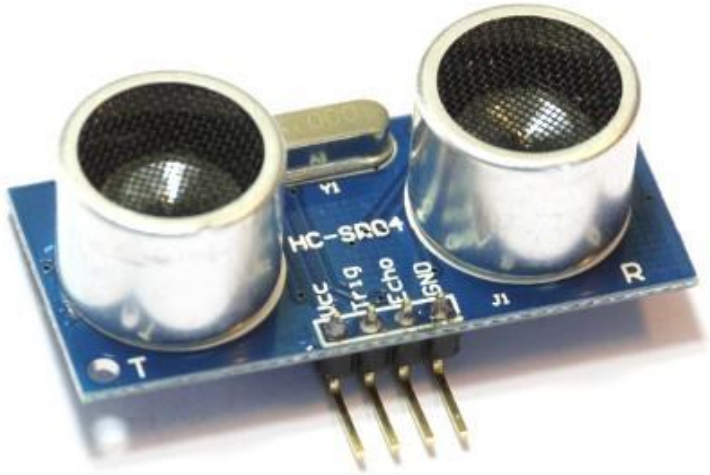
SENSOR ULTRASSOM HC-SR04

O sensor ultrassom é amplamente utilizado em aplicações onde se deseja medir distâncias ou evitar colisões, como na robótica móvel e de reabilitação. Nesse tutorial utilizaremos o sensor ultrassom HC-SR04 .



PRINCÍPIO DE FUNCIONAMENTO

Tudo começa pela emissão de um pequeno pulso sonoro de alta frequência que se propagará na velocidade do som do meio em questão. Quando este pulso atingir um objeto, um sinal de eco será refletido para o sensor. A distância entre o sensor e o objeto pode então ser calculada se soubermos o tempo entre a emissão e a recepção do sinal e a velocidade do som no meio em questão. Como mostra a FIGURA 1.

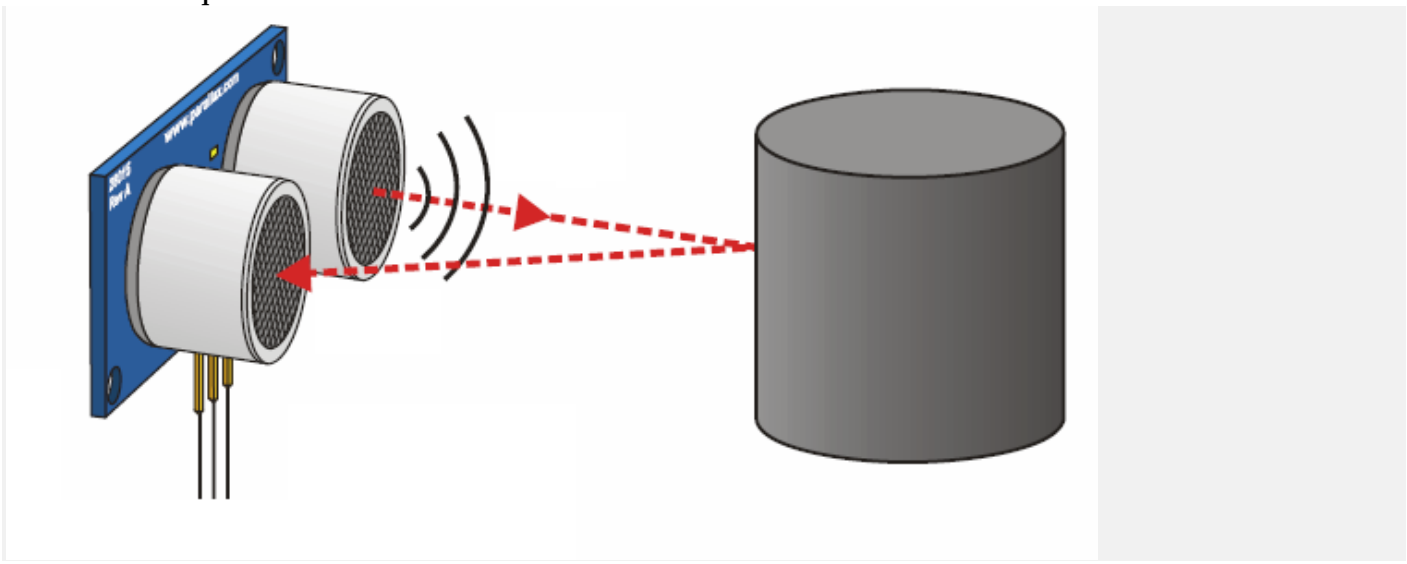
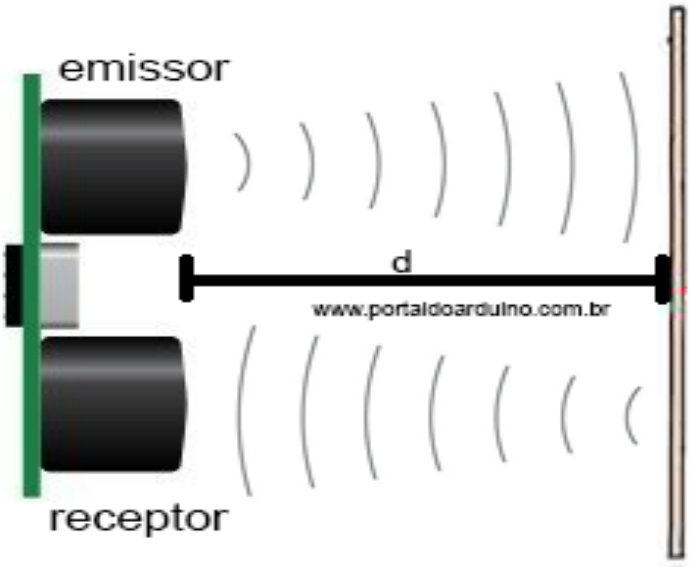
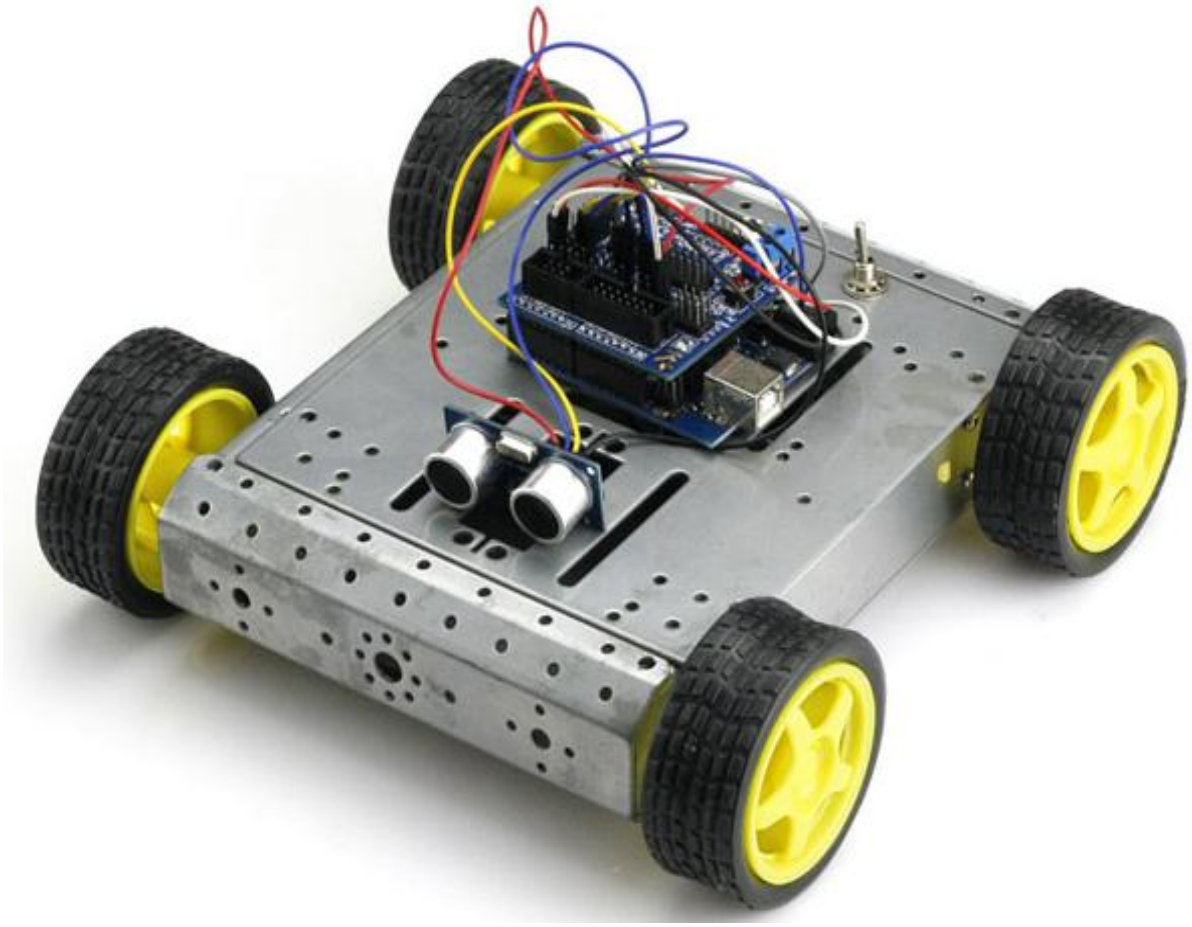


Figura 1 – Princípio de Funcionamento sensor ultrassom



Como a onda percorre a distância ‘d’ uma vez para chegar até o obstáculo, e uma segunda vez para voltar ao receptor, a distância total é igual a ‘2d’.



UTILIZANDO O HC-SR04 NO ARDUINO

O módulo é composto por transmissor, receptor e circuito de controle. Para iniciarmos uma medição, o pino Trig deve receber um pulso de 5V por pelo menos 10 us. Isso fará com que o sensor emita 8 pulsos ultrassônicos em 40kHz e inicie a espera pelas ondas refletidas. Assim que uma onda refletida for detectada, o pino Echo que que estava em 0V será alterado para 5V por um período igual ao tempo de propagação da onda.

O sensor ultrasônico HC-SR04 que pode ser utilizado como um detector de objetos ou na área de robótica um componente que pode ser usado para encontrar/evitar obstáculos ou corrigir rotas na movimentação do robô.

Esse sensor utiliza sinais ultrasônicos (40 Khz, acima da capacidade de audição do ouvido humano, que é de 20 Khz), para determinar a distância entre o sensor e o obstáculo. Ele pode medir distâncias entre 2 cm e 4 m, com precisão de 3mm. Seu ângulo de detecção é de aproximadamente 15 graus, segundo informações do datasheet do sensor.

O módulo possui 4 pinos : Vcc (alimentação 5V), Trigger, Echo e GND, sendo ideal para utilização em projetos compactos, já que consome apenas 15mA, se adaptando bem à projetos que utilizam as placas e microcontroladores mais comuns do mercado como Arduino, Raspberry e PIC.

Seu funcionamento consiste basicamente em enviar um sinal que, ao atingir um objeto, volta para o sensor e com base nesse tempo entre o envio e recebimento, é calculada a distância entre o sensor e o objeto.

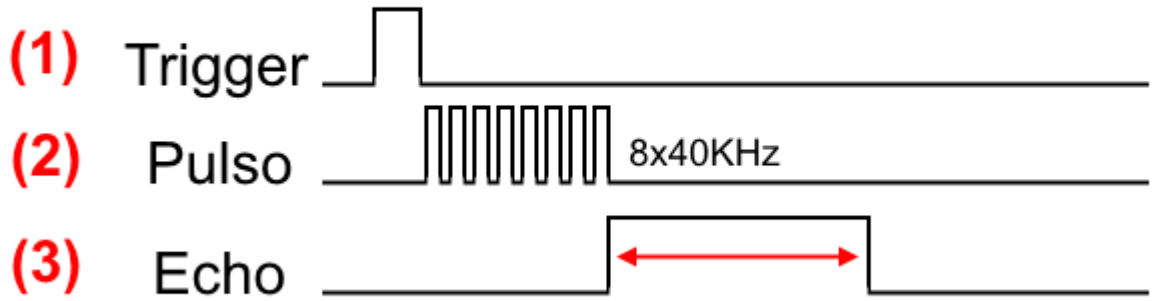
Analisando com mais detalhes esse processo de medição, que ocorre em 3 etapas:

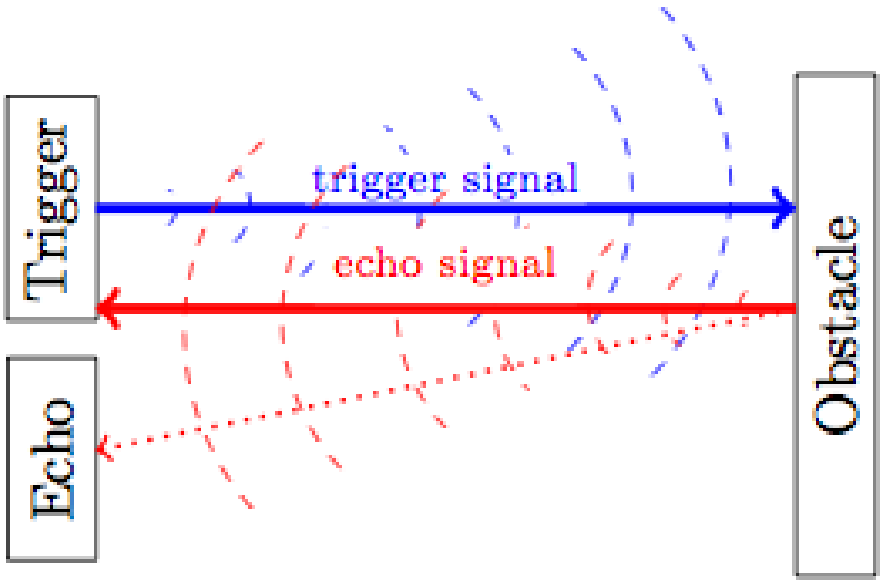
1. É enviado um sinal com duração de 10 us (microssegundos) ao pino trigger, indicando que a medição terá início
2. Automaticamente, o módulo envia 8 pulsos de 40 KHz e aguarda o retorno do sinal pelo receptor
3. Caso haja um retorno de sinal (em nível HIGH), determinamos a distância entre o sensor e o obstáculo utilizando a seguinte equação: DISTANCIA = (PULSO EM NÍVEL ALTO X VELOCIDADE DO SOM (340M/S) /2

A divisão por 2 é necessária já que estamos contando o tempo de ida e volta do sinal.

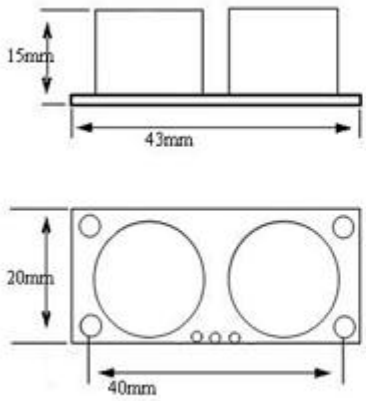
A ilustração abaixo mostra, graficamente, esse processo:

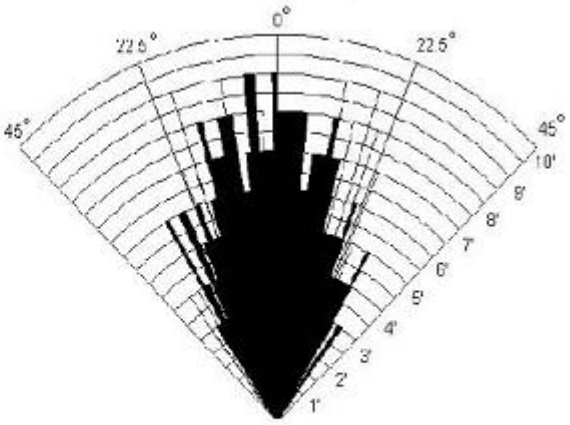
Diagrama de tempo HC-SR04





- Para uma melhor medição da distância, a área do objeto na qual a onda será refletida deve ser de pelo menos 0,5 m^2.

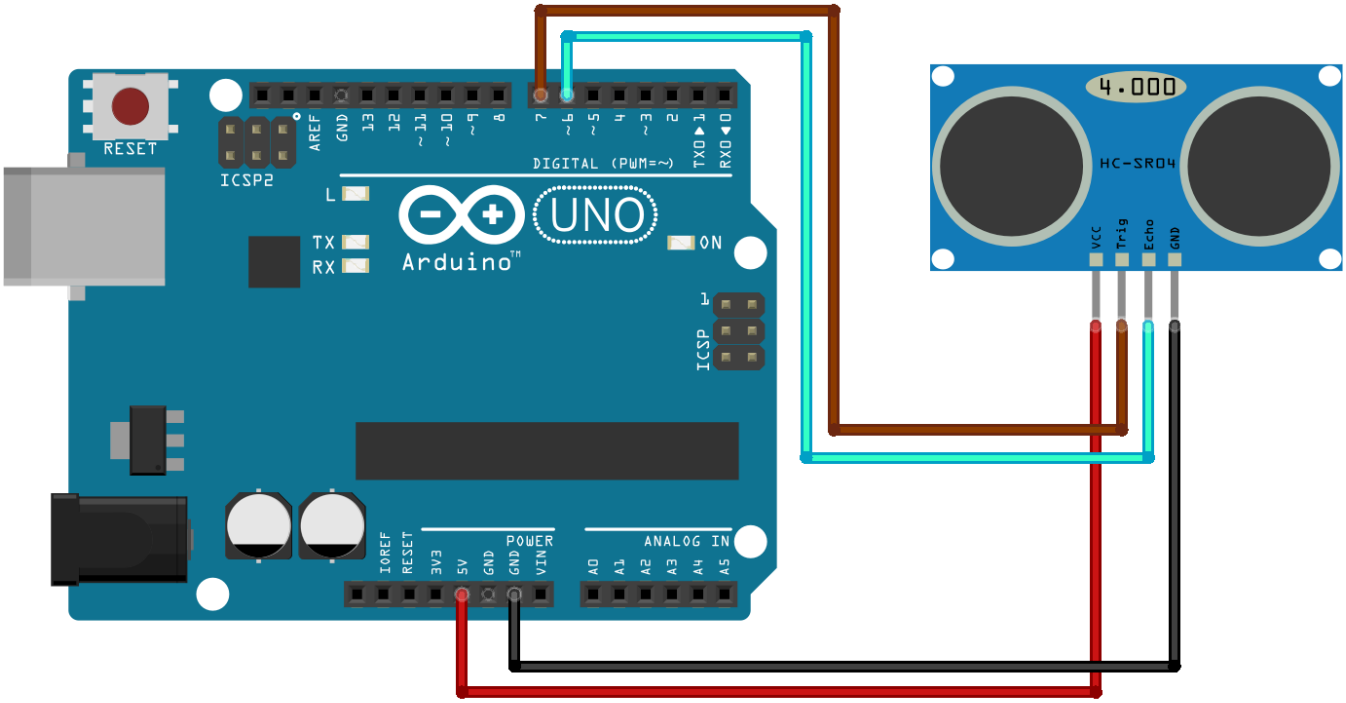




Practical test of performance,

Best in 30 degree angle

Exemplo 01:

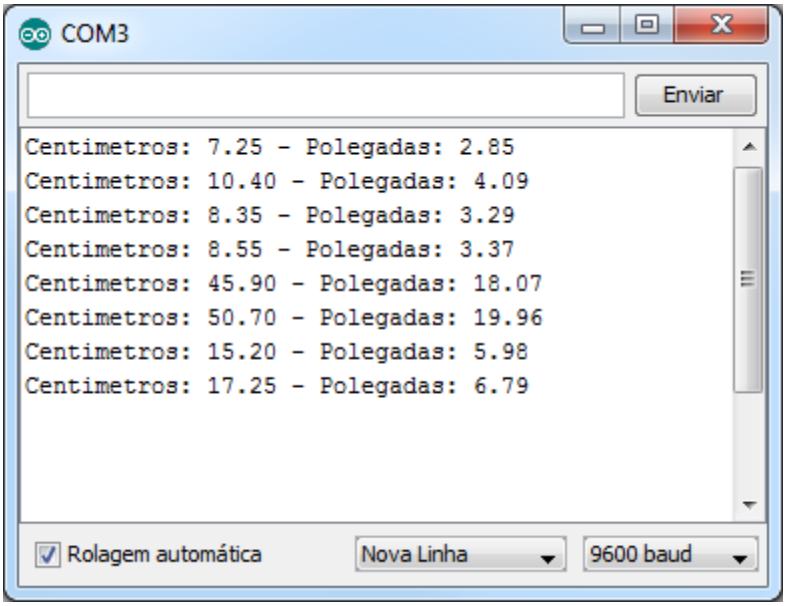


O programa para esse circuito utiliza a biblioteca Ultrasonic, que funciona muito bem quando você precisa trabalhar com apenas um sensor HC-SR04 no seu projeto. Você pode baixar a biblioteca Ultrasonic nesse link. Descompacte o arquivo e coloque a pasta dentro da pasta libraries da IDE do seu Arduino. Em seguida, carregue o programa abaixo :

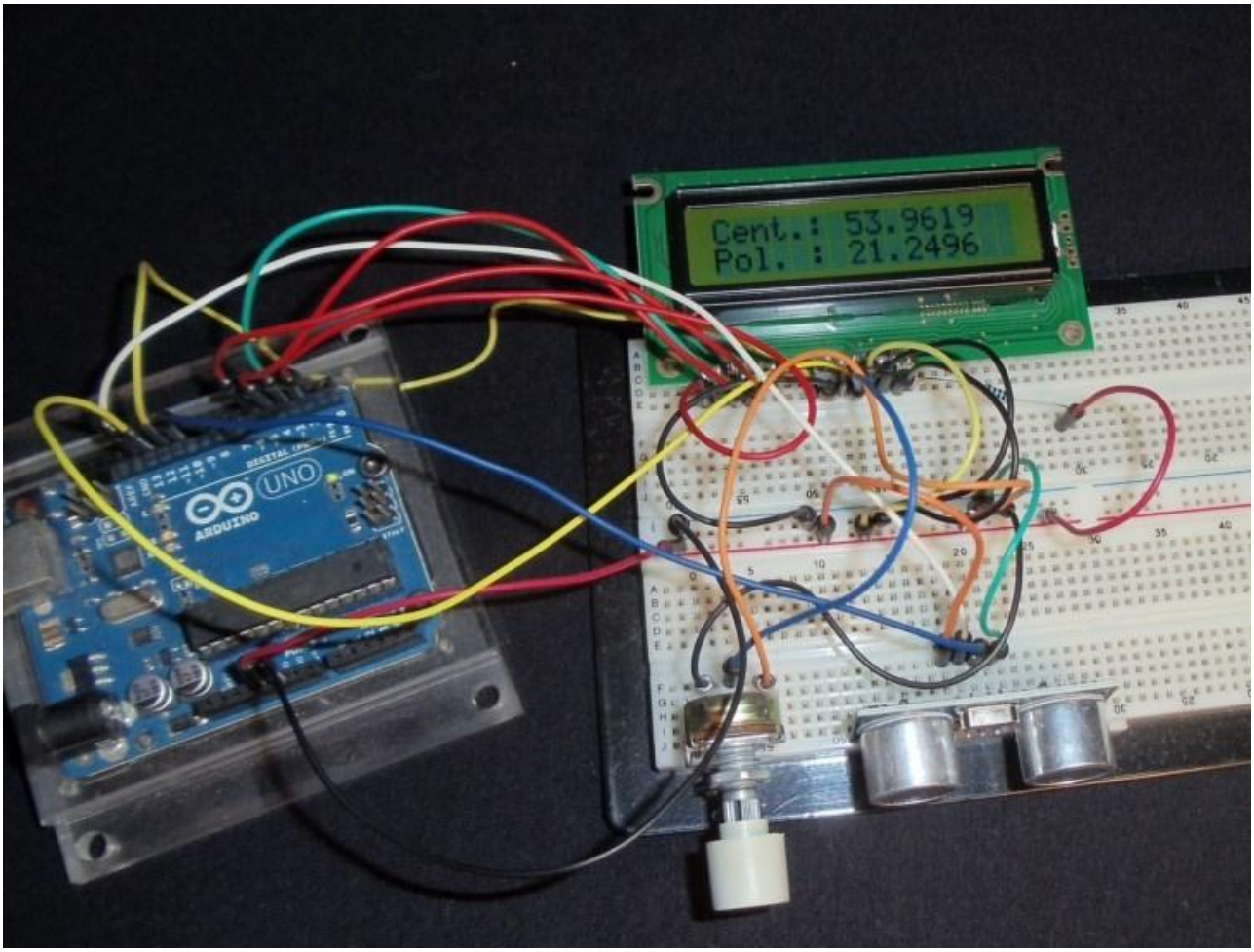
```
1   //Programa : Teste HC-SR04 e biblioteca Ultrasonic
2
3   #include <Ultrasonic.h>
4
5   //Define os pinos do Arduino ligados ao Trigger e Echo
6   #define PINO_TRG  7
7   #define PINO_ECHO 6
8
9   //Inicializa o sensor ultrasonico nos pinos especificados
10  Ultrasonic ultrasonic(PINO_TRG, PINO_ECHO);
11
12  void setup()
13  {
14          //Inicializa a serial
15          Serial.begin(9600);
16  }
17
18  void loop()
19  {
20          //Variaveis para guardar os valores em
21          //cm (cmSec) e polegadas (inMsec)
22          float cmMsec, inMsec;
23
24          //Le os valores do sensor ultrasonico
25          long microsec = ultrasonic.timing();
26          //Atribui os valores em cm ou polegadas as variaveis
```

```
27          cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);
28          inMsec = ultrasonic.convert(microsec, Ultrasonic::IN);
29
30              //Mostra os valores na serial
31              Serial.print("Centimetros: ");
32              Serial.print(cmMsec);
33              Serial.print(" - Polegadas: ");
34              Serial.println(inMsec);
35
36              //Aguarda 1 segundo e reinicia o processo
37              delay(1000);
38      }
39
```

Esse programa lê as informações do sensor e as envia para o serial monitor, mostrando a distância do sensor ao objeto em centímetros e também em polegadas.



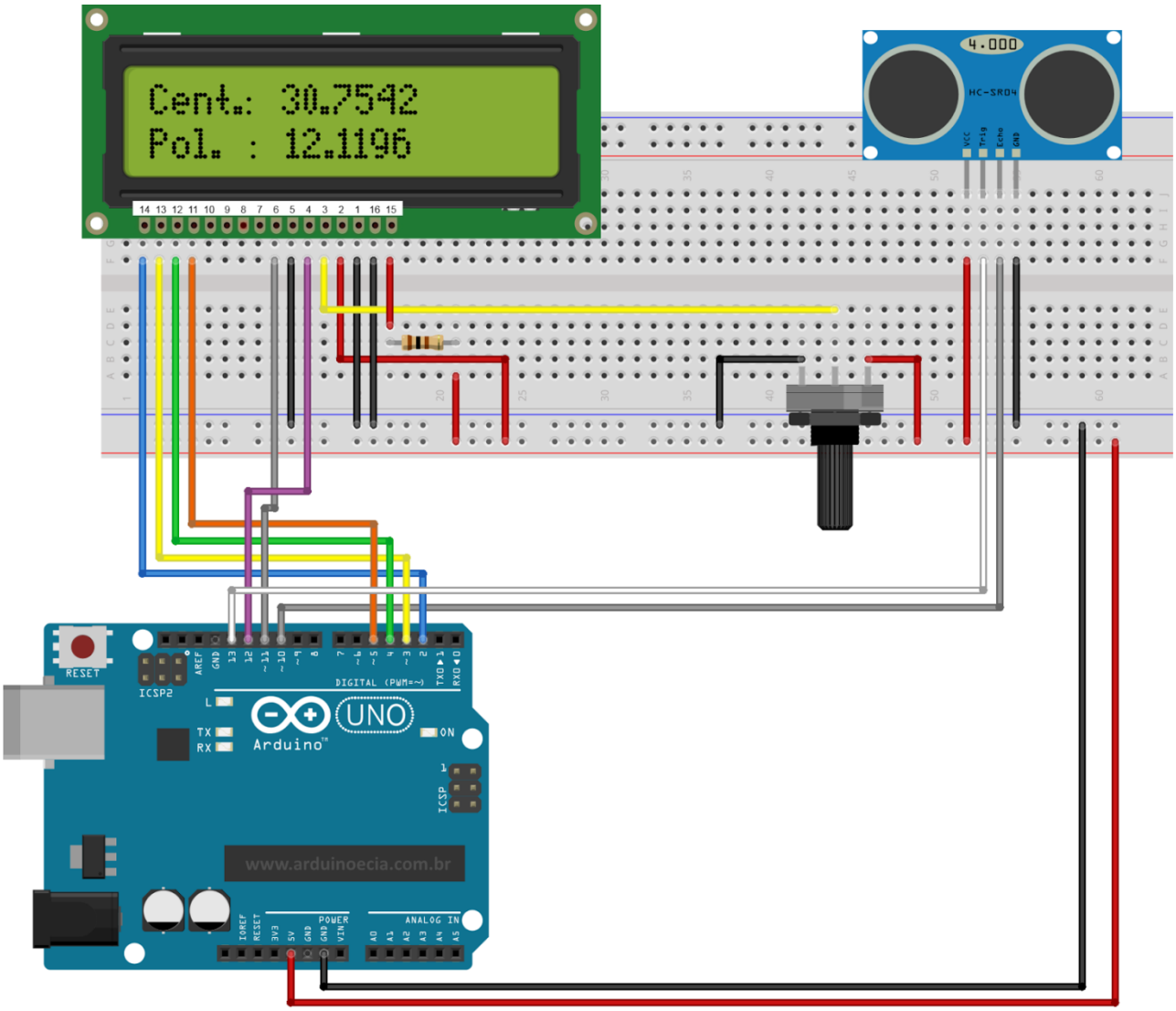
Exemplo 02: Medidor de distância com o sensor ultrassônico HC-SR04 e display LCD



O HC-SR04 pode ser usado para medir distâncias de até 4m com precisão de 3mm (dados do datasheet do produto) e identificar a presença de objetos ou captar movimentos.

Isso nos permite criar alarmes, sensores de presença, sistemas de escuta, sensores de estacionamento, sensores de obstáculos para serem usados em robôs, e muitas outras aplicações.

Resumindo o funcionamento: é enviado um sinal ultrassônico pelo módulo, o mesmo detecta o retorno deste sinal (eco), e com base no tempo entre esses dois eventos, gera um sinal que permite medir a distância até o objeto.



O programa usa a biblioteca Ultrasonic, que voce pode baixar neste link. O programa faz a leitura dos dados do sensor, calcula a distancia em centímetros e polegadas, e apresenta os dados no LCD 16x2.

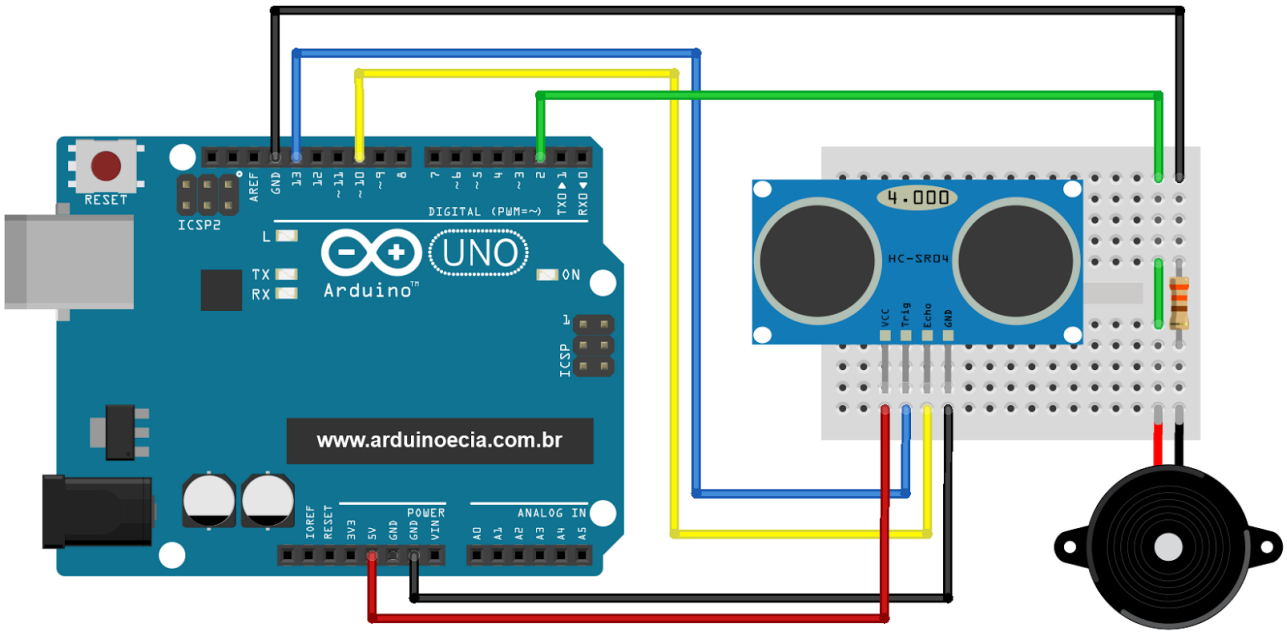
```
1  //Programa : Medidor de distancia com HC-SR04
2  //Autor : Arduino e Cia
3
4  #include <Ultrasonic.h>     //Carrega a biblioteca Ultrasonic
5  #include <LiquidCrystal.h>  //Carrega a biblioteca LCD
6
7  //Define o pino do Arduino a ser utilizado com o pino Trigger do sensor
8  #define PINO_TRIGGER  13
9
10 //Define o pino do Arduino a ser utilizado com o pino Echo do sensor
11 #define PINO_ECHO     10
12
13 //Inicializa o sensor ultrasonico
14 Ultrasonic ultrasonic(PINO_TRIGGER, PINO_ECHO);
15
16 //Define os pinos que serão ligados ao LCD
```

```
17 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
18
19 void setup()
20 {
21         Serial.begin(9600); //Inicializa a serial
22         lcd.begin(16,2); //Inicializa LCD
23         lcd.clear();     //Limpa o LCD
24 }
25
26 void loop()
27 {
28         float cmMsec, inMsec;
29
30         //Le os dados do sensor, com o tempo de retorno do sinal
31         long microsec = ultrasonic.timing();
32
33         //Calcula a distancia em centimetros
34         cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);
35
36         //Calcula a distancia em polegadas
37         inMsec = ultrasonic.convert(microsec, Ultrasonic::IN);
38
39         //Apresenta os dados, em centimetros, no LCD e na Serial
40         lcd.setCursor(0,0);
41         lcd.print("Cent.: ");
42         lcd.print("       ");
43         lcd.setCursor(7,0);
44         lcd.print(cmMsec);
45
46         Serial.print("Cent: ");
47         Serial.print(cmMsec);
48
49         //Apresenta os dados, em polegadas, no LCD e na Serial
50         lcd.setCursor(0,1);
51         lcd.print("Pol. : ");
52         lcd.print("       ");
53         lcd.setCursor(7,1);
54         lcd.print(inMsec);
55
56         Serial.print(", Pol. : ");
57         Serial.println(inMsec);
58
59         delay(1000);
60 }
```

Exemplo 03: Sensor de estacionamento / ré com Arduino e sensor ultrasônico HC-SR04

Neste projeto vamos utilizar o Arduino juntamente com o **sensor ultrasônico HC-SR04** para montar um sensor de estacionamento (ou sensor de ré), que emite um "bip" conforme o sensor se aproxima de um obstáculo. Quanto mais próximo o obstáculo, maior a frequência do bip.

O circuito é composto apenas pelo Arduino, um **sensor ultrasônico HC-SR04** e um buzzer, com uma resistência de 330 ohms para limitar a corrente.



No programa, utilizei a biblioteca **NEWTONE**, que pode ser baixada **nesse link** . O motivo de usar essa biblioteca é que a função **TONE**, normalmente usada para produzir sons com Arduino, apresenta conflito com a biblioteca ***Ultrasonic***, também utilizada nesse projeto.

O som do bip pode ser alterado conforme a necessidade, alterando o valor das variáveis **frequência** e **tempo**, setadas no começo do programa.

O pino **Trigger** do sensor vai ligado ao pino 13 do Arduino, e o pino **Echo** do sensor vai ligado ao pino 10 do Arduino. A alimentação do sensor é de 5 volts. O buzzer vai na porta 2 do Arduino.

A variável **cmMsec** armazena o valor da distância entre o sensor e o obstáculo. Uma série de comandos **IF** verificam o valor de **cmMsec** e atualizam a variável **atraso**, que é utilizada no final do programa para determinar a frequência de acionamento do bip.

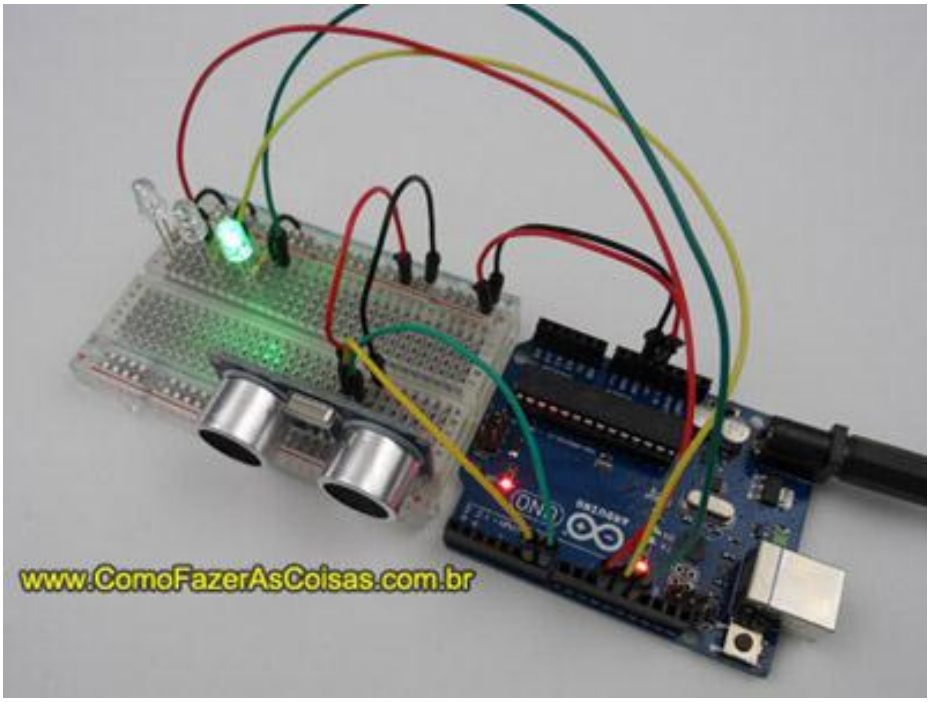
Os valores lidos pelo sensor também podem ser acompanhados pelo serial monitor e, com algumas alterações no programa, exibidos em um display.

```
1  // Programa : Sensor de estacionamento com HC-SR04
2
3  #include <Ultrasonic.h>   //Carrega a biblioteca Ultrasonic
4  #include <NewTone.h>    //Carrega a biblioteca Newtone
5
6  //Dados do buzzer
7  #define tempo 500
8  int frequencia = 2000;
9  int Pinofalante = 2;
10
11 int atraso = 1000;
12
13 //Define o pino do Arduino a ser utilizado com o pino Trigger do sensor
14 #define PINO_TRIGGER 13
15 //Define o pino do Arduino a ser utilizado com o pino Echo do sensor
16 #define PINO_ECHO   10
17
18 //Inicializa o sensor ultrasonico
19 Ultrasonic ultrasonic(PINO_TRIGGER, PINO_ECHO);
20
21 void setup()
22 {
23         pinMode(Pinofalante,OUTPUT); //Pino do buzzer
24         Serial.begin(9600); //Inicializa a serial
25 }
26
27 void loop()
28 {
29         float cmMsec, inMsec;
30
31         //Le os dados do sensor, com o tempo de retorno do sinal
32         long microsec = ultrasonic.timing();
33
34         //Calcula a distancia em centimetros
35         cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);
36
37         //Ajusta o atraso de acordo com a distancia
```

```
38          if (cmMsec > 80)
39          {
40                  atraso = 2000;
41          }
42          else if (cmMsec >50 && cmMsec<80)
43          {
44                  atraso = 1500;
45          }
46          else if (cmMsec >30 && cmMsec<50)
47          {
48                  atraso = 1200;
49          }
50          else if (cmMsec > 10 && cmMsec < 30)
51          {
52                  atraso = 700;
53          }
54          else if (cmMsec < 10)
55          {
56                  atraso = 300;
57          }
58
59          //Apresenta os dados, em centimetros, na Serial
60          Serial.print("Cent: ");
61          Serial.print(cmMsec);
62          Serial.print(" atraso : ");
63          Serial.println(atraso);
64          //Emite o bip
65          NewTone(Pinofalante, frequencia, tempo);
66
67          delay(atraso);
68  }
69
```

Exemplo 04: Arduino com sensor de obstáculos ultrassônico HC-SRO4

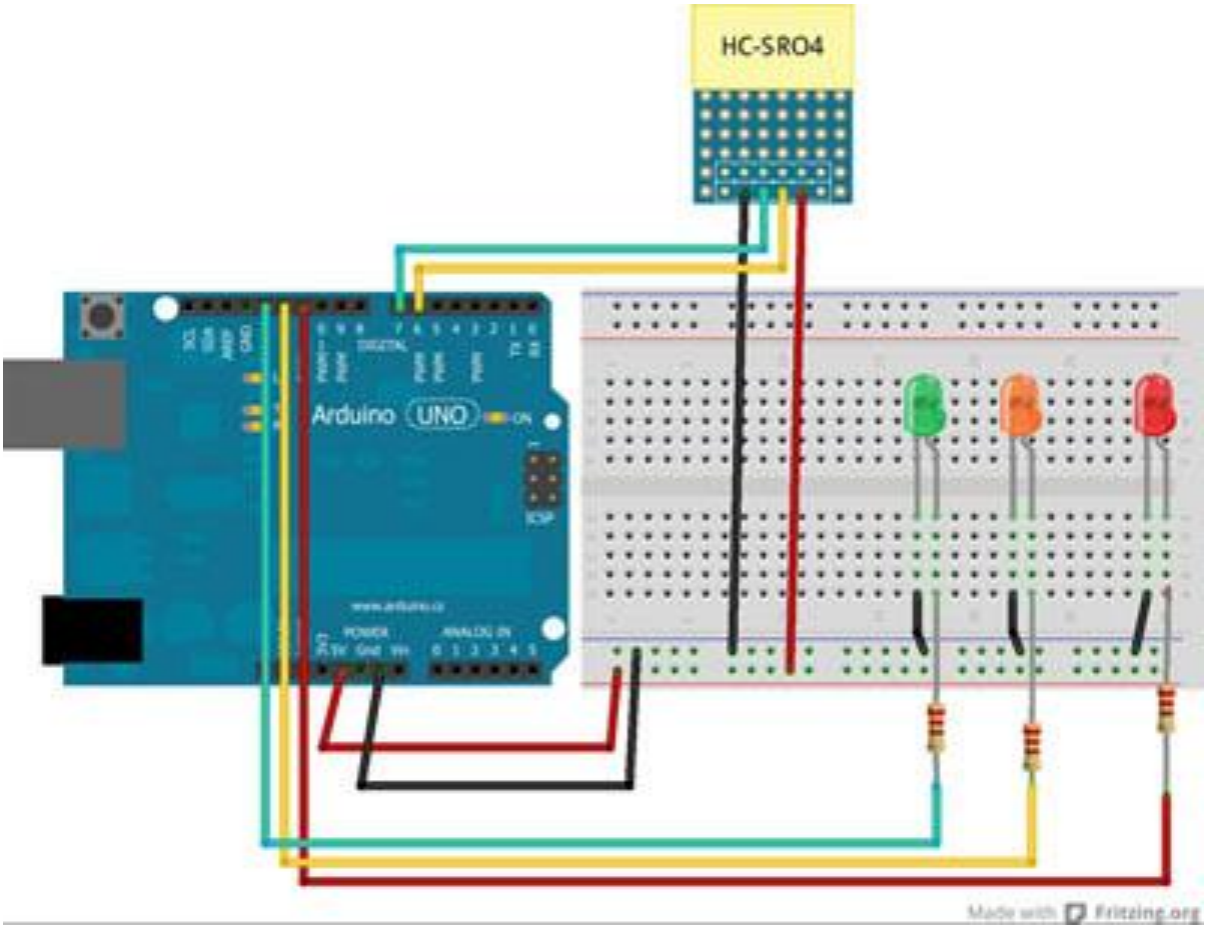
O funcionamento do projeto é simples, à medida que você aproxima algum objeto do sensor os LEDs acendem conforme a distância do objeto. O LED verde se estiver longe, o amarelo se estiver em uma distância média e o vermelho se estiver próximo. As distâncias de acendimento dos LEDs são programáveis, se você quiser alterar é só fazer isso no código fonte do projeto.



O primeiro passo é instalar a biblioteca do sensor HC-SRO4 para que seu programa funcione. Agora monte seu projeto físico conforme o esquema abaixo.

As ligações ficaram da seguinte forma:

- portas digitais 11, 12 e 13 do arduino ligadas nos resistores, que por sua vez estão ligados nas pernas positivas dos leds;
- pernas negativas dos leds no GND (terra) do arduino;
- pino VCC do sensor ultrasônico HC-SRO4 no 5V do arduino;
- pino TRIG do sensor HC-SRO4 na porta digital 6 do arduino;
- pino ECHO do sensor HC-SRO4 na porta digital 7 do arduino;
- pino GND do sensor HC-SRO4 no GND do arduino.



Abaixo segue o código fonte deste projeto:

```
1   /*
2   Projeto Arduino
3   Arduino com sensor de proximidade ultrasonico HHC-SRO4
4   */
5
6   //Incluindo biblioteca Ultrasonic.h
7   #include "Ultrasonic.h"
8
9   //criando objeto ultrasonic e definindo as portas digitais
```

```
10  //do Trigger - 6 - e Echo - 7
11  Ultrasonic ultrasonic(6,7);
12
13  //Declaração das constantes referentes aos pinos digitais.
14  const int ledVerde = 13;
15  const int ledAmarelo = 12;
16  const int ledVermelho = 11;
17
18  long microsec = 0;
19  float distanciaCM = 0;
20
21  void setup()
22  {
23          Serial.begin(9600); //Inicializando o serial monitor
24
25          //Definindo pinos digitais
26          pinMode(ledVerde,OUTPUT); //13 como de saída.
27          pinMode(ledAmarelo,OUTPUT); //12 como de saída.
28          pinMode(ledVermelho,OUTPUT); //11 como de saída.
29  }
30
31  void loop()
32  {
33          //Lendo o sensor
34          microsec = ultrasonic.timing();
35
36          //Convertendo a distância em CM
37          distanciaCM = ultrasonic.convert(microsec, Ultrasonic::CM);
38
39          ledDistancia();
40
41          Serial.print(distanciaCM);
42          Serial.println(" cm");
43          delay(1000);
44  }
45
46  //Método que centraliza o controle de acendimento dos leds.
47  void ledDistancia()
48  {
49            //Apagando todos os leds
50          digitalWrite(ledVerde,LOW);
51          digitalWrite(ledAmarelo,LOW);
52          digitalWrite(ledVermelho,LOW);
53
54          //Acendendo o led adequado para a distância lida no sensor
55          if (distanciaCM > 20)
56          {
57              digitalWrite(ledVerde,HIGH);
58          }
59
60          if (distanciaCM <=20 and distanciaCM >= 10)
61          {
62              digitalWrite(ledAmarelo,HIGH);
63          }
```

```
64
65              if (distanciaCM < 10)
66              {
67                      digitalWrite(ledVermelho,HIGH);
                }
        }
```