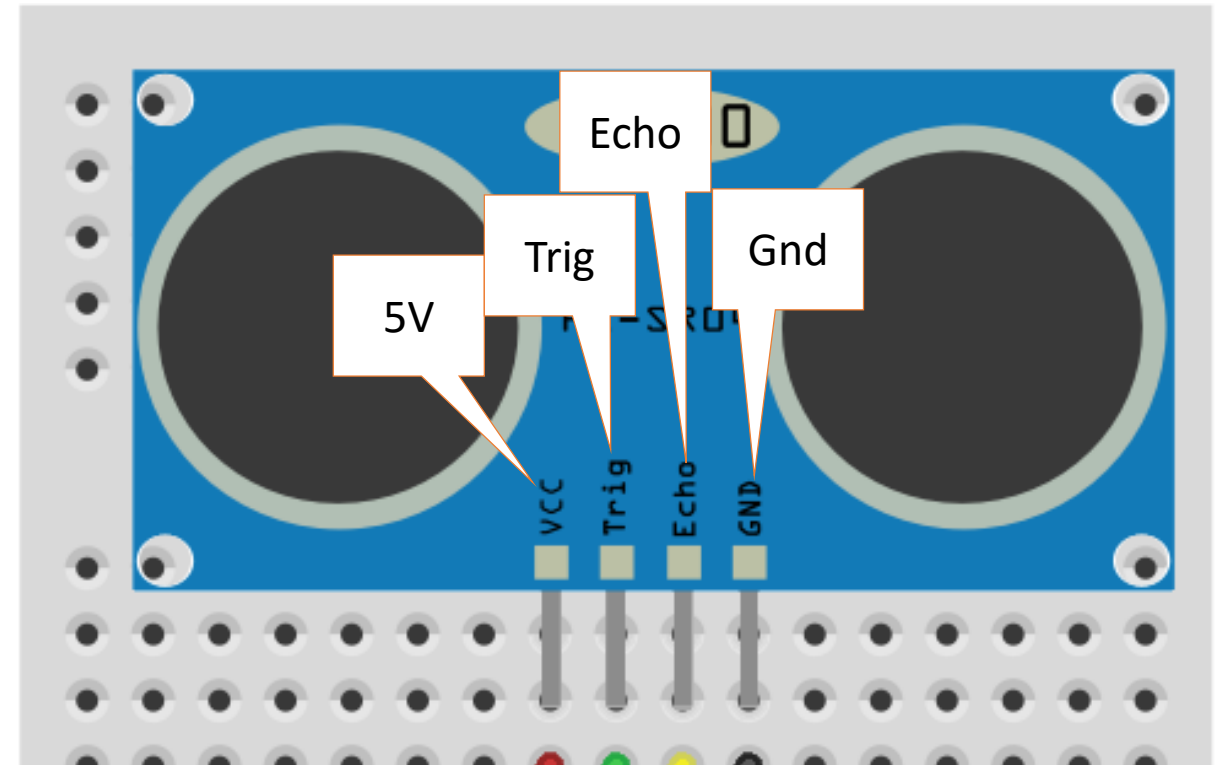
O HC SR04 possui 4 pinos sendo eles:

Vcc – Deve ser conectado a um pino 5V do Arduino.

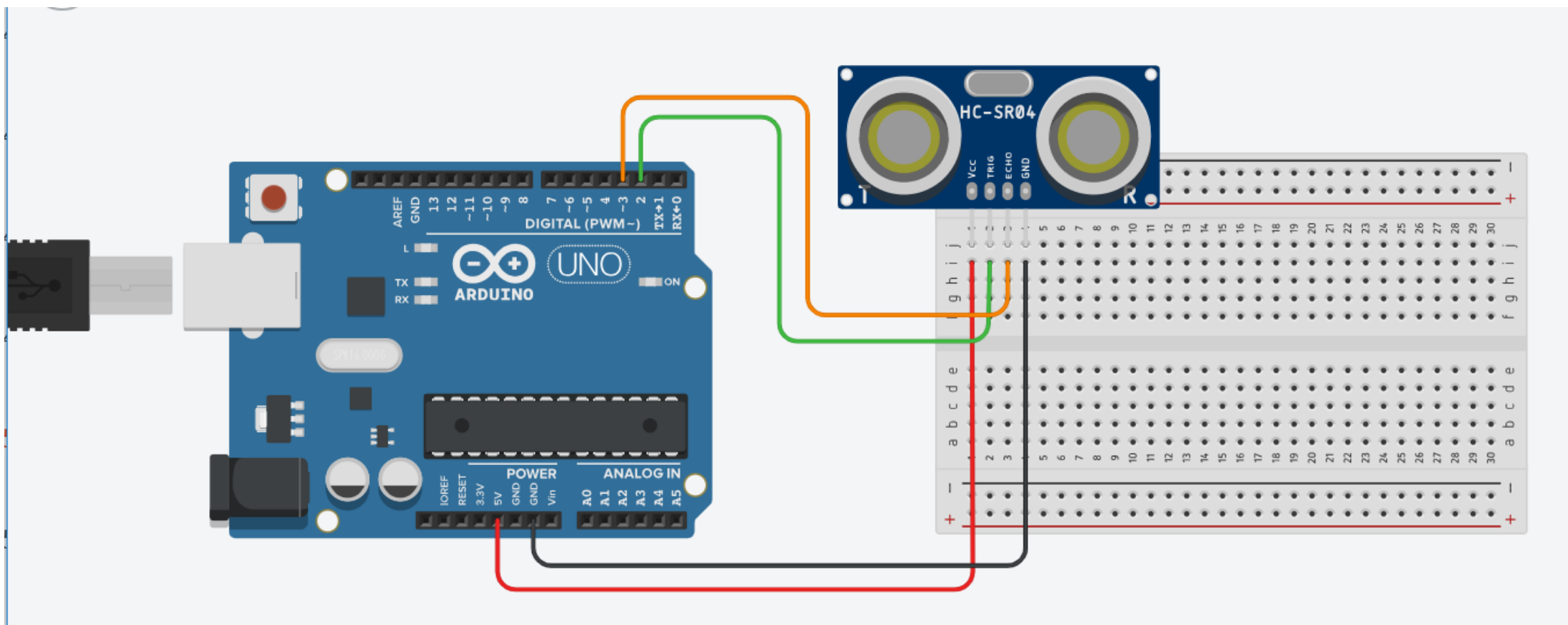
Trig – Deve ser conectado a um pino digital configurado como saída. Utilizaremos o pino 8.

Echo – Deve ser conectado a um pino digital configurado como entrada. Utilizaremos o pino 7.

Gnd – Deve ser conectado a um pino GND do Arduino.



Ultrassom – no tinkercad ou no “real”



Código teste 1

Definições de variáveis:

```
//Programa : Controle Sensor ultrassom HC-SR04 - Versão 1.0
//Detecta distância
//Definicoes pinos Arduino ligados ao sensor
const int trigPin = 2;
const int echoPin = 3;
long duracao;
int distancia;
```

Código setup()

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

Código loop()

```
void loop() {  
    digitalWrite(trigPin, LOW); //desliga o ultrassom  
    delayMicroseconds(2);  
    digitalWrite(trigPin, HIGH); //dispara o ultrassom  
    delayMicroseconds(10);  
    digitalWrite(trigPin, LOW); //desliga o ultrassom  
    duracao = pulseIn(echoPin, HIGH); //aguarda o retorno  
    distancia= duracao*0.034/2; //calcula a distância  
    Serial.print("Distancia: ");  
    Serial.println(distancia);  
}
```


Incluindo Bibliotecas

Uma grande vantagem das placas Arduino é a grande diversidade de bibliotecas disponíveis que podem ser usadas em seu programa.

Isso faz com o trabalho pesado em programação seja abstraído e resumido em simples comandos.

Com isso, o desenvolvedor não precisa de conhecimento muito aprofundado em programação, podendo gastar menos tempo nisso, resultando em mais tempo para trabalhar com empenho na estratégia de controle.

A seguir, iremos aprender como adicionar uma biblioteca em sua IDE. Esse mesmo procedimento poderá usado para outros sensores.

Usando uma biblioteca

Nesse exemplo utilizaremos a biblioteca Ultrasonic.h.

A utilização dessa biblioteca é bastante simples.

Apenas devemos instalar a biblioteca pelo IDE do Arduino como visto a seguir

Como instalar uma biblioteca no arduino

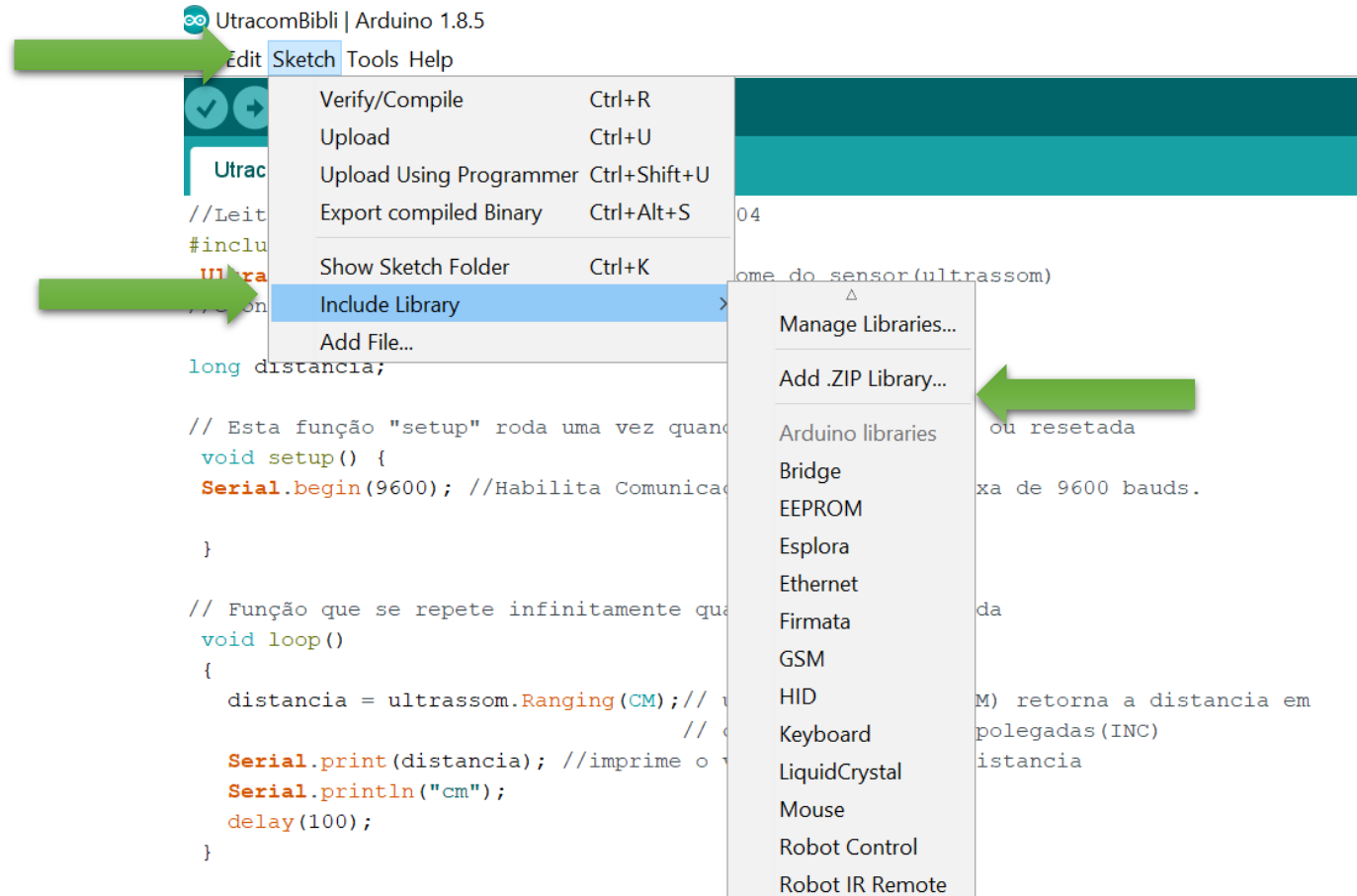
Primeiramente, faça o download dos arquivos da biblioteca compactados no formato zip.

Em seguida, basta abrir o IDE e ir em “Sketch -> Incluir Biblioteca -> Adicionar biblioteca .ZIP”, como mostrado a seguir

<https://www.arduinelibraries.info/libraries/>

<https://portal.vidadesilicio.com.br/wp-content/uploads/2017/05/Ultrasonic.zip>

Instalando a biblioteca pelo IDE do Arduino



Código teste 2

Com biblioteca

Definições de variáveis

```
//Programa : Controle Sensor HC-SR04 - Versão 2.0
//Detecta distância
#include <Ultrasonic.h>
// define o nome do sensor como trig(2) e o
//echo(3) respectivamente
Ultrasonic ultrasonic(
long distancia;
```

Cuidado, em
linguagem C o
maiusculo e o
minúsculo são
diferentes!!!

Código setup()

```
// Esta função "setup" roda uma vez quando a placa é ligada ou
//resetada

void setup() {
//Habilita Comunicação Serial a uma taxa de 9600 bauds.
    Serial.begin(9600);
}
```

Código loop()

```
// Função que se repete infinitamente quando a placa é ligada
void loop()
{
// a função ultrassom.Ranging(CM) retorna a distancia em
// centímetros(CM) ou polegadas(INC)
    distancia = ultrassom.Ranging(CM);
//imprime o valor da variável distancia
    Serial.print(distancia);
    Serial.println("cm");
    delay(100);
}
```


Atuador: Micro Servo Motor Tower Pro Sg90

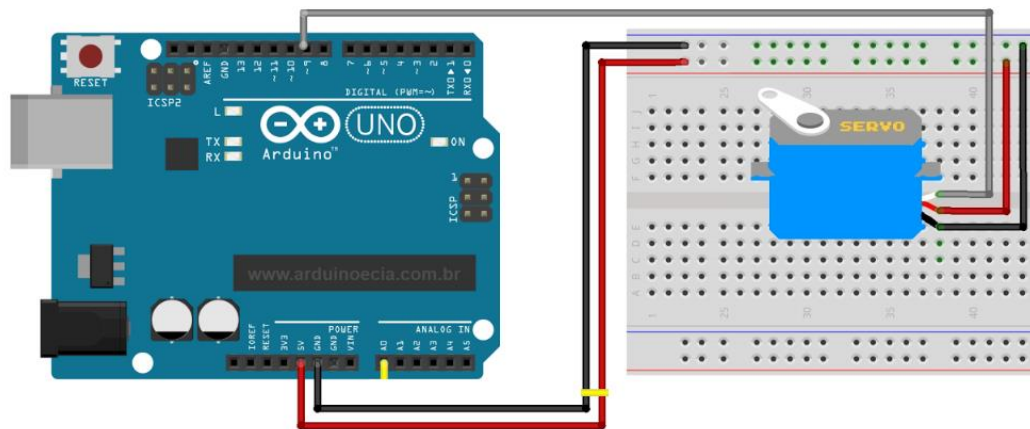
Sg90

Especificações:

Voltagem de Operação: 3,0 – 7,2v
Velocidade: 0,12 seg/60Graus (4,8v) sem carga
Torque: 1,2 kg.cm (4,8v) e 1,6 kg.cm (6,0v)
Temperatura de Operação.: -30C ~ +60C
Dimensões.: 32x30x12 mm
Tipo de Engrenagem: Nylon
Tamanho cabo: 245 mm
Peso: 9g



Ligação: Conecte a alimentação do Micro Servo 9g ao Arduino.



Fio Marrom com
GND,

Fio Vermelho com
5v e


Fio Laranja na Porta
Digital 6.

Código teste

Com biblioteca

Definições de variáveis:

```
//Programa : Controle Sensor ultrassom HC-SR04 - Versão 2.0
include <Servo.h>
#define SERVO 6           // Porta Digital 6 PWM
Servo s;                 // Variável Servo
int pos;                 // Posição Servo
```



Código setup()

```
void setup ()  
{  
    s.attach(SERVO);  
    Serial.begin(9600);  
    s.write(0);           // Inicia motor posição zero  
}
```

Código loop()

```
void loop() {  
  for(pos = 0; pos < 90; pos++)  
  {  
    s.write(pos);  
    delay(15);  
  }  
  delay(1000);  
  for(pos = 90; pos >= 0; pos--)  
  {  
    s.write(pos);  
    delay(15);  
  }  
}
```

Sensor de Cor

TCS230

Sensor de cor TCS230

Baseia-se em uma matriz 8×8 (64) de fotodiodos combinados com um conversor corrente-frequência.

Desses 64 possuem filtros :

16 para a luz verde,

16 para a vermelha,

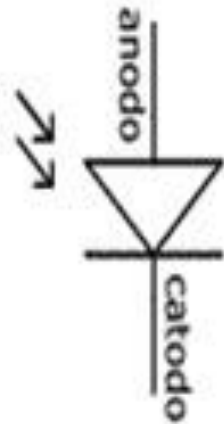
16 para a luz azul e

16 não possuem filtro algum.

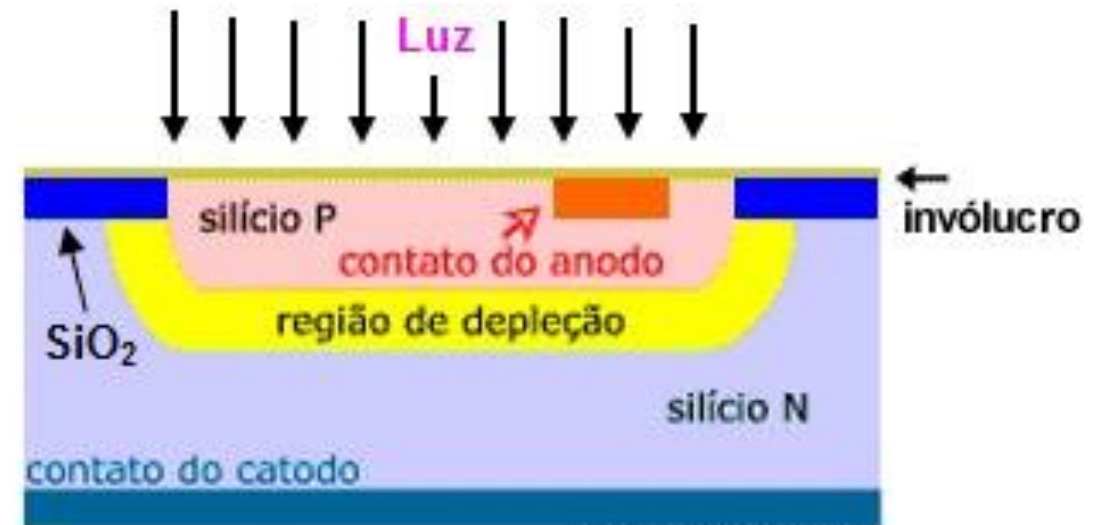


Sensor de cor TCS230

Sob esse conjunto de fotodiodos incide a luz refletida pelos objetos posicionados ao alcance do sensor.



Símbolo Fotodiodo



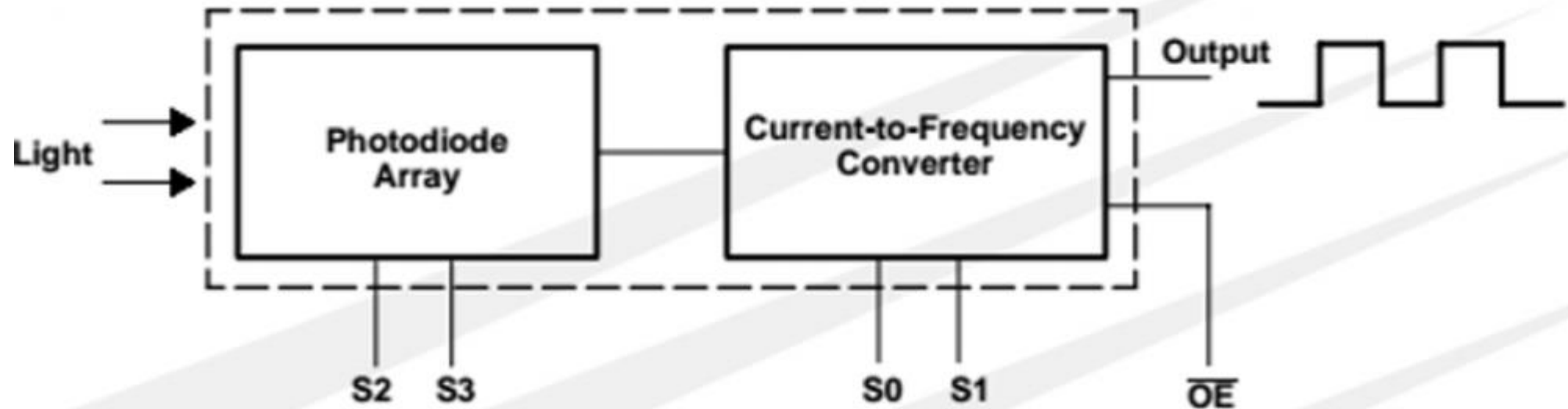
Fotodiodo

Esquema construtivo em corte

Fig-12

Sensor de cor TCS230

Os fotodiodos geram uma corrente de saída de acordo com a intensidade da luz refletida e também de acordo com seus filtros.

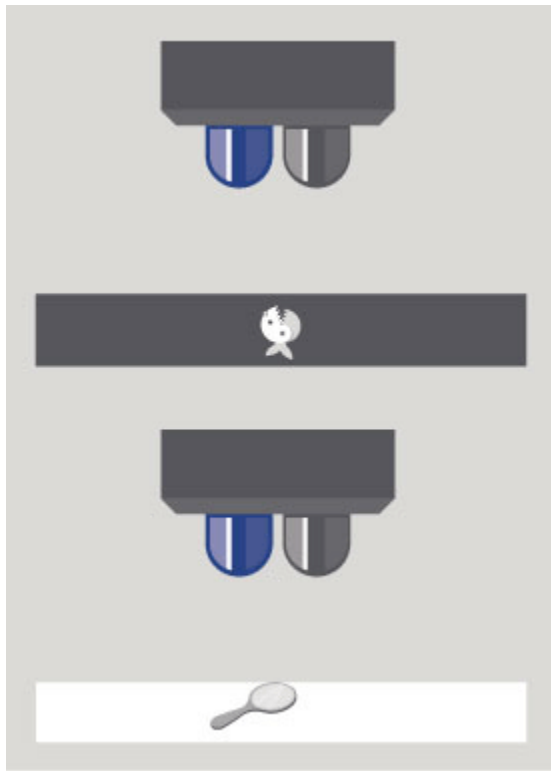


Sensor de cor TCS230

Essa corrente de saída é convertida em uma **onda quadrada de 50% de duty cycle**, isto é, uma onda em que metade do período está em nível alto e outra metade em nível baixo.



Sensor de cor TCS230



Absorve

Reflete

A frequência dessa onda quadrada é diretamente proporcional à intensidade da luz refletida percebida pelos conjuntos de fotodiodos.

Sensor de cor TCS230

Cada bloco de 16 fotodiodos pode ser selecionado pelos pinos S2 e S3.

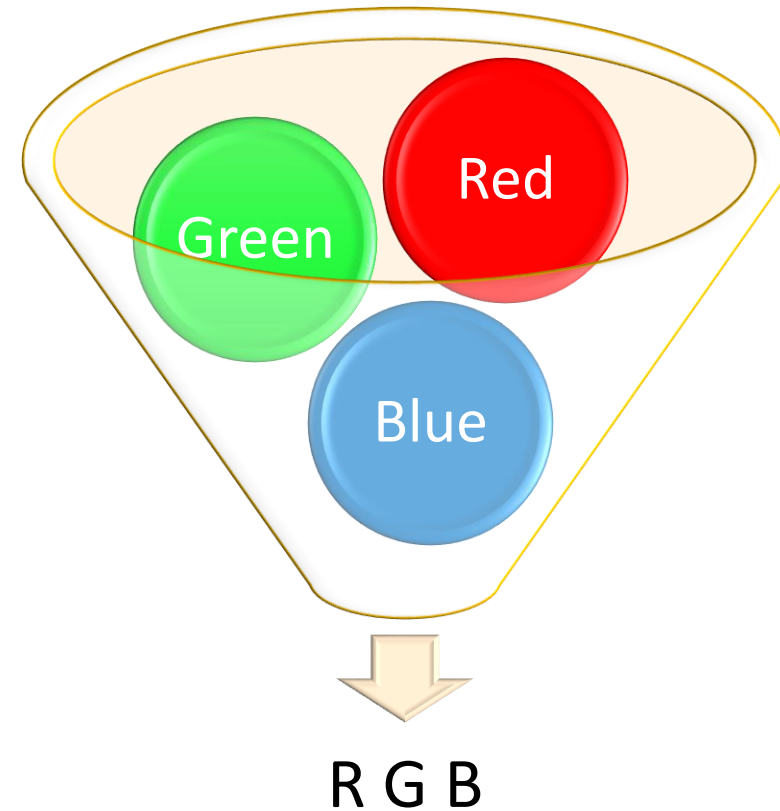
Os pinos S0 e S1 permitem regular a escala de frequência e também desligar a saída. Veja a tabela abaixo, também retirada do datasheet do sensor:

Pino		Escala da frequencia de saída
S0	S1	
Low	Low	Desligado
Low	High	2%
High	Low	20%
High	High	100%

Pino		Fotodiodo
S2	S3	
Low	Low	Vermelho / Red
Low	High	Azul / Blue
High	Low	Sem filtro
High	High	Verde / Green

Sensor de cor TCS230

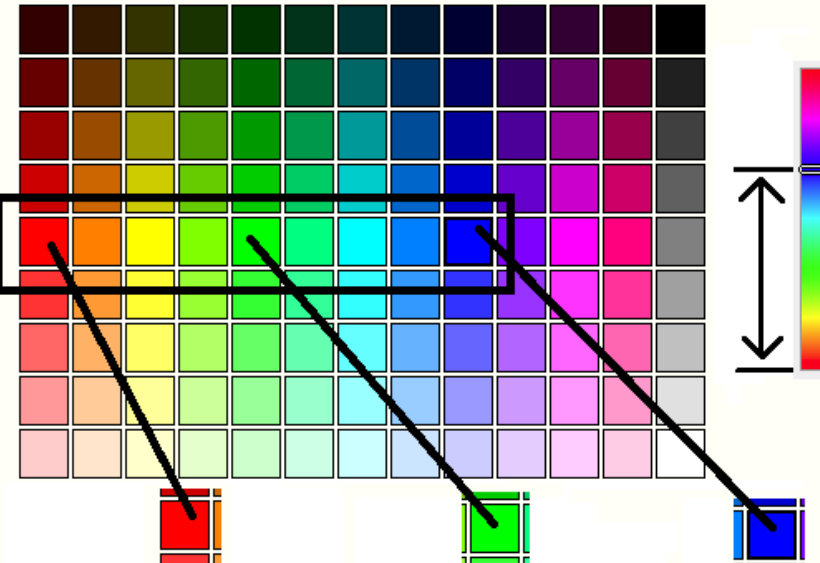
Por meio da matriz de fotodiodos e os seus respectivos filtros, pode-se identificar as cores primárias do padrão RGB (red, green, blue) lendo a frequência de saída para cada filtro de cor diferente e obter uma aproximação do código RGB do objeto



Sensor de cor TCS230

RGB color codes chart

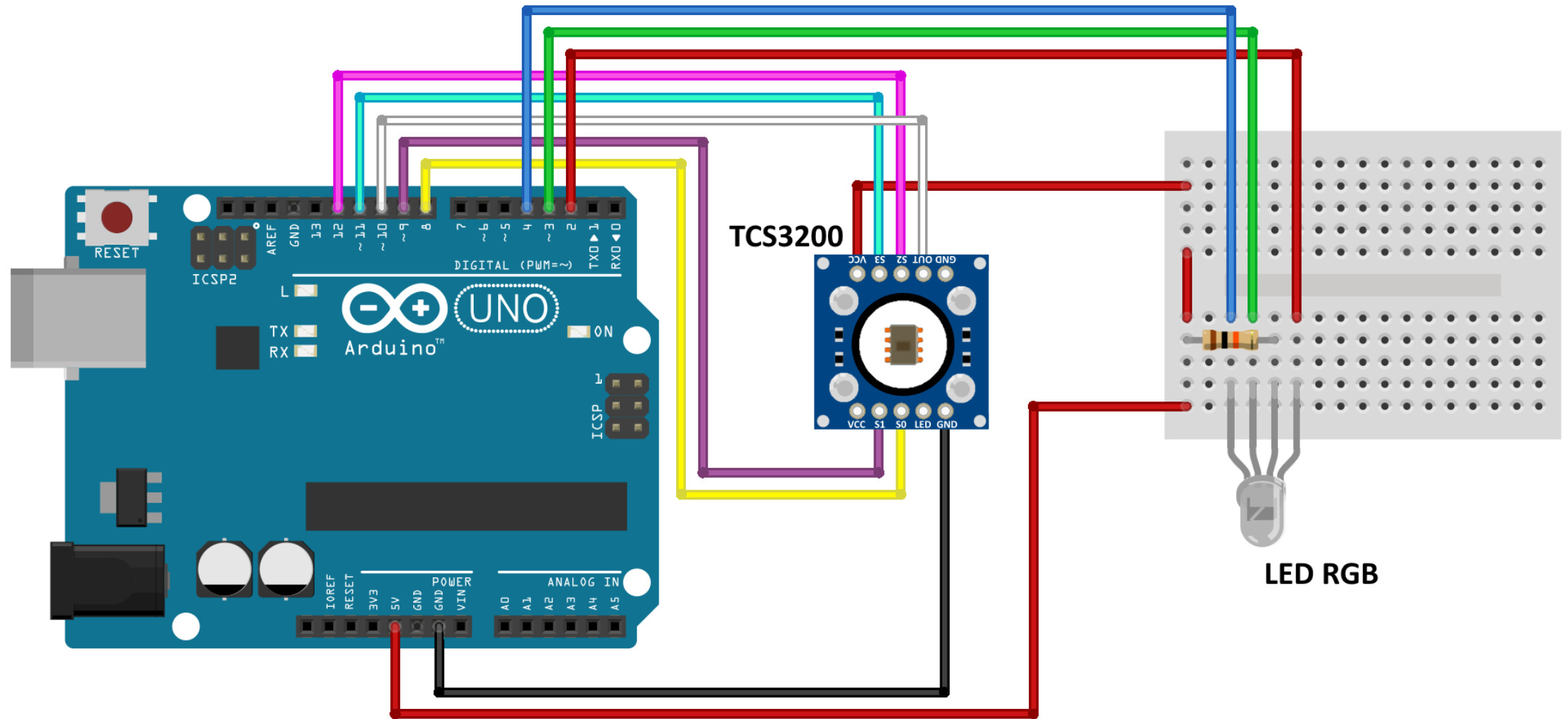
Hover with cursor on color to get the hex and decimal color codes below:



Hex: #	FF0000	Hex: #	00FF00	Hex: #	0000FF
Red:	255	Red:	0	Red:	0
Green:	0	Green:	255	Green:	0
Blue:	0	Blue:	0	Blue:	255



Um modelo para testar:



Código teste 1

Sensor de cor

Definições de variáveis:*

```
//Programa : Controle Sensor TCS3200 e led RGB - Versão 1.0
```

```
//Pinos de conexao do modulo
```

```
const int s0 = 8;  
const int s1 = 9;  
const int s2 = 12;  
const int s3 = 11;  
const int out = 10;
```

```
//Pinos do led RGB
```

```
int pinoledverm = 2;  
int pinoledverd = 3;  
int pinoledazul = 4;  
//Variaveis cores  
int red = 0;  
int green = 0;  
int blue = 0;
```

Código setup()

```
void setup() {  
    pinMode(s0, OUTPUT);  
    pinMode(s1, OUTPUT);  
    pinMode(s2, OUTPUT);  
    pinMode(s3, OUTPUT);  
    pinMode(out, INPUT);  
    pinMode(pinoledverm, OUTPUT);  
    pinMode(pinoledverd, OUTPUT);  
    pinMode(pinoledazul, OUTPUT);  
    Serial.begin(9600);  
    digitalWrite(s0, HIGH);  
    digitalWrite(s1, LOW);  
}
```

Código loop()

```
void loop() {  
  //Detecta a cor  
  color();  
  
  //Mostra no monitor serial  
  Serial.print("Vermelho :");  
  Serial.print(red, DEC);  
  Serial.print(" Verde : ");  
  Serial.print(green, DEC);  
  Serial.print(" Azul : ");  
  Serial.print(blue, DEC);  
  Serial.println();  
}
```

```
//Verifica se a cor vermelha foi detectada  
if (red < blue && red < green && red < 100) {  
  Serial.println("Vermelho");  
  digitalWrite(pinoledverm, LOW); //Acende o led  
vermelho  
  digitalWrite(pinoledverd, HIGH);  
  digitalWrite(pinoledazul, HIGH);  
}  
  
//Verifica se a cor azul foi detectada  
else if (blue < red && blue < green && blue < 1000) {  
  Serial.println("Azul");  
  digitalWrite(pinoledverm, HIGH);  
  digitalWrite(pinoledverd, HIGH);  
  digitalWrite(pinoledazul, LOW); //Acende o led azul  
}
```

Código loop()

```
// continua void loop() {  
  //Verifica se a cor verde foi detectada  
  else if (green < red && green < blue) {  
    Serial.println("Verde");  
    digitalWrite(pinoledverm, HIGH);  
    digitalWrite(pinoledverd, LOW);  
  //Acende o led verde  
    digitalWrite(pinoledazul, HIGH);  
  }  
}
```

```
Serial.println();  
  //Delay para apagar os leds e reiniciar o processo  
  delay(50);  
  digitalWrite(pinoledverm, HIGH);  
  digitalWrite(pinoledverd, HIGH);  
  digitalWrite(pinoledazul, HIGH);  
}  
  // final de loop()
```

Código color()

```
void color() {  
    //Rotina que le o valor das cores  
    digitalWrite(s2, LOW);  
    digitalWrite(s3, LOW);  
    //count OUT, pRed, RED  
    red = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);  
    digitalWrite(s3, HIGH);  
    //count OUT, pBLUE, BLUE  
    blue = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);  
    digitalWrite(s2, HIGH);  
    //count OUT, pGreen, GREEN  
    green = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);  
}
```

Dicas



Fita dupla face

Dicas

Carrinho não roda – Este é um problema complexo, pois podem ser infinitas possibilidades.

Tente isolar os componentes e testar 1 por 1, principalmente os motores e a ponte H.

Em alguns casos pode ser problemas de aterramento da bateria também.



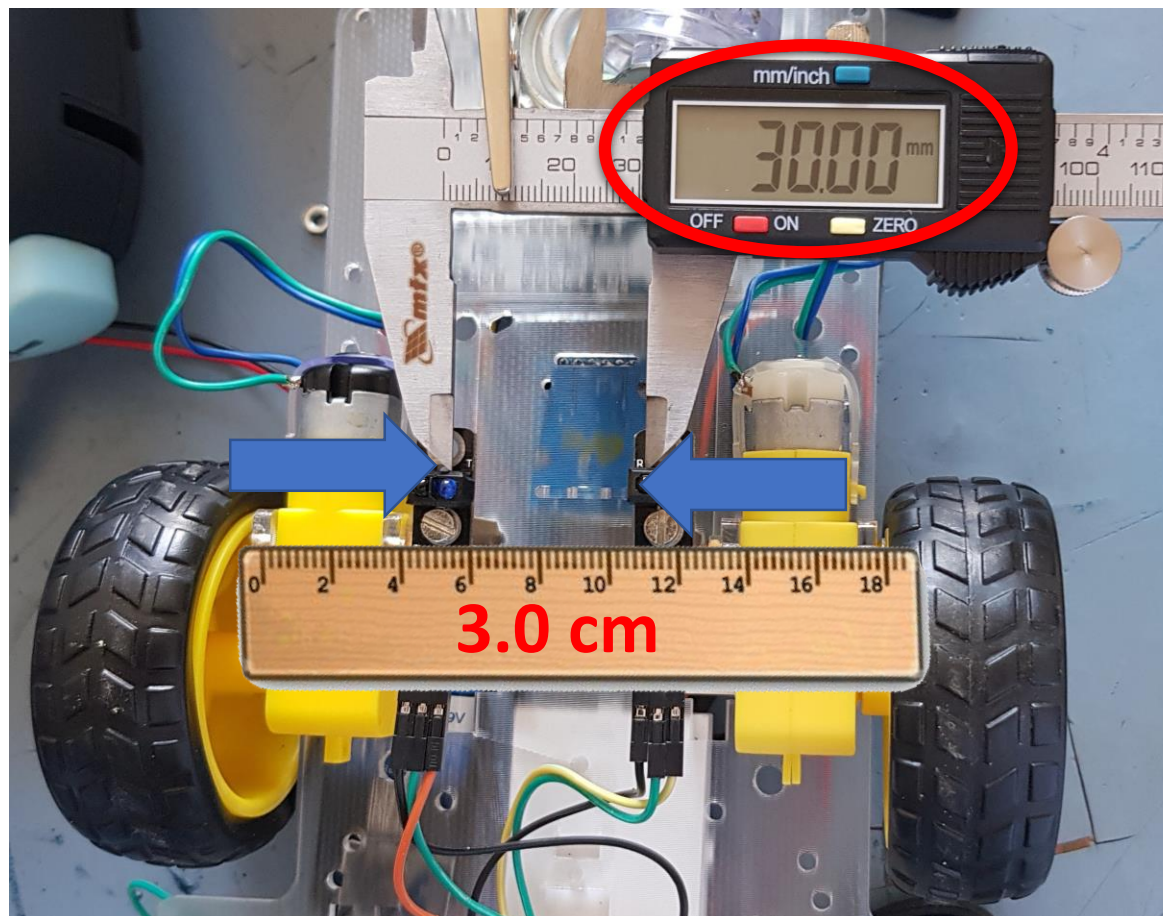
Dicas

Bateria fraca – Os testes podem ter consumido a bateria e talvez seja necessário o uso de uma nova. Baterias abaixo de 6,5 Volts já começam a diminuir a eficiência do carrinho.

Carrinho saindo da pista – Isso pode acontecer por ele estar rápido demais ou por falha do infravermelho.

Se o problema for com o contraste da pista (talvez parte dela esteja mais escura) use 2 LEDs de alto brilho na frente do carrinho para iluminar a pista próximo aos sensores.

Distância entre os sensores infravermelhos



Referencias

Motor DC com Driver Ponte H L298N - FelipeFlop

- <https://www.filipeflop.com/blog/motor-dc-arduino-ponte-h-l298n/>

Robô seguidor de linha com sensor infravermelho e PWM - Vida de silício)

- <https://portal.vidadesilicio.com.br/robo-seguidor-de-linha-sensor-infravermelho-e-pwm/>

Fim da Aula IV