

# Internet das Coisas

## Aula 15 – Leitura de sensores especiais





# Leitura de sensores especiais

Vimos até aqui como ler os valores de sensores digitais de resposta binária (de forma direta ou com uso de interrupções) e como ler valores de sensores analógicos (de forma direta ou com o uso de mapping de valores). Os sensores de cada um desses dois grupos, possuem um modo de uso similar, sendo perceptível que ao aprender um, entendemos facilmente o funcionamento de outros, facilitando a nossa aprendizagem.

# Leitura de sensores especiais

Porém, alguns sensores possuem um funcionamento todo próprio e que difere totalmente daqueles vistos até aqui. Alguns deles são digitais, mas ao invés de nos devolver uma resposta binária (LOW ou HIGH), devolvem uma sequência binária através de uma combinação de mudanças de estado em uma determinada frequência (senso necessário um sincronismo para a leitura e, após, a conversão do valor para base decimal). Outros, nos devolvem mais de um resultado em uma única sequência binária, sendo necessárias operações de base binária para separar os bits de cada informação.

# Leitura de sensores especiais

Outros por causa de sua complexidade nos oferecem bibliotecas que já possuem métodos para adquirirmos facilmente o valor, mas que devemos instanciar um objeto através do qual os métodos são acessados. Alguns, possuem objetos que devem ser configurados ou cujos pinos devem ser previamente informados. Com isso, alguns sensores necessitam de um estudo em separado para entendermos o seu funcionamento e o seu consequente uso.

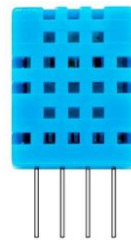
# Leitura de sensores especiais



sensor RFID



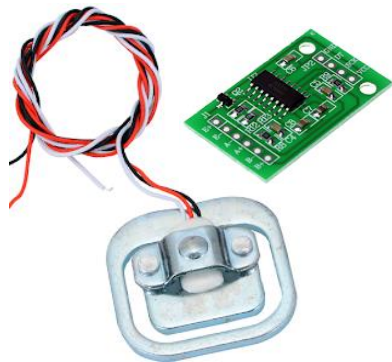
sensor ultrassônico  
de distância



sensor de umidade +  
temperatura



sensor de  
reconhecimento de voz



sensor de carga com  
amplificador



sensor de cores RGB



sensor de gestos



sensor acelerômetro e  
giroscópio



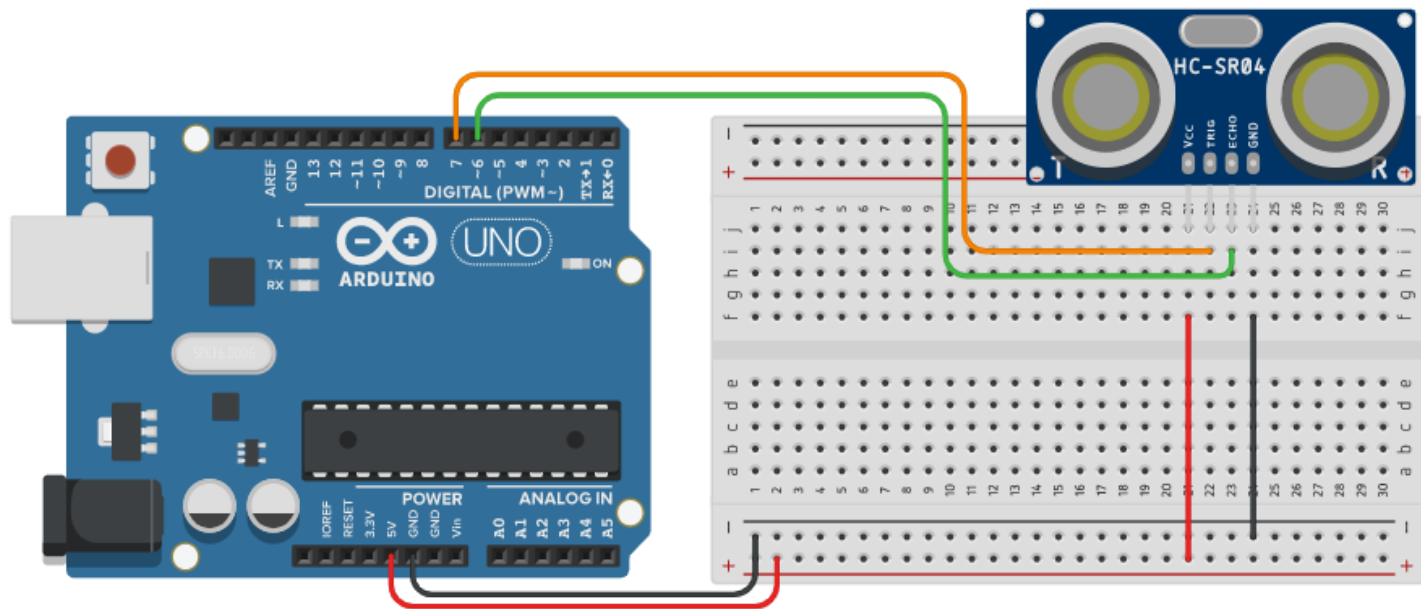
# Leitura de sensores especiais

Não vamos nesta aula aprofundar em todos eles, como fizemos nas aulas anteriores com os sensores digitais de respostas binária e com os sensores analógicos. Porém, para alguns deles vamos mostrar exemplos de uso (alguns detalhados, outros resumidos) e para outros vamos comentar o seu uso e importância e indicar sites que explicam de forma detalhada projetos com o seu uso.

# Ex1: sensor ultrassônico (HC-SR04)

O sensor ultrassônico de distância é um sensor capaz de medir a distância do objeto mais próximo à sua frente. Imagine então que queremos que um sensor ultrassônico de distância nos informe a distância do objeto mais próximo encontrado. Este tipo de sensor possui, além do 5V e do GND, dois outros pinos que devem ser ligados à duas portas digitais: o Echo (responsável por emitir um sinal ultrassônico) e o Trigger (responsável por receber de volta esse sinal refletido e contar o tempo que ele demorou para chegar no sensor). Para isso, vamos fazer a seguinte ligação eletrônica, como no esquema abaixo (trigger na porta 7 e echo na 6).

# Ex1: sensor ultrassônico (HC-SR04)

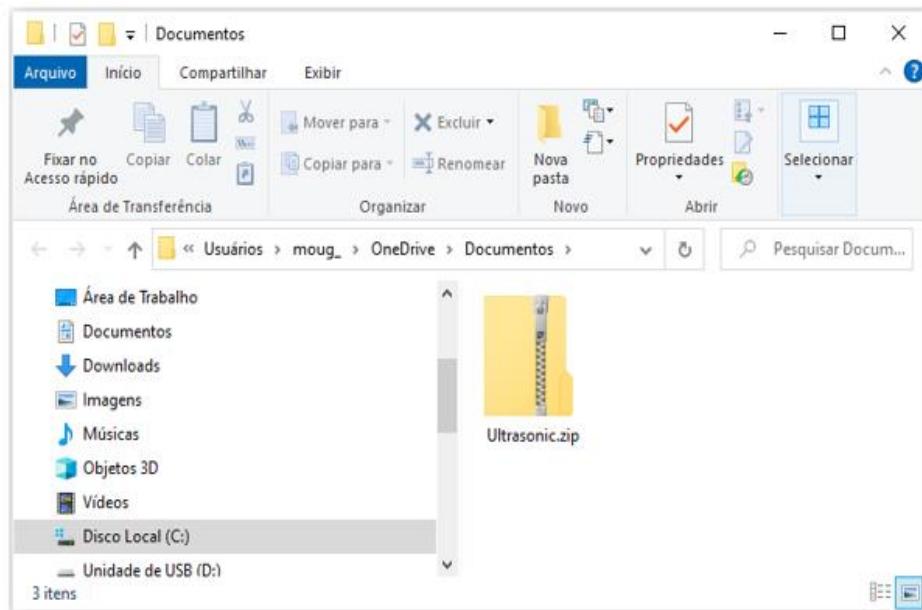
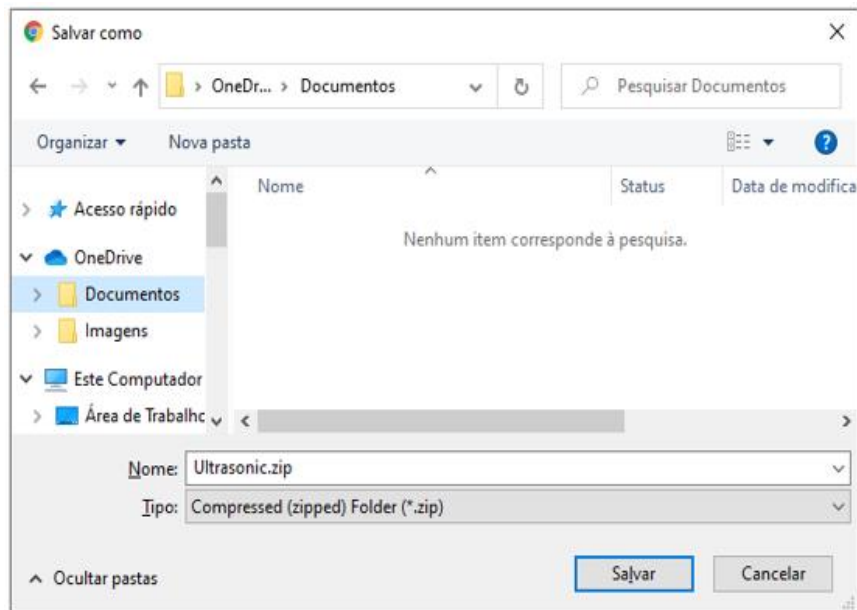




## Ex1: sensor ultrassônico (HC-SR04)

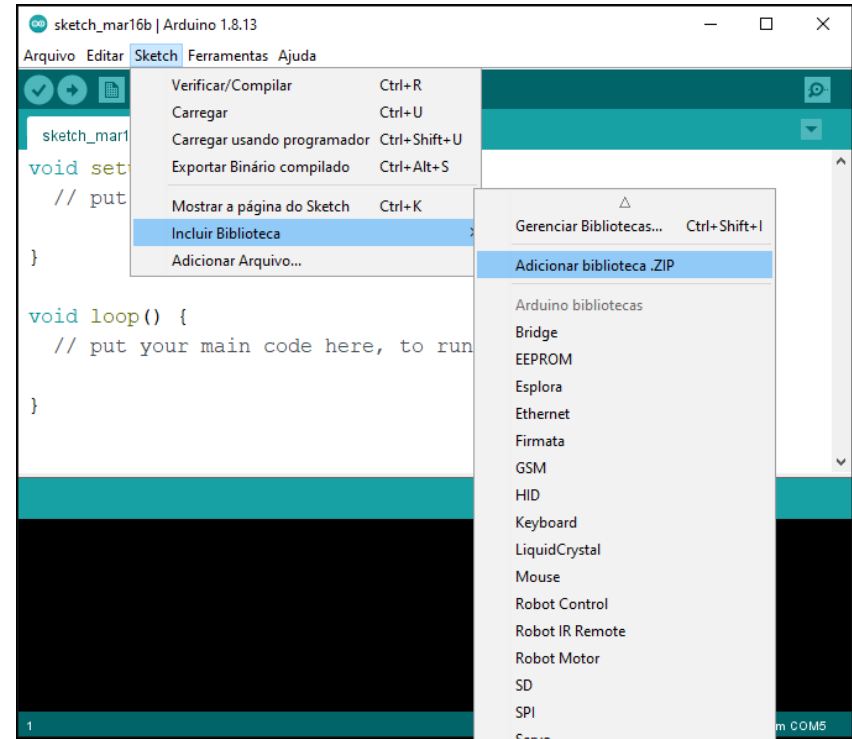
Por ser um sensor de valor binário, este sensor ultrassônico devolve como resposta uma sequência de bits (0 e 1, através da alternância do estado baixo e alto na porta), cuja captura e manipulação direta exigiria um trabalho grande. Por isso, assim como todos os sensores de valor binário, ele possui uma biblioteca própria que contém métodos já programados para uso e que facilitam a captura do valor medido. Neste caso, a biblioteca é a [“Ultrasonic.h”](#). Precisamos então fazer o download da mesma.

# Ex1: sensor ultrassônico (HC-SR04)



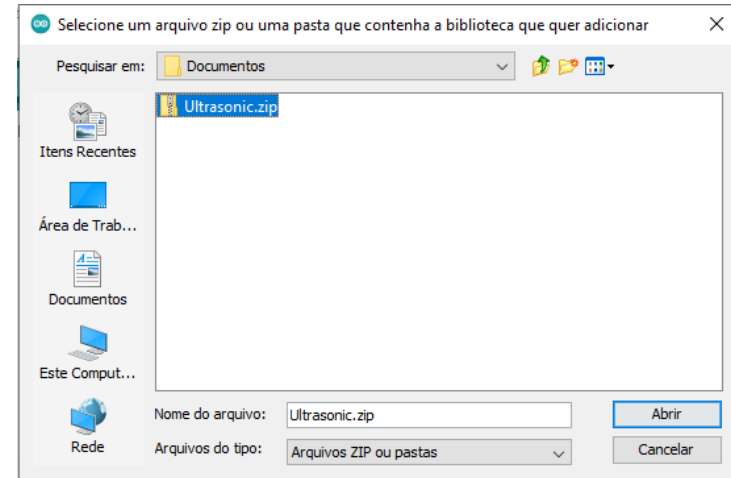
# Ex1: sensor ultrassônico (HC-SR04)

Após o download, precisamos adicionar à biblioteca à nossa Arduino IDE. Para isso, com ela aberta, devemos clicar no menu “Sketch”, após em “Incluir biblioteca” e finalmente na opção “Adicionar biblioteca .ZIP”:



# Ex1: sensor ultrassônico (HC-SR04)

Após, devemos procurar o local onde a biblioteca foi salva, selecioná-la e clicar no botão “Abrir”. Após isso, a biblioteca está incluída em nossa IDE e pode ser utilizada não somente nesse projeto, mas em qualquer futuro projeto desenvolvido nela.



# Ex1: sensor ultrassônico (HC-SR04)

Antes de irmos para a escrita do código, precisamos saber de 3 coisas importantes:

- Mesmo com a biblioteca já incluída na IDE, para utilizá-la em nosso projeto devemos realizar um include (inclusão) no código;
- O acesso ao sensor ultrassônico, conforme definição da biblioteca, se dá através da declaração de um objeto Ultrasonic, que ao ser criado deve receber como parâmetro duas informações: a porta digital do Arduino onde está ligado o pino Trigger (no nosso exemplo, na 7) e a porta digital do Arduino onde está ligado o pino Echo (no nosso exemplo, na 6);

## Ex1: sensor ultrassônico (HC-SR04)

- Também conforme definição da biblioteca, o objeto Ultrasonic possui um método específico que pode ser chamado para devolver o valor da distância lido pelo sensor. Esse método é o Ranging e quando o utilizamos, devemos passar por parâmetro ou o valor CM (caso queiramos a distância em centímetros) ou o valor INC (caso queiramos a distância em polegadas).



## Ex1: sensor ultrassônico (HC-SR04)

Com essas informações, vamos construir um código que a cada 1 segundo, ativa o sensor ultrassônico e pede a ele o valor da distância do objeto mais próximo, capturando esse valor e escrevendo na nossa Serial, para que possamos visualizá-lo através do Monitor Serial da nossa IDE. Deste modo temos:

# Ex1: sensor ultrassônico (HC-SR04)

```
#include <Ultrasonic.h>

Ultrasonic ultrassom(7,6);

void setup(){
  Serial.begin(9600);
}

void loop(){
  delay(1000);

  float distancia = ultrassom.Ranging(CM);
  Serial.print(distancia);
  Serial.println(" cm ");
}
```

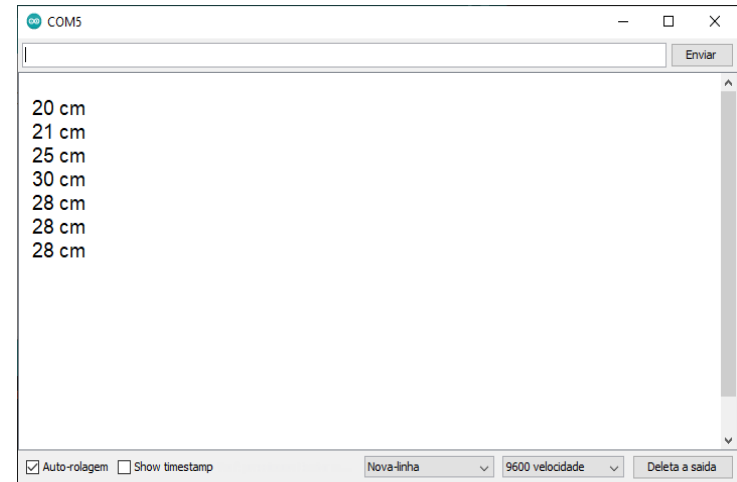


# Ex1: sensor ultrassônico (HC-SR04)

**Entendendo o código:** começamos declarando um include à biblioteca “Ultrasonic.h” para que nosso projeto faça uso dela e, logo abaixo, criamos um objeto do tipo Ultrasonic (ao qual demos o nome de ultrassom) ao qual informamos que o pino Trigger está ligado à porta digital 7 e o pino Echo está ligado à porta digital 6. No setup, a única coisa necessária é a inicialização da nossa Serial para poder mandar para ela o resultado lido, tornando possível a visualização no monitor Serial. Perceba que em nenhum momento informamos se nosso pino 7 e 6 são de entrada ou saída. A própria biblioteca fará isso.

# Ex1: sensor ultrassônico (HC-SR04)

Após, em nosso loop, a cada 1 segundo criamos uma variável distancia e jogamos para dentro dela o resultado da chamada do método Ranging, que nos devolve a distância lida pelo sensor (em centímetros, por causa do parâmetro CM utilizado). Após, esta distância é escrita na Serial, acompanhada do texto “ cm ”. Assim temos um resultado como o abaixo dentro de nosso Monitor Serial:





## Ex1: sensor ultrassônico (HC-SR04)

Podemos perceber que o acesso aos valores do sensor ultrassônico, apesar de um pouco mais elaborado do que os sensores de resposta binária, não traz um grau de dificuldade alto. A maior dificuldade no uso de sensores deste tipo é justamente saber qual biblioteca utilizar, de onde baixar, qual o objeto a ser criado e seus parâmetros de inicialização e qual os métodos para o acesso aos valores mensurados ou estimados pelo sensor.

## Ex2: sensor de cor RGB (TCS230)

O sensor de cor RGB é um sensor capaz de estimar a cor de um objeto à sua frente, devolvendo os valores de R, G e B que compõe esta cor. Este sensor possui três fotodiodos capazes de reagir a determinadas cores, produzindo correntes elétricas. Este tipo de sensor possui, além do 5V e do GND, uma série de outros pinos responsáveis por ajustar a frequência de comunicação (S0 e S1), o filtro para cada uma das cores que formam o RGB (S2 e S3), a saída de sinal (OUT) além de pinos auxiliares.

## Ex2: sensor de cor RGB (TCS230)

Primeiramente, a frequência determina a velocidade da saída dos pulsos e a modulação entre eles, para que seja possível a correta configuração de leitura. No Arduino e demais placas de prototipagem, o comum é a configuração em 20%. Obtemos isso, modificando o estado da porta à qual o pino S0 está conectado para HIGH e modificando o estado da porta à qual o pino S1 está conectado para LOW. A configuração das velocidades pode ser verificada na tabela:

S0	S1	Escala de frequência de saída
LOW	LOW	Desligado
LOW	HIGH	2%
HIGH	LOW	20%
HIGH	HIGH	100%

## Ex2: sensor de cor RGB (TCS230)

Após, precisamos informar ao sensor qual é o filtro de cor que queremos ativar. Se ativarmos, por exemplo, o filtro vermelho, o fotodiodo sensível à cor vermelha irá ser ligado e reagir à mesma, produzindo uma corrente baixa que dará origem a um pulso que pode durar de 0 a 255 unidades de tempo e que será enviado à saída OUT. Essa configuração de qual filtro de cor estará ativo é realizada através dos pinos S2 e S3 e pode ser verificada na tabela abaixo:

S2	S3	Filtro de cor ativo
LOW	LOW	vermelho
LOW	HIGH	azul
HIGH	LOW	nenhum (sem filtro ativo)
HIGH	HIGH	verde

## Ex2: sensor de cor RGB (TCS230)

No momento que um filtro de cor é ativado e o pino OUT passa a enviar uma frequência produzida pela reação do fotodiodo àquela cor (dentro da escala de saída configurada), podemos medir o total de unidades de tempo de 0 a 255 de duração da onda. Essa é a forma do sensor nos devolver o valor da cor. Podemos fazer isso através do método `pulseIn`, que pode ser visto abaixo:

```
pulseIn( <porta>, <tipo> );
```

```
<porta>
```

número da porta que queremos medira duração dos pulsos da frequência de saída

```
<tipo>
```

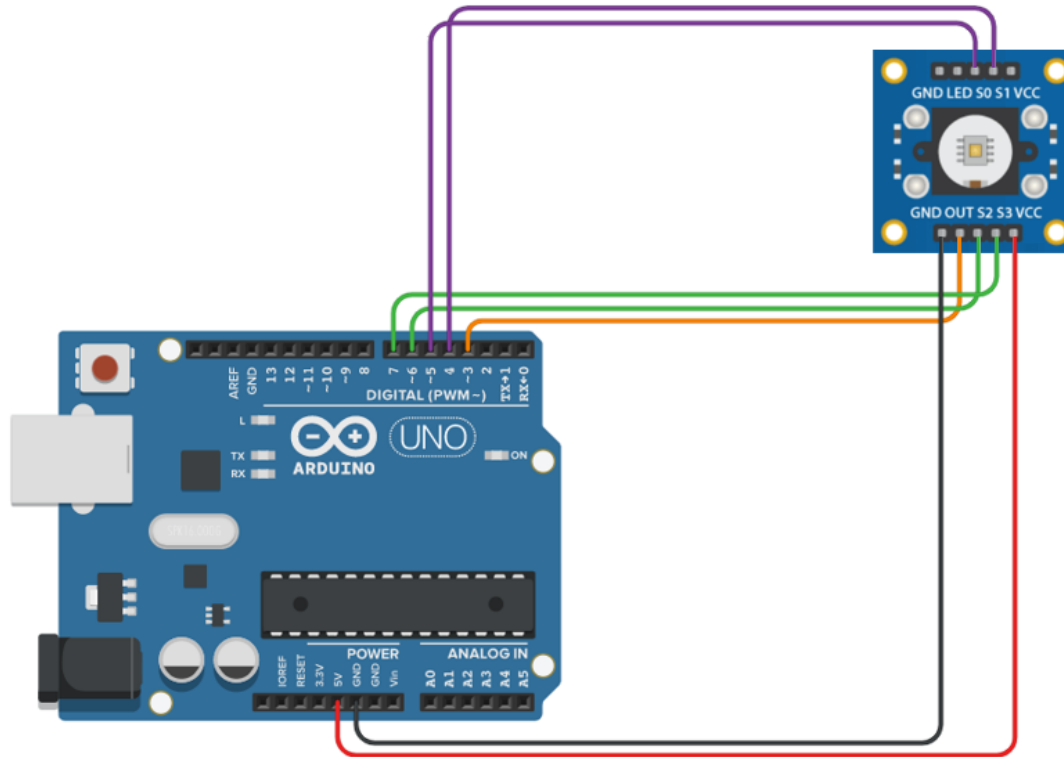
Tipo de pulso a ser lido (LOW - tempo em estado baixo / HIGH - tempo em estado alto)

## Ex2: sensor de cor RGB (TCS230)

Neste nosso exemplo, vamos ligar os pinos S0, S1, S2 e S3 respectivamente nas portas 4, 5, 6 e 7 do Arduino. Além disso, o pino de saída de sinal (OUT) será ligado à porta 3 do Arduino. Deste modo, teremos um circuito eletrônico como o mostrado a seguir (simplificado pela quantidade de fios, sem protoboard e com o sensor ligado direto à placa):



## Ex2: sensor de cor RGB (TCS230)



## Ex2: sensor de cor RGB (TCS230)

Nosso código será baseado na mudança de filtros e na leitura da duração do pulso emitido pela saída, um por vez, para termos os valores de RGB em separado.

```
//criando constantes para mapear os pinos
#define OUT 3;
#define S0 5;
#define S1 4;
#define S2 6;
#define S3 7;

void setup(){
  pinMode(OUT, INPUT);
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);

  //configurando a escala de frequência para 20%
  digitalWrite(S0, HIGH);
  digitalWrite(S1, LOW);

  Serial.begin(9600);
}
```

```
void loop(){
  delay(1000);

  //configurando o filtro vermelho e lendo o valor de R
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  int r = pulseIn(OUT, LOW);

  //configurando o filtro verde e lendo o valor de G
  digitalWrite(S2, HIGH);
  digitalWrite(S3, HIGH);
  int g = pulseIn(OUT, LOW);

  //configurando o filtro azul e lendo o valor de B
  digitalWrite(S2, LOW);
  digitalWrite(S3, HIGH);
  int b = pulseIn(OUT, LOW);

  Serial.println("R: " + (String)r);
  Serial.println("G: " + (String)g);
  Serial.println("B: " + (String)b);
}
```

## Ex2: sensor de cor RGB (TCS230)

**Entendendo o código:** começamos criando constantes que contêm os nomes dos pinos e o valor respectivo. O uso de constantes é bem interessante no C++ para organização do nosso código, principalmente quando temos objetos que possuem um valor fixo, que não se altera ao longo do código. Duas constantes do Arduino bem conhecidas são o LOW (cujo valor mapeado é 0) e o HIGH (cujo valor mapeado é 1). Do modo como foram criadas, ao usar no código S0, por exemplo, automaticamente o mesmo é substituído por seu valor respectivo (no caso 4) no momento da compilação. Após, os pinos correspondentes às constantes criadas, são configurados como saída dentro do setup.

## Ex2: sensor de cor RGB (TCS230)

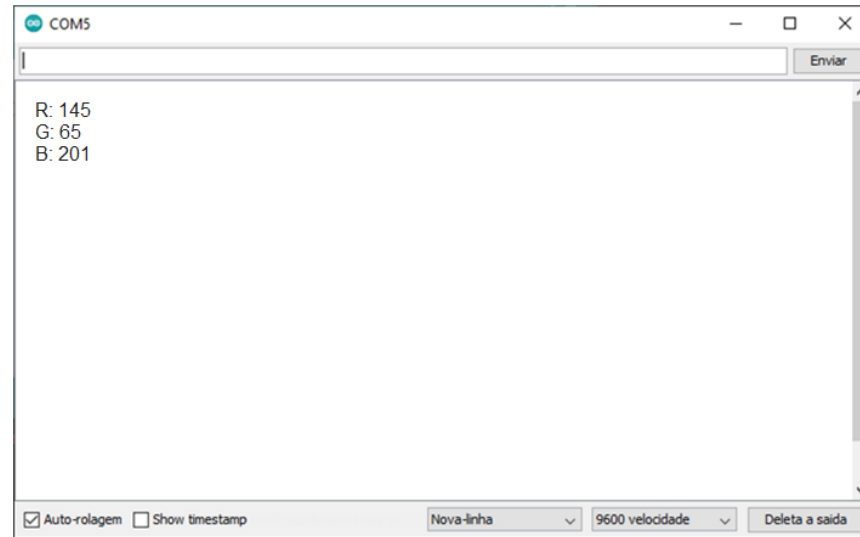
Ainda no setup, a escala de frequência de 20% do sensor (valor padrão para as placas de prototipagem) é configurada através da combinação de S0 em estado HIGH e S1 em estado LOW (conforme a primeira tabela apresentada). Além disso, nossa Serial é inicializada para podermos escrever e visualizar os resultados. Já dentro do loop, para que cada cor seja lida é necessária a configuração do filtro da cor respectivo e a consequente leitura do tempo do pulso gerado pela reação do fotodiodo àquela cor. Como esse valor é mapeado para 0 à 255 unidades de tempo (para que gere um valor de cor entre 0 e 255), a leitura simples do mesmo na saída (OUTPUT) corresponde ao valor daquela cor RGB.

## Ex2: sensor de cor RGB (TCS230)

Primeiramente configuramos o filtro de cor para vermelho (S2 e S3 em LOW conforme a segunda tabela apresentada) e lemos o tempo do pulso com o `pulseIn`, armazenando o resultado na variável inteira `r`. Depois, configuramos o filtro de cor para verde (S2 e S3 em HIGH conforme a segunda tabela apresentada) e lemos o tempo do pulso com o `pulseIn`, armazenando o resultado na variável inteira `G`. Finalmente, configuramos o filtro de cor para azul (S2 em LOW e S3 em HIGH conforme a segunda tabela apresentada) e lemos o tempo do pulso com o `pulseIn`, armazenando o resultado na variável inteira `b`. Para finalizar, escrevemos os valores das 3 variáveis em nossa Serial.

## Ex2: sensor de cor RGB (TCS230)

Com isso, um resultado similar a este deverá ser visualizado através do nosso Monitor Serial:

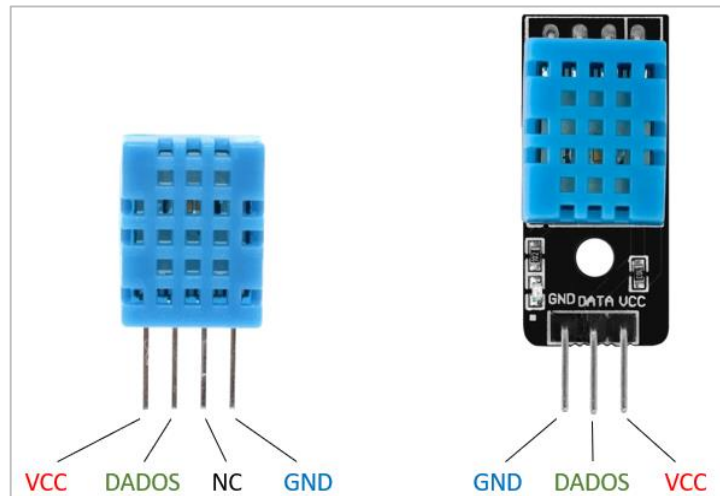


## Ex3: sensor de temp + umidade (DHT11)

Este sensor permite medir temperaturas entre 0°C e 50°C e umidade relativa do ar entre 20% e 95%. Pode ser encontrado de forma avulsa (com 4 pinos) ou encapsulado em placas (geralmente com apenas 3 pinos). Os pinos que nos importam são o VCC (5V), GND e o pino de dados, que deve ser ligado a uma porta digital. Em sua versão avulsa (apenas o sensor), a sequência é sempre, da esquerda para a direita, VCC – DADOS – N.C – GND, onde o N.C não é utilizado. Já quando encapsulado em placas, cada fabricante pode alterar a sequência de pinos, sendo importante a análise.

## Ex3: sensor de temp + umidade (DHT11)

Aqui, mostramos um exemplo onde a sequência de pinos definida pelo fabricante do encapsulamento é GND – DADOS – VCC.





## Ex3: sensor de temp + umidade (DHT11)

Em funcionamento, este sensor informa a umidade e a temperatura através de sequências binárias e para facilitar a captura dessas informações podemos utilizar a biblioteca DTH que contém a implementação de um objeto DTH através do qual podemos chamar dois métodos: `readHumidity` (que nos retorna a umidade relativa do ar) e `readTemperatura` (que nos retorna a temperatura em graus celsius). Ao ser inicializado, o objeto DHT necessita de dois parâmetros: o primeiro é o número da porta digital onde está ligado o pino de dados do sensor e o segundo é qual o tipo de sensor, visto que a mesma biblioteca serve para outros sensores da mesma família (e no nosso caso, utilizamos a constante `DHT11`).



## Ex3: sensor de temp + umidade (DHT11)

Esta biblioteca necessita de uma biblioteca secundária desenvolvida pela Adafruit e utilizada na implementação de diversos sensores, chamada de AdafruitSensor. Porém, em nosso código, apenas a primeira biblioteca necessita de include. A seguir é demonstrado um exemplo de como ler as informações de umidade e temperatura e escrever as mesmas na Serial, com o pino de dados do sensor DHT11 ligado à porta digital 3 do Arduino.

# Ex3: sensor de temp + umidade (DHT11)

```
#include <DHT.h>

DHT dht(3,DHT11);

void setup(){
  Serial.begin(9600);
  dht.begin();
}

void loop(){
  delay(1000);

  int umidade = dht.readHumidity();
  int temperatura = dht.readTemperature();

  Serial.println("Umidade: " + (String)umidade + " porcento");
  Serial.println("Temperatura: " + (String)temperatura + "°C");
}
```

## Ex4: sensor RFID (RC522)

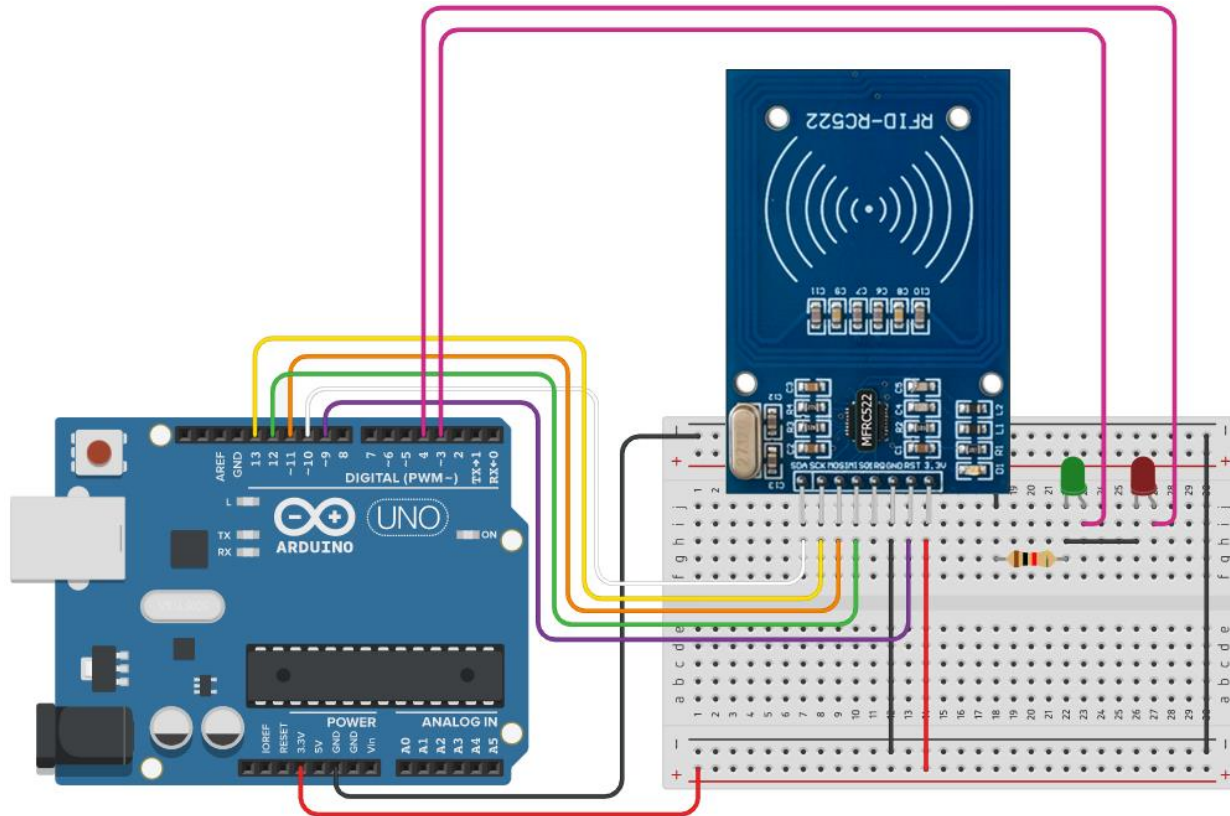
Este sensor permite a leitura do valor de ID (código único de identificação) de tags RFID (etiquetas ativas, etiquetas passivas, cartões e chaveiros). Isso pode ser bastante útil em identificação (de produtos, por exemplo) ou em controle de acesso (de funcionários, por exemplo). Ele pode ser utilizado inclusive para gravar dados dentro das tags graváveis (e um exemplo de como fazer isso pode ser visto no link:

<https://www.filipeflop.com/blog/como-gravar-dados-no-cartao-rfid>

## Ex4: sensor RFID (RC522)

Porém, esta funcionalidade é um pouco mais complexa (avançada) e neste exemplo o que nos interessa é descobrir o ID já gravado nas tags e utilizá-los para controle de acesso. Para entender o uso do sensor RFID, vamos iniciar por seu esquema eletrônico que necessita do uso de uma série de portas (algumas fixas e que não podem ser alteradas). Nesse esquema, foram colocados também dois leds (um verde e um vermelho) como um auxílio visual mostrando se o cartão liberou ou não o acesso.

## Ex4: sensor RFID (RC522)



## Ex4: sensor RFID (RC522)

Os 4 primeiros pinos do sensor RFID, são ligados nos 4 pinos digitais que implementam a comunicação SPI no Arduino e esta sequência é fixa, não podendo ser alterada para que tenhamos o funcionamento desejado. A ligação efetuada está descrita na tabela abaixo:

Pino do sensor	Porta do Arduino
1 - SDA	10
2 - DAS	13
3 - MOSI	11
4 - MISO	12

## Ex4: sensor RFID (RC522)

Além destes, o pino RST do sensor (pino 7) deve ser ligado à uma porta digital PWM que pode ser escolhida (neste exemplo optou pela porta 9), o pino GND ao GND do Arduino e o pino VCC ao 3,3V do Arduino. Muito cuidado aqui, pois este sensor pode ser danificado se submetido à uma tensão de 5V. Para finalizar, o pino positivo do led verde foi ligado à porta digital 3 e o pino positivo do led vermelho foi ligado à porta digital 4.

**OBSERVAÇÃO:** Para este projeto, para o uso do sensor RFID teremos que importar 1 biblioteca não nativa para nossa IDE: MFRC522.



## Ex4: sensor RFID (RC522)

Um exemplo de código para identificação e leitura de nossas tags (ligando os leds indicativos para simular um acesso negado ou permitido) é mostrado a seguir:

```
#include <SPI.h>
#include <MFRC522.h>

//criação do objeto MFR522 informando as portas dos pinos SDA e RST
MFRC522 sensor(10, 9);

//criação de um vetor de tags liberadas para acesso
String tagsLiberadas[] = {"44AB18C9", "AB1515C0"};

void setup(){
    Serial.begin(9600);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);

    //inicialização da comunicação SPI e do objeto que representa o sensor
    SPI.begin();
    sensor.PCD_Init();
    Serial.println("Aproxime a sua TAG RFID do sensor");
}
```

## Ex4: sensor RFID (RC522)

Um exemplo de código para identificação e leitura de nossas tags (ligando os leds indicativos para simular um acesso negado ou permitido) é mostrado a seguir:

```
void loop(){
  //busca novos cartões e quando encontra seleciona um para ser lido
  if( !sensor.PICC_IsNewCardPresent() ) { return; }
  if( !sensor.PICC_ReadCardSerial() ) { return; }

  //mostra o ID da TAG RFID lida
  String idLido = "";
  for(byte i = 0; i < sensor.uid.size; i++) {
    idLido.concat(String(sensor.uid.uidByte[i], HEX));
  }

  Serial.println("ID da TAG lida: " + (String)idLido);

  //compara se o ID da TAG lida é algum dos liberados existentes no vetor
  bool liberado = false;
  for(int i = 0; i < (sizeof(tagsLiberadas)/sizeof(String)); i++) {
    if(idLido.equalsIgnoreCase(tagsLiberadas[i]) ){
      liberado = true;
    }
  }
}
```

## Ex4: sensor RFID (RC522)

Um exemplo de código para identificação e leitura de nossas tags (ligando os leds indicativos para simular um acesso negado ou permitido) é mostrado a seguir:

```
//verifica se a variável liberado mudou para true e libera ou não o acesso
//simulamos isso através de uma mensagem na Serial e dos leds
if(liberado == true){
    Serial.println("Acesso liberado!");
    digitalWrite(3, HIGH);
    digitalWrite(4, LOW);
} else {
    Serial.println("Acesso negado!");
    digitalWrite(3, LOW);
    digitalWrite(4, HIGH);
}
}
```

## Ex4: sensor RFID (RC522)

**Entendendo o código:** o código utilizado é de um nível mais avançado e não cabe aqui explicar de forma aprofundada cada linha e método apresentado. O importante é entender a ideia geral para poder utilizar o sensor RFID nos projetos desejados. Primeiramente criamos um objeto MFRC522, responsável por representar o sensor RFID e oferecer acesso à métodos para trabalhar com ele. Ainda antes do setup, um vetor (array) de Ids permitidos foi criado, já recebendo dois Ids que quando lidos podem liberar o acesso. Quando for reproduzir os seus exemplos, aproxime a TAG desejada do sensor, anote o Id mostrado no monitor serial e acrescente ou substitua os Ids do vetor.

## Ex4: sensor RFID (RC522)

Dentro do setup, após a inicialização da Serial e da definição das portas dos leds como saída (iniciando em estado LOW), a comunicação SPI e o objeto que representa o sensor RFID são inicializados. Já no loop, forçamos que o sensor RFID fique continuamente tentando detectar novas TAG RFID e, quando detectada, que a mesma seja lida. No momento que isso acontece, forçamos um laço de repetição que percorre todo vetor interno do sensor (que armazena o Id lido) e pegamos caractere a caractere hexadecimal para que esse Id lido possa ser mostrado na nossa Serial.

## Ex4: sensor RFID (RC522)

Após, em um novo laço, percorremos o vetor de lds permitidos, comparando um a um dos presentes nele com aquele que acabou de ser lido pelo sensor, para verificar se ele está permitido. Caso esteja, informamos sobre o acesso liberado na Serial e ligamos o led verde. Caso contrário, informamos sobre o acesso negado e ligamos o led vermelho. Esse exemplo poderia ser modificado para, por exemplo, ao invés de ligar um led, ligar um relé pela porta digital que solte ou puxe uma trava elétrica, liberando ou bloqueando uma porta ou portão, por exemplo.

## Ex5: sensor aceler./giroscópio (MPU6050)

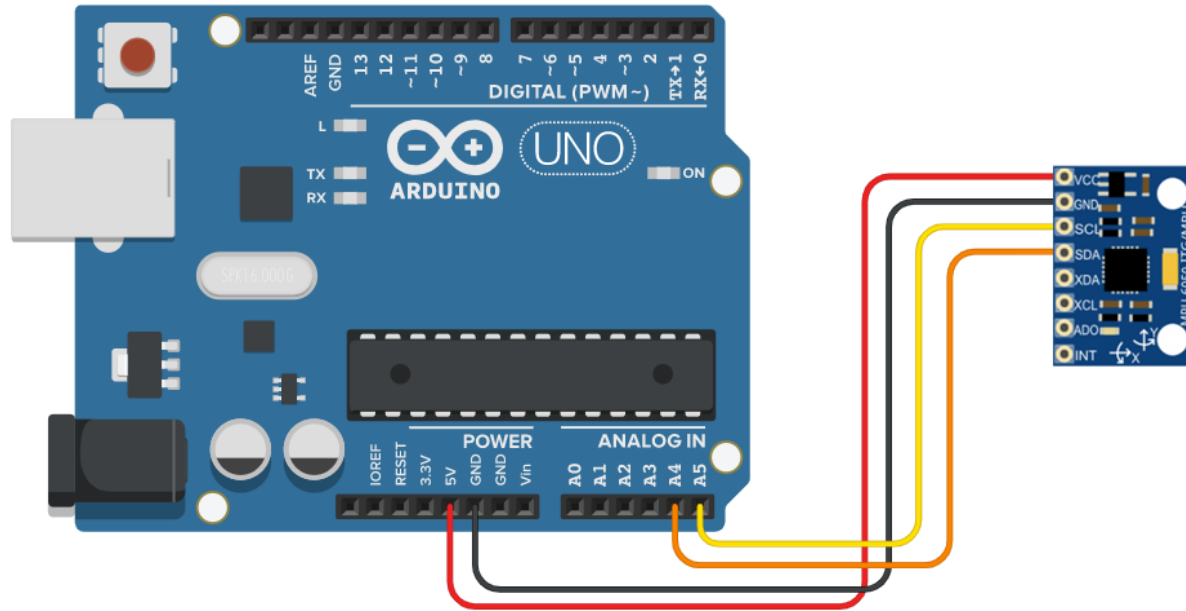
Este tipo de sensor nos permite obter a velocidade e a rotação nos seus 3 eixos (X, Y e Z) enquanto é movimentado. Assim, conseguimos capturar os valores deste movimento e tomar alguma ação ou reproduzi-lo em atuadores. Imagine um braço mecânico que deve acompanhar o mesmo movimento do braço de um cirurgião, ou mesmo um joystick que transforme movimentos em ação no jogo (como faz o Wii).

## Ex5: sensor aceler./giroscópio (MPU6050)

Aqui neste exemplo de uso, por simplificação, não iremos implementar ações aos movimentos detectados, mas mostrar como capturar suas informações para uso futuro. Este sensor possui 8 pinos, mas na prática utilizamos apenas os 4 primeiros: em ordem, VCC, GND, SCL (que obrigatoriamente deve ser conectado à porta A5) e SDA (que obrigatoriamente deve ser conectado à porta A4). Assim, teremos o seguinte esquema eletrônico:



## Ex5: sensor aceler./giroscópio (MPU6050)



## Ex5: sensor aceler./giroscópio (MPU6050)

Um exemplo de código para a leitura e apresentação das velocidades e giros nos 3 eixos do acelerômetro/giroscópio é mostrado a seguir:

```
#include <Wire.h>

//criação uma constante para armazenar o endereço I2C do sensor
const int MPU = 0x68;

void setup(){
  Serial.begin(9600);

  //inicialização da comunicação I2C (Wire) e do sensor
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
}
```

## Ex5: sensor aceler./giroscópio (MPU6050)

Um exemplo de código para a leitura e apresentação das velocidades e giros nos 3 eixos do acelerômetro/giroscópio é mostrado a seguir:

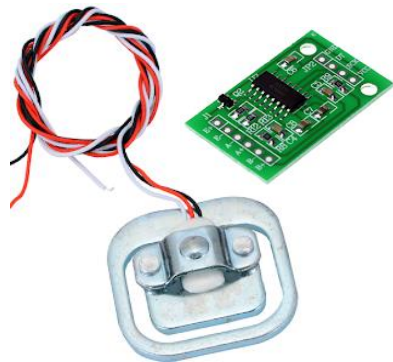
```
void loop(){
    delay(10);

    //solicitação dos dados do sensor
    Wire.beginTransmission(MPU);
    Wire.write(0x3B);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU, 14, true);

    //leitura dos valores e armazenamento em variáveis
    inc aX = Wire.read()<<8|Wire.read();
    inc aY = Wire.read()<<8|Wire.read();
    inc aZ = Wire.read()<<8|Wire.read();
    inc tmp = Wire.read()<<8|Wire.read();
    inc gX = Wire.read()<<8|Wire.read();
    inc gY = Wire.read()<<8|Wire.read();
    inc gZ = Wire.read()<<8|Wire.read();

    Serial.print("ACEL|X:" + (String)aX + ",Y:" + (String)aY + ",Z:" + (String)aZ + "|");
    Serial.print("  GIRO |X:" + (String)gX + ",Y:" + (String)gY + ",Z:" + (String)gZ + "|");
    Serial.println("  TEMP |" + (String)tmp + "|");
}
```

## Ex6: sensor de carga (HX711)

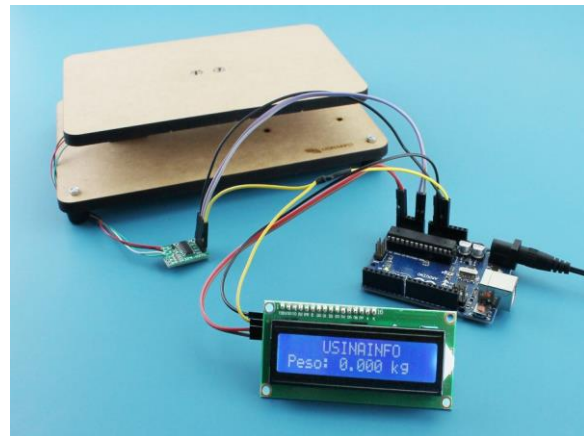


O sensor de carga (ou célula de carga) é uma espécie de sensor piezo que possui duas placas que, ao serem comprimidas, provocam micro correntes (mais fortes conforme mais força é aplicada). Deste modo, estes sensores podem ser utilizados como balança, fazendo uma relação entre a corrente medida (e amplificada pelo HX711 para poder ser percebida pelas placas de prototipagem) e o peso necessário para produzi-la.

## Ex6: sensor de carga (HX711)

Para entender o seu funcionamento, acesse o link abaixo para visualizar um exemplo completo desenvolvido por Matheus Gebert Straub no site da UsinaInfo, explicando o passo a passo da construção de uma balança digital caseira para até 5Kg.

<https://www.usinainfo.com.br/blog/balanca-arduino-com-celula-de-peso-e-hx711-tutorial-calibrando-e-verificando-peso/>



## Ex7: sensor de gestos (APDS-9960)



Os sensores de gestos são capazes de detectar quando realizamos determinados movimentos próximos deles, com as mãos ou os dedos. Os sensores mais comuns (como o APDS-9960) detectam o movimento da mão de cima para baixo, de baixo para cima, da esquerda para a direita e da direita para a esquerda. Alguns sensores mais sofisticados detectam movimentos especiais como, por exemplo, a mão afastando ou aproximando ou mesmo gestos circulares.

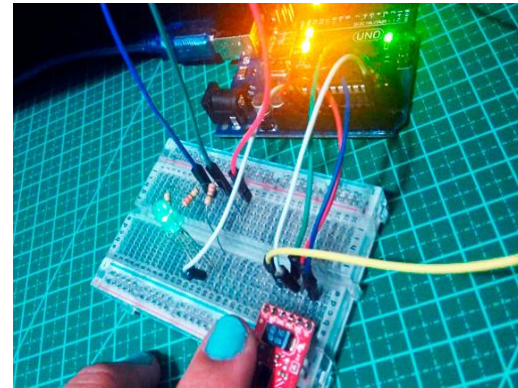


## Ex7: sensor de gestos (APDS-9960)

Detectando esses movimentos, podemos programar o Arduino para executar determinadas ações para cada um (imagine controlar uma TV da cama, podendo trocar o canal, ajustar o volume e ligar e desligar ela, apenas movimentando a mão). Apesar do APDS-9960 detectar apenas os 4 movimentos básicos citados, ele tem mais uma funcionalidade extra: funciona também como um detector de cores RGB.

## Ex7: sensor de gestos (APDS-9960)

Para entender o seu funcionamento, acesse os dois links abaixo. No primeiro, Gedeane Kenshima, do site FelipeFlop, mostra como usar o APDS-9960 para detectar cores RGB e transferir a cor detectada para um led RGB. No segundo, Euler Oliveira, do blog da Master Walker Shop, mostra como detectar os 4 gestos básicos com o sensor.



<https://www.filipeflop.com/blog/como-utilizar-o-sensor-de-gestos-e-rgb-sparkfun/>

<https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-sensor-de-gestos-e-de-cor-apds-9960>



## Ex8: sensor de reconhecimento de voz (V3)



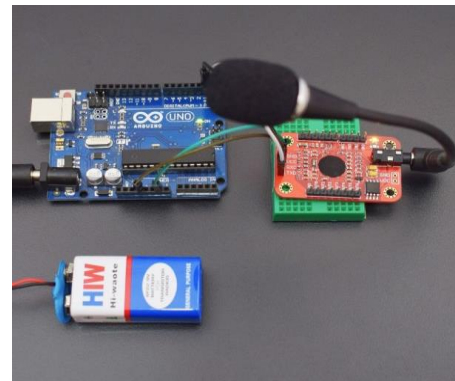
A integração de reconhecimento de voz no Arduino é muito interessante e nos permite controlar as suas ações apenas falando, com o uso de comandos de voz. Geralmente para isso, precisamos criar um aplicativo para smartphone que é o responsável por capturar a voz, reconhecer os comandos e mandar ordens ao Arduino através de uma conexão serial (por bluetooth, wifi ou mesmo cabo USB). Porém, isso acrescenta um grau de complexidade ao nosso projeto. O sensor de reconhecimento de voz V3 simplifica este processo.

## Ex8: sensor de reconhecimento de voz (V3)

Ele é capaz de armazenar internamente até 80 palavras diferentes, narradas pelo usuário através de um programa de treinamento. Destas 80, 7 podem ser ativadas por vez e caso alguma delas seja detectada, o sensor avisa ao Arduino enviando esta palavra e a placa, por sua vez, pode tomar uma ação. Assim, conseguimos desenvolver sistemas que sejam totalmente controlados por voz, sem a necessidade de apps adicionais, de smartphone e mesmo de internet.

## Ex8: sensor de reconhecimento de voz (V3)

Para entender melhor o seu funcionamento, entre no link abaixo para visualizar o exemplo desenvolvido por Matheus Gebert Straub do site UsinaInfo, explicando deste o treinamento das palavras no sensor até a detecção destas palavras pelo Arduino.



<https://www.usinainfo.com.br/blog/projeto-reconhecimento-de-voz-arduino/>