

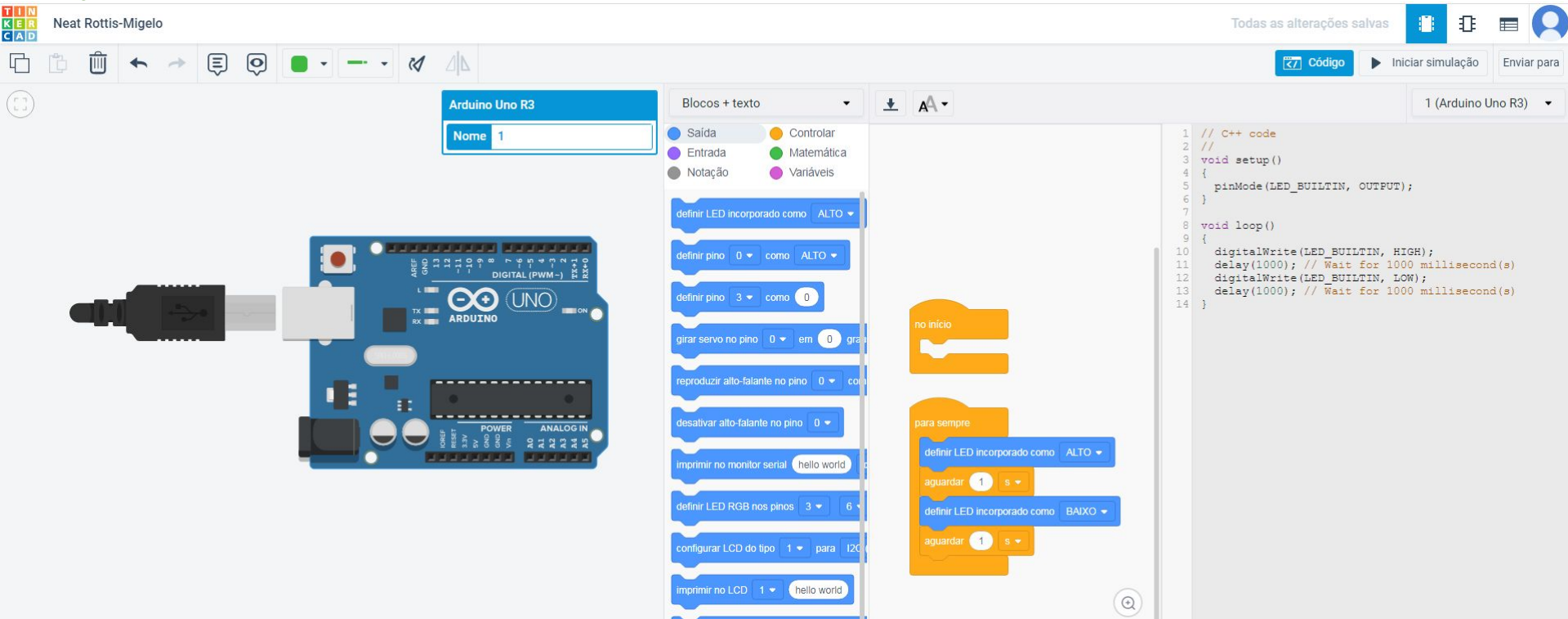
Laboratórios de Arduíno

Prof^a: Luciana Menezes Xavier de Souza
Prof^a: Maria Cláudia de Almeida Castro

Piscando o LED do Arduino

Exemplo

Escolher o Arduíno UNO do TinkerCad, clicar em código e executar (iniciar a simulação) o programa padrão para piscar o led do pino 13 (LED_BUILTIN), mantendo aceso por 1 seg e em seguida apagado por 1 seg. O led do pino 13 é um led instalado na própria placa do Arduíno. O modo de edição pode ser Blocos + Texto ou apenas Texto



The screenshot displays the TinkerCad web interface. On the left, a 3D model of an Arduino Uno R3 board is shown with a USB cable connected. The top bar includes the TinkerCad logo, the user name 'Neat Rottis-Migelo', and various tool icons. The main workspace is divided into three panels:

- Left Panel:** A list of components under the heading 'Blocos + texto'. It includes categories like 'Saída' (Output), 'Entrada' (Input), 'Notação' (Notation), 'Controlar' (Control), 'Matemática' (Mathematics), and 'Variáveis' (Variables). Specific blocks visible include 'definir LED incorporado como' (set built-in LED as), 'definir pino' (define pin), 'gitar servo no pino' (rotate servo on pin), 'reproduzir alto-falante no pino' (play speaker on pin), 'desativar alto-falante no pino' (disable speaker on pin), 'imprimir no monitor serial' (print to serial monitor), 'definir LED RGB nos pinos' (define RGB LED on pins), 'configurar LCD do tipo' (configure LCD type), and 'imprimir no LCD' (print on LCD).
- Middle Panel:** A block-based programming editor. It shows a 'no início' (when started) block followed by a 'para sempre' (forever) loop. Inside the loop, there are blocks to 'definir LED incorporado como' (set built-in LED as) 'ALTO' (HIGH), 'aguardar' (wait) for 1 second, 'definir LED incorporado como' (set built-in LED as) 'BAIXO' (LOW), and another 'aguardar' (wait) for 1 second.
- Right Panel:** A C++ code editor showing the equivalent code for the block-based program:

```
1 // C++ code
2 //
3 void setup()
4 {
5   pinMode(LED_BUILTIN, OUTPUT);
6 }
7
8 void loop()
9 {
10  digitalWrite(LED_BUILTIN, HIGH);
11  delay(1000); // Wait for 1000 millisecond(s)
12  digitalWrite(LED_BUILTIN, LOW);
13  delay(1000); // Wait for 1000 millisecond(s)
14 }
```

The top right corner of the interface shows the status 'Todas as alterações salvas' (All changes saved) and buttons for 'Iniciar simulação' (Start simulation) and 'Enviar para' (Send to).

Piscando o LED do Arduino - Estrutura do programa

EXERCÍCIO 1

Modificar o tempo (delay) e verificar a mudança na frequência do piscar do led.

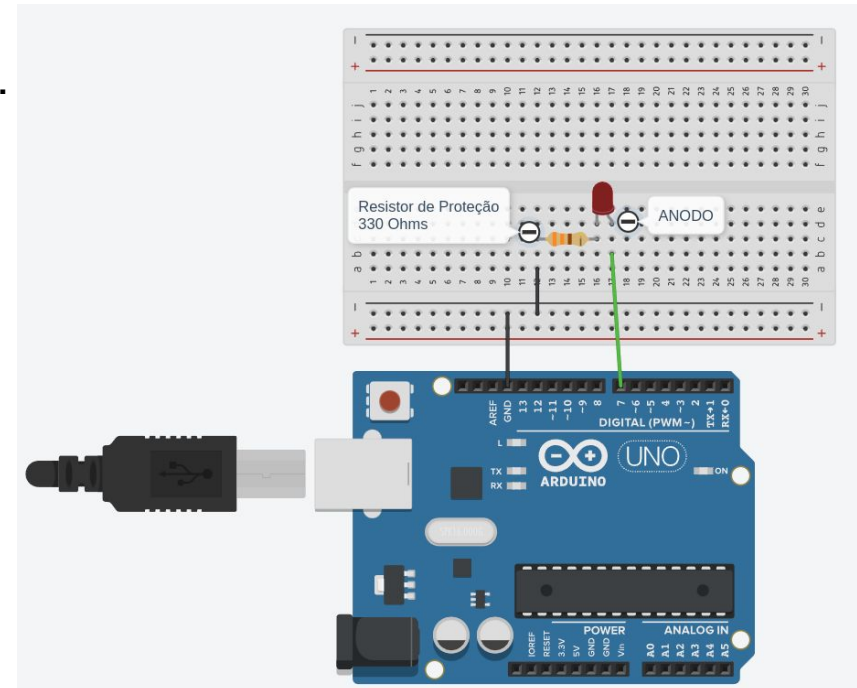
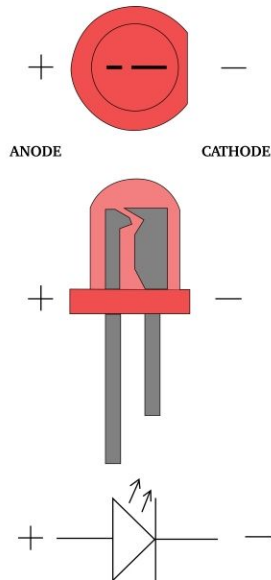
A função `delay(1000);` estabelece um atraso em ms (mili segundos), logo $1000 \text{ ms} = 1 \text{ s}$.

```
1 //Programa : Pisca Led Arduino
2 //Autor : FILIPEFLOP
3
4 void setup()
5 {
6     //Define a porta do led como saida
7     pinMode(13, OUTPUT);
8 }
9
10 void loop()
11 {
12     //Acende o led
13     digitalWrite(13, HIGH);
14
15     //Aguarda o intervalo especificado
16     delay(1000);
17
18     //Apaga o led
19     digitalWrite(13, LOW);
20
21     //Aguarda o intervalo especificado
22     delay(1000);
23 }
```

Piscando o LED com Arduino

EXERCÍCIO 2

1. Modificar programa anterior para piscar um LED montado na matriz de contato (placas de ensaio), **não esqueça de utilizar um resistor de proteção adequado de 330 Ohms.**
2. Utilize agora a **saída digital 7.**
3. Compile e carregue no arduino.

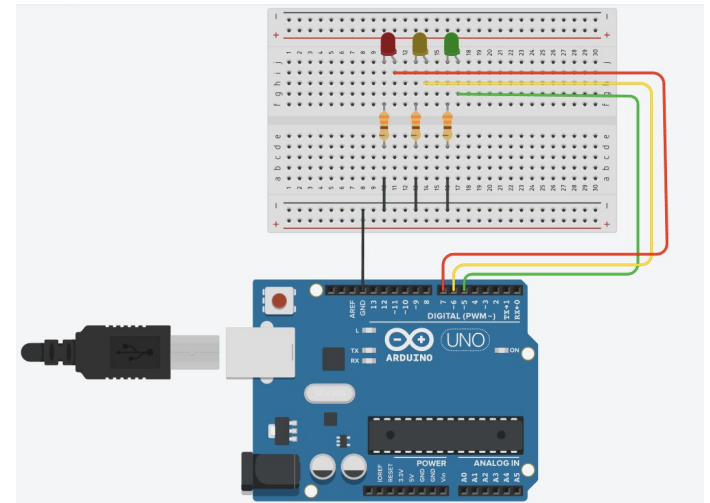


Semáforo automatizado com leds



EXERCÍCIO 3

1. Montar *hardware* para acionamento de 3 leds (**vermelho, verde e amarelo**), **comandados por 3 saídas digitais independentes** do Arduino, utilize leds independentes utilizando resistores de proteção adequados (330 Ohms para cada resistor).
2. Modificar programa anterior para acionar os leds como um semáforo, com temporização adequada para cada cor.
3. Compile e carregue no arduino.

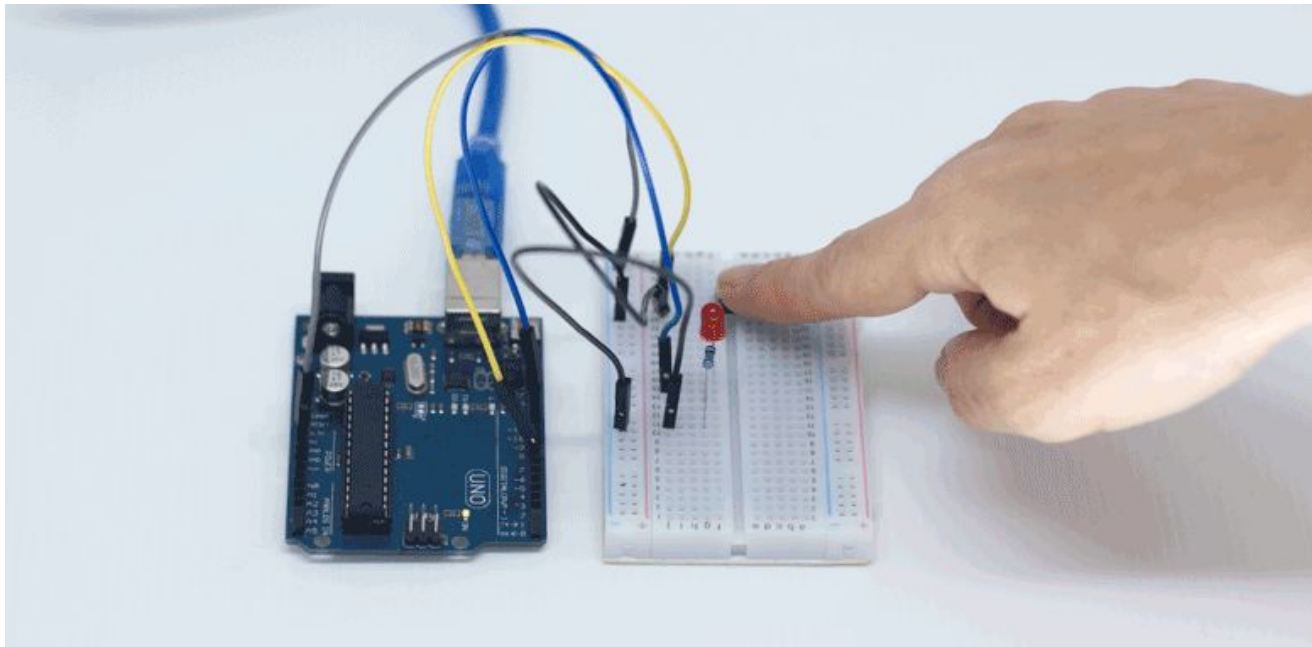


Acionar um LED através de uma chave push-button (botão)



EXERCÍCIO 4: Neste projeto, veremos como funcionam os **sensores**. **Entrada** de dados através de pinos do arduino e não só saída como eram os projetos até aqui.

Objetivo: verificar como o Arduino pode receber ou ler informações externas e tomar ações a partir deles.

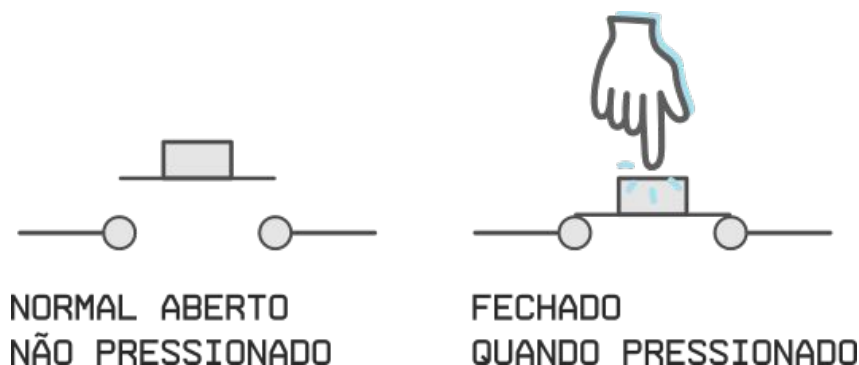


O que é push-button (botão)?



Uma chave push-button funciona como um contato que abre e fecha (0 ou 1).

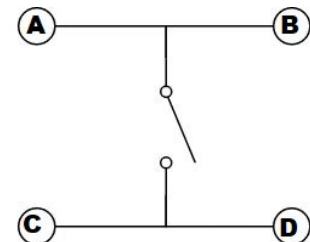
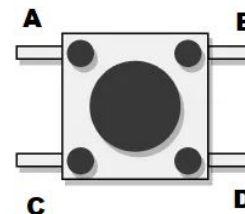
Conectando uma chave a uma porta do Arduino podemos ler o valor 0 ou 1 da chave e assim tomar uma ação, que no caso do nosso exemplo será acionar o LED.



O botão, quando pressionado, faz contato entre um lado e outro dele. Quando esse contato é fechado, essa corrente elétrica “entra” na placa e ela percebe que o botão foi pressionado. Ao escrevermos o programa, decidimos o que fazer com essa informação.

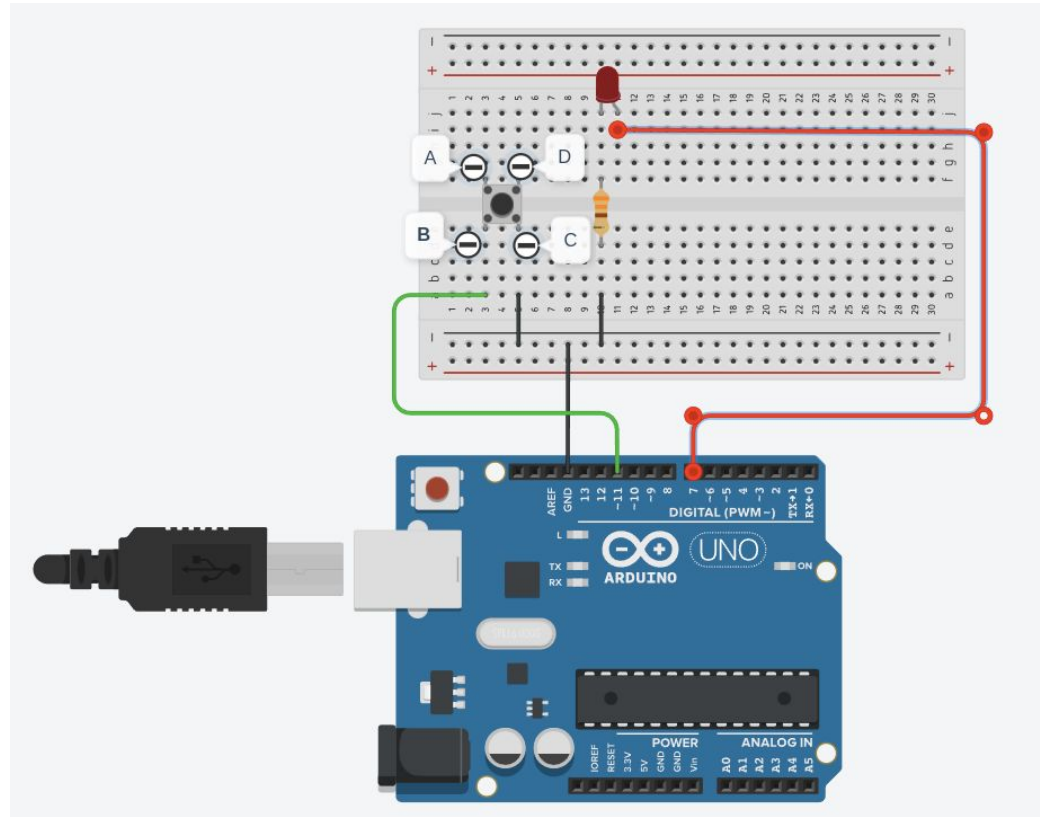
Como conectar a chave push-button

1. A maioria das chaves possuem **2 terminais**: Um terminal da chave vai no pino do arduino e outro no GND.
2. A proteção é dada através de um resistor interno do Arduino. Para isto, basta habilitar o resistor pull-up interno do Arduino através da programação.
3. A chave disponibilizada possui **4 terminais**. Este tipo de botão não pode ser ligado de qualquer forma, tem os pinos certos que devem ser conectados, observe a imagem abaixo e vamos entender o seu funcionamento.



Montagem para acionar um LED através de uma chave push-button (botão)

1. Os terminais A e B estão em curto, assim como os C e D.
2. Conecte o pino de entrada do arduino no terminal A ou B e conecte os pinos C ou D no GND.
3. O pino do LED é uma saída digital e o pino da chave é uma entrada digital.



Código para acionar um LED através de uma chave push-button (botão)

// Exemplo 2

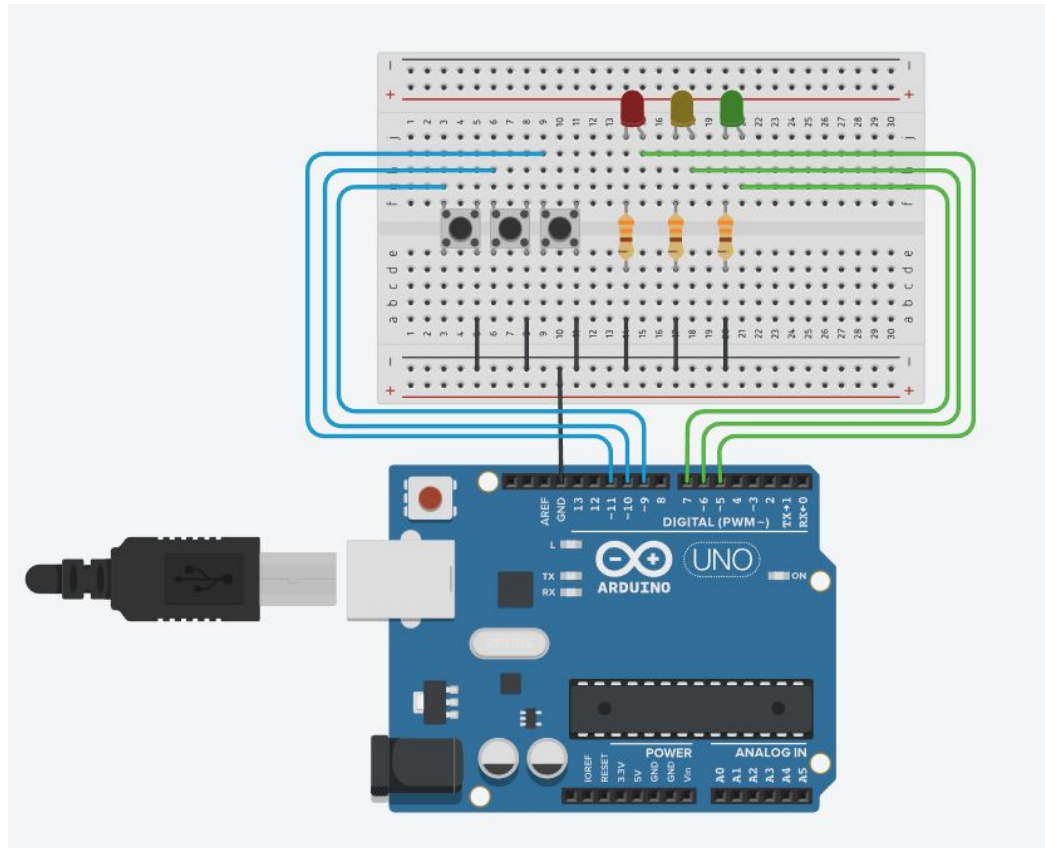
```
const int ledPin = 7; //led no pino 7
const int Botao = 11; //botao no pino 11
int EstadoBotao = 0; //Variavel para ler o status do pushbutton

void setup(){
  pinMode(ledPin, OUTPUT); //Pino do led será saída
  pinMode(Botao, INPUT_PULLUP); //Pino com botão será entrada
}

void loop(){
  EstadoBotao = digitalRead(Botao); /*novo estado do botão vai ser igual ao que
  Arduino ler no pino onde está o botão.
  Poderá ser ALTO (HIGH)se o botão estiver solto
  ou BAIXO (LOW),se o botão
  estiver Pressionado */
  if (EstadoBotao == LOW){ //Se botão estiver pressionado (LOW)
    digitalWrite(ledPin, HIGH); // acende o led do pino 5.
  }
  else{ //se não estiver pressionado
    digitalWrite(ledPin, LOW); //deixa o led do pino 5 apagado
  }
}
```

Acionar vários LEDs através de chaves push-button

EXERCÍCIO 5 - Tente modificar o código ANTERIOR para utilizar 3 LED's e 3 botões. Cada botão deve acionar um dos LEDS.



Brincando com sons



Como reproduzir sons no arduino utilizando um buzzer.



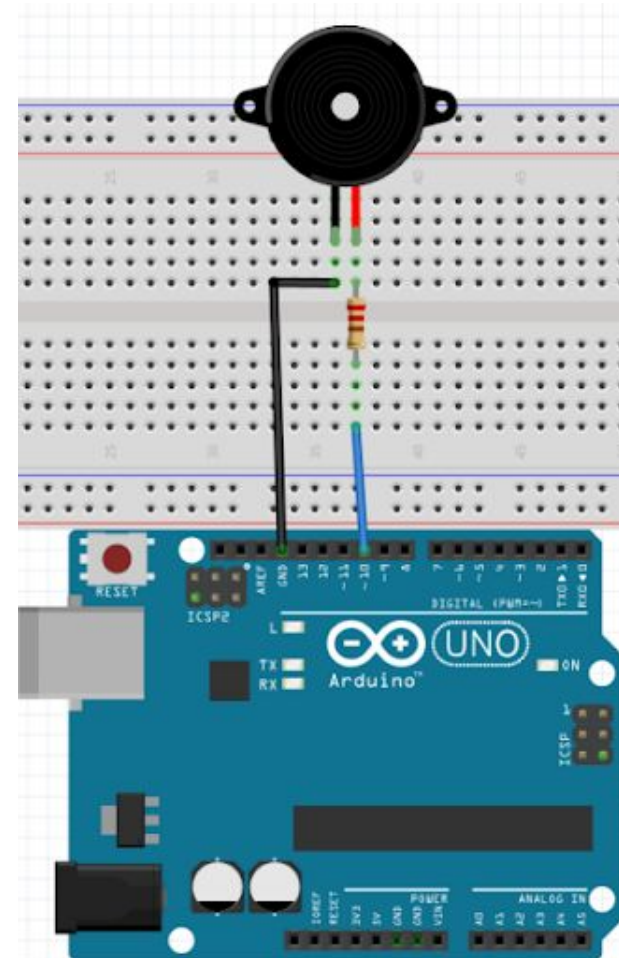
O que é Buzzer?

Nada mais é do que um transdutor piezoelétrico encapsulado. Quando se aplica um sinal elétrico e em determinada frequência, o dispositivo piezoelétrico produz uma nota musical. As notas variam de acordo com a frequência utilizada. (O ouvido humano pode ouvir sons nas frequências entre 20 e 20.000 Hz).



Como conectar o buzzer

- É necessário resistor de proteção (**330 ohms**).
- O buzzer tem polaridade. Portanto, cuidado para não ligar o buzzer invertido. Se você retirar o adesivo superior do buzzer poderá ver um sinal de positivo (+). Esse será o terminal ligado ao resistor e ao pin do arduíno.



Ligando um buzzer

EXERCÍCIO 6

/*

Projeto Arduino beep com buzzer.

//Constante que representa o pino onde o positivo

//do buzzer será ligado.

const int buzzer = 10;

//Método setup, executado uma vez ao ligar o Arduino.

void setup() {

 //Definindo o pino buzzer como de saída.

 pinMode(buzzer,OUTPUT);

}

//Método loop, executado enquanto o Arduino estiver ligado.

void loop() {

 //Ligando o buzzer com uma frequencia de 1500 hz.

 tone(buzzer,1500);

 delay(500);

 //Desligando o buzzer.

 noTone(buzzer);

 delay(500);

}

Ligando um buzzer

EXERCÍCIO 7

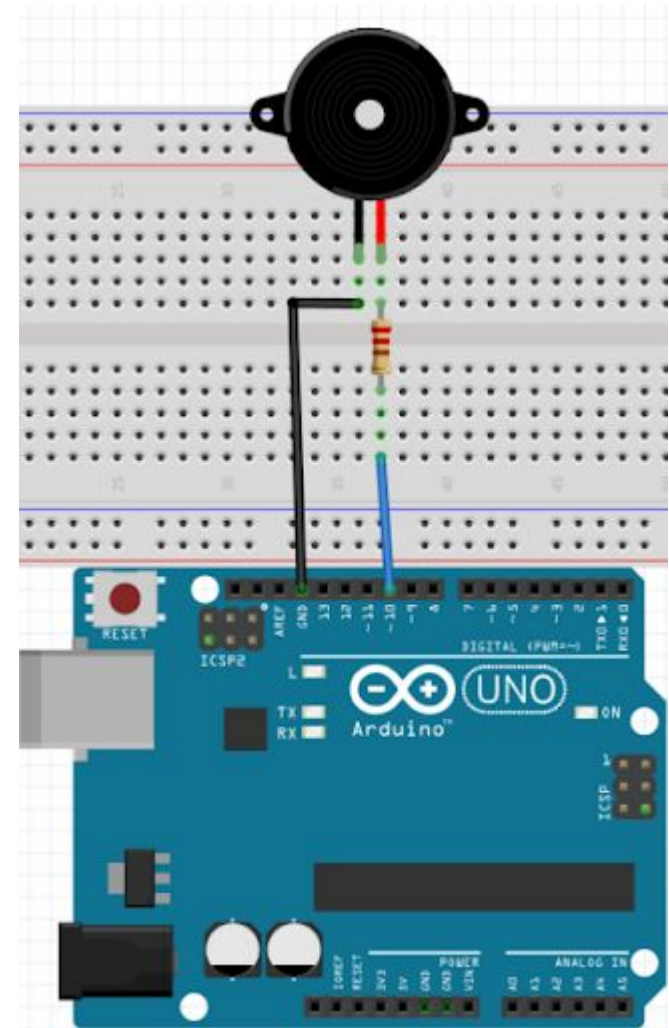
Tente modificar o código ANTERIOR utilizando **LED verde** para sinalizar que o **buzzer** está em funcionamento e o **LED amarelo** quando esse estiver no **período de intervalo**.

Brincando com sons

EXERCÍCIO 8

Com o arduíno desconectado:

1. Monte o *hardware* do circuito em sua matriz de contato: **Buzzer utilizando resistores de proteção adequado.**
2. Baixe no sigaa e analise o código “**Musica_cai_cai_balao.ino**”. A função “tone” é pré-definida pelo arduíno, ela reproduzirá uma determinada frequência por um tempo pré-definido. O código tocará a música programada.
3. Conecte o buzzer, compile o código, verifique o seu funcionamento. Altere o tempo da nota e perceba o que alterou.

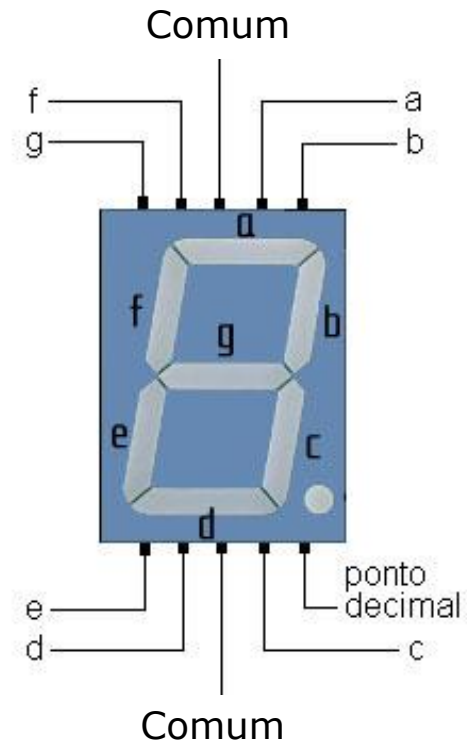


Display 7 segmentos

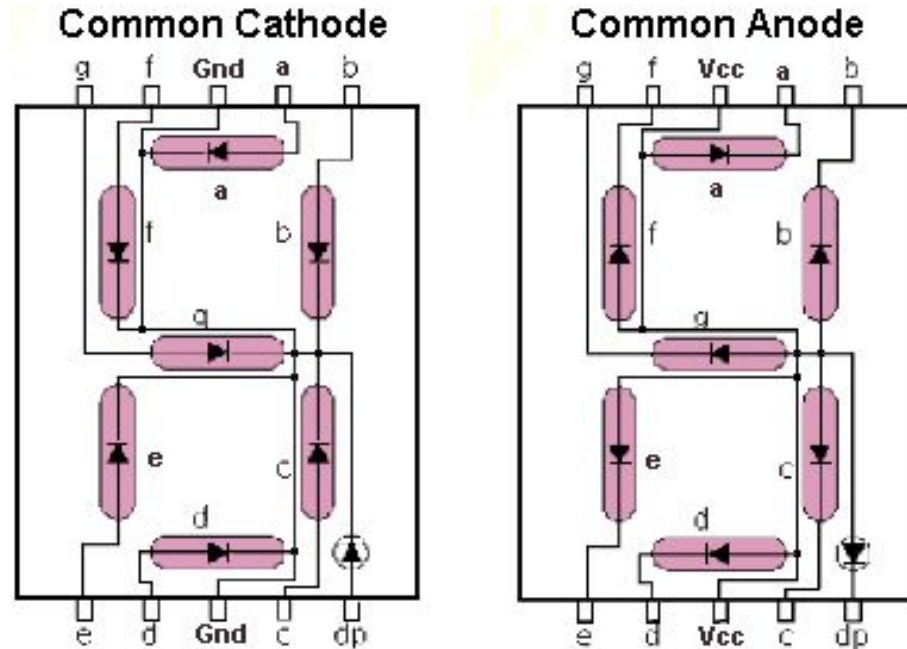
Display de 7 segmentos



Segmentos e terminais



Display Catodo e Anodo Comum



Para acender, normalmente o *display* necessita de uma corrente entre 10 e 20 *mA*, o que provoca uma queda de tensão da ordem de 1,2 V. Desta forma, trabalhando-se com 5 Volts de alimentação, é comum utilizarmos um resistor de 330 Ω para cada segmento visando atingir estes valores.

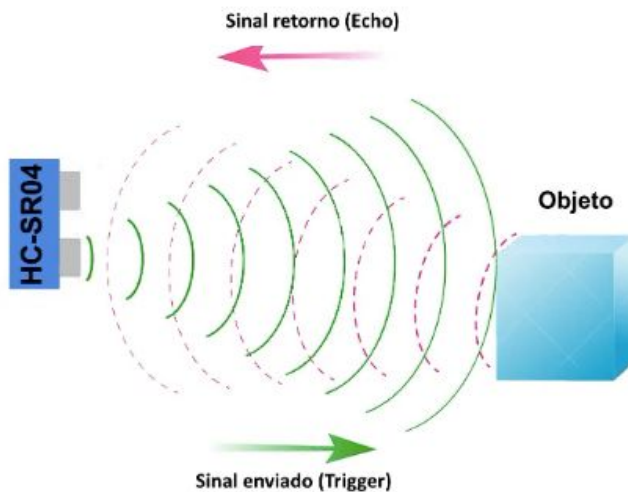
Display de 7 segmentos Cátodo comum

EXERCÍCIO 9

Carregar no Arduíno o programa exemplo Arquivo/Exemplos/Basic/Blink e alterá-lo para que o display de 7 segmentos (**7 leds independentes**) efetuem uma contagem de 0 até 9.

Sensor de Distância Ultrassônico HC-SR04

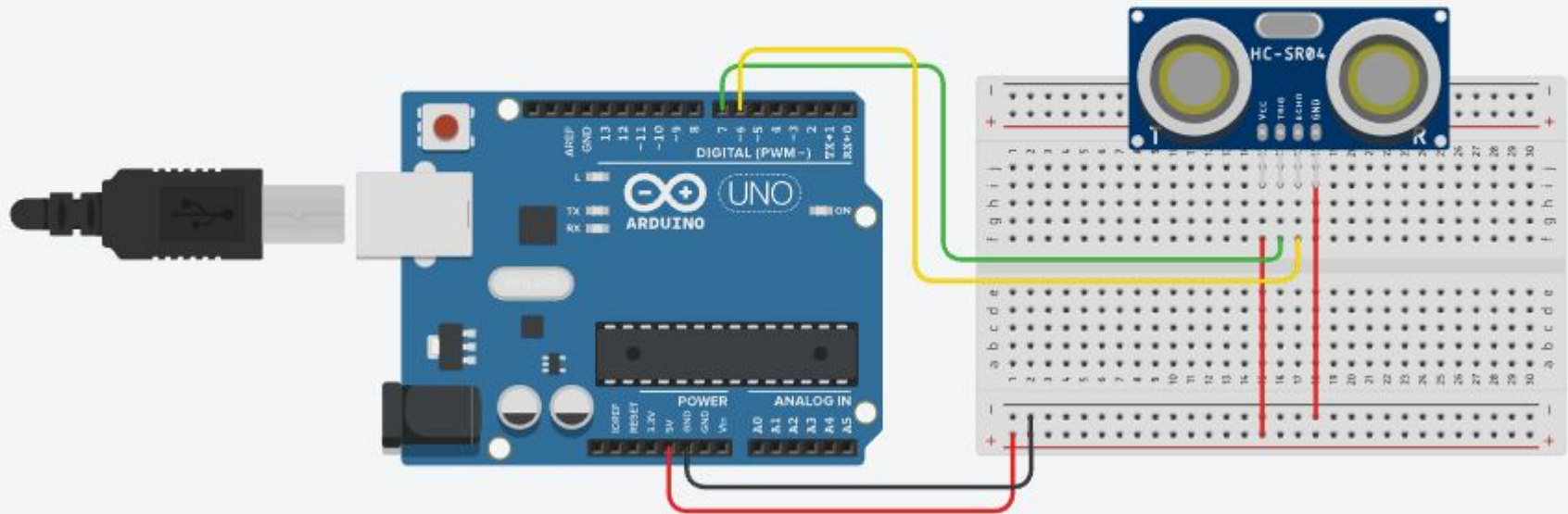
O Sensor Ultrassônico é um componente muito comum em projetos com Arduino, e permite que você faça **leituras de distâncias entre 2 cm e 4 metros**, com precisão de 3 mm. Pode ser utilizado simplesmente para medir a distância entre o sensor e um objeto, como para desviar um robô de obstáculos, acionar alarmes, etc.



O funcionamento do HC-SR04 (datasheet) se baseia no envio de sinais ultrassônicos pelo sensor, que aguarda o retorno (echo) do sinal, e com base no tempo entre envio e retorno, calcula a distância entre o sensor e o objeto detectado.



Conectando o Sensor HC-SR04 ao Arduino



Código para operar o Sensor HC-SR04

```
int SonarTrigger = 7; // define o pino 7 como Trigger
int SonarEcho = 6; // define o pino 6 como Echo

int distancia = 0; // Variável utilizada para calcular a distância em cm
int tempo = 0; // Tempo medido pelo sensor em microsegundos

void setup() { // Definindo os pinos de entrada e saída

    pinMode(SonarTrigger, OUTPUT); // Função de emitir o som (altofalante - Saída)
    pinMode(SonarEcho, INPUT); // Função de receber um som (microfone - Entrada)
    Serial.begin(9600); // Para escrever no monitor serial

}

void loop() {

    digitalWrite(SonarTrigger, LOW); // Estabiliza sensor
    delay(2);

    digitalWrite(SonarTrigger, HIGH); // Envia um pulso para ativar
    delay(10);
    digitalWrite(SonarTrigger, LOW);

    tempo = pulseIn(SonarEcho, HIGH); // Função que recebe um pulso e retorna o valor em tempo da duração deste pulso, em microsegundos
    distancia = tempo/58.2; // Distancia = tempo * velocidade do som / 2 (Velocidade do som é 340 m/s)

    Serial.print("Distancia medida: ");
    Serial.print(distancia);
    Serial.println("cm");

    delay(1000); // Mede a cada 1 segundo

}
```

Operando o Sonar

EXERCÍCIO 10

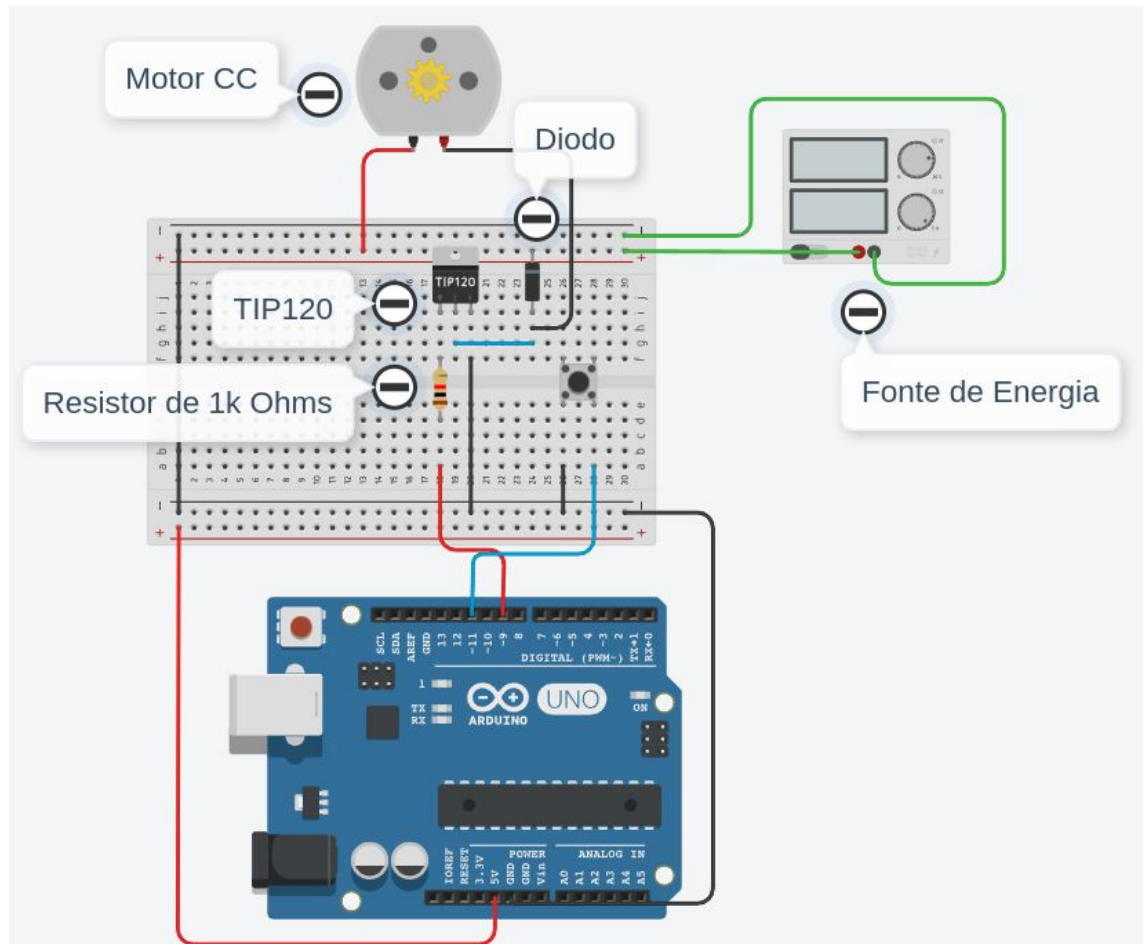
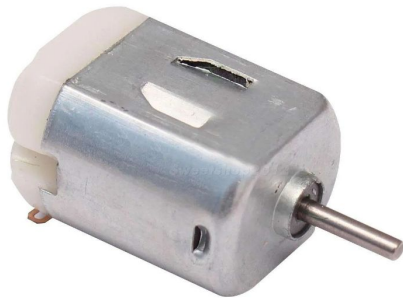
Altere o programa anterior de forma que quando o sonar estiver a uma distância menor do que 100 cm acenda um led vermelho, caso contrário acenda um led verde. Lembre de apagar os LEDS.

DICA:

```
if (distancia < 100) { //Verifica se a distância é menor do que 100 cm
    Coloque o comando para Acender o LED VERMELHO;
} else {
    Coloque o comando para Acender o LED VERDE;
}
```

Uso de Motor de Corrente Contínua (motor CC)

Motor de Corrente Contínua (motor CC), como o nome supõe, é um motor acionado por uma fonte de alimentação de corrente contínua.



Código para operar o Motor CC com botão

```
const int Motor = 9; //motor no pino 9
const int Botao = 11; //botao no pino 11
int EstadoBotao = 0; //Variavel para ler o status do pushbutton

void setup(){
  pinMode(Motor, OUTPUT); //Pino do led será saída
  pinMode(Botao, INPUT_PULLUP); //Pino com botão será entrada
}

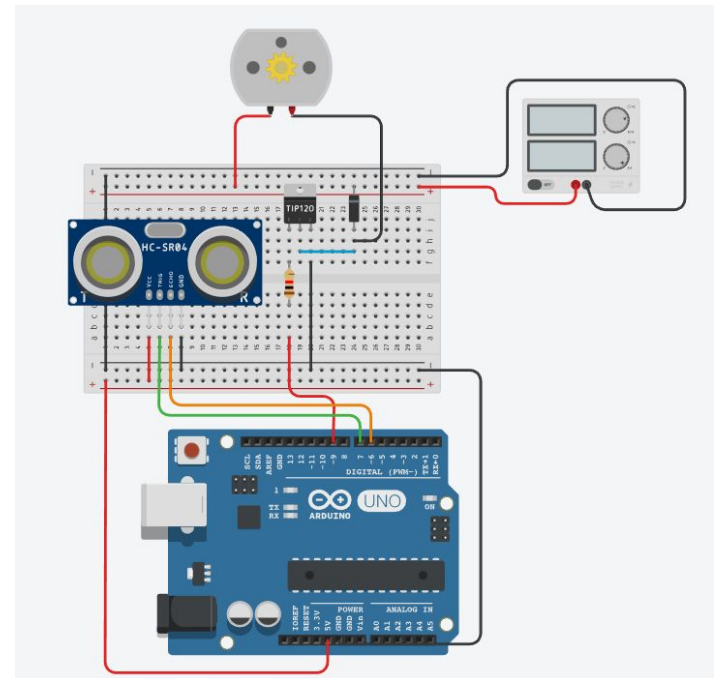
void loop(){
  EstadoBotao = digitalRead(Botao); /*novo estado do botão vai ser igual ao que
  Arduino ler no pino onde está o botão.
  Poderá ser ALTO (HIGH)se o botão estiver solto
  ou BAIXO (LOW),se o botão
  estiver Pressionado */
  if (EstadoBotao == LOW){ //Se botão estiver pressionado (LOW)
    digitalWrite(Motor, HIGH); // acende o led do pino 9.
  }
  else{ //se não estiver pressionado
    digitalWrite(Motor, LOW); //deixa o led do pino 9 apagado
  }
}
```

Controlando um Motor CC com um sonar

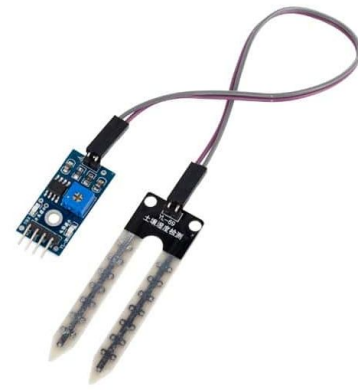
EXERCÍCIO 11

Altere o código anterior para que o **motor CC** seja acionado quando o sonar detectar uma distância menor do que 100 cm.

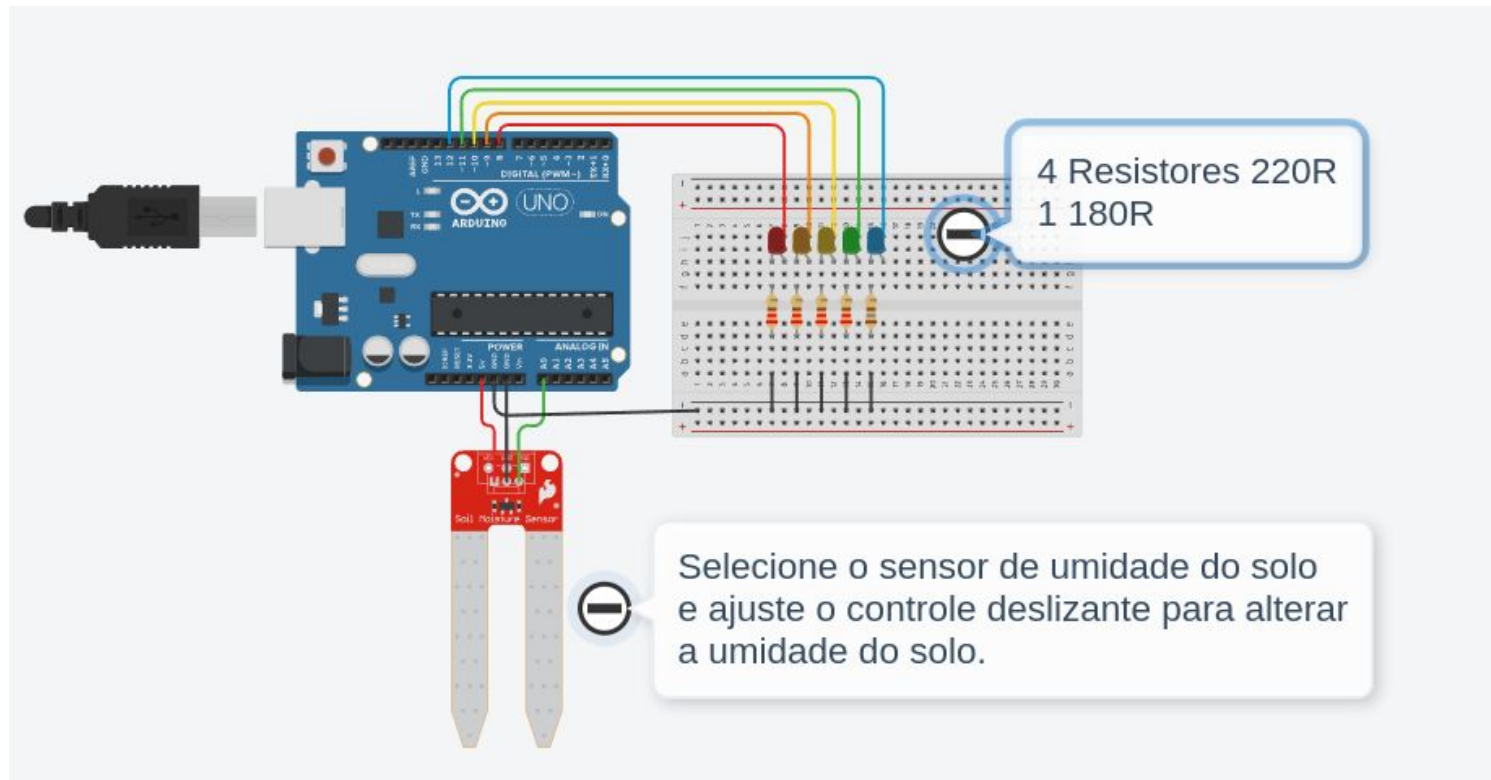
Esse exercício simulará o acionamento do motor



Sensor de umidade do Solo



Um Sensor de Umidade do Solo é um módulo detector da resistividade da terra, ou seja, são sensores que medem as variações de umidade do solo.



Controlando o Sensor de Umidade de Solo

Exercício 12 - Observando o funcionamento de um sensor de umidade do solo

//Exercício 12

int umidade = 0;

```
void setup()
{
  Serial.begin(9600);
  pinMode(A0, INPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
}
```

```
void loop()
{
  umidade = analogRead(A0);
  Serial.println(umidade);
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
  digitalWrite(11, LOW);
  digitalWrite(12, LOW);
  if (umidade < 200) {
    digitalWrite(8, HIGH);
  } else {
    if (umidade < 400) {
      digitalWrite(9, HIGH);
    } else {
      if (umidade < 600) {
        digitalWrite(10, HIGH);
      } else {
        if (umidade < 800) {
          digitalWrite(11, HIGH);
        } else {
          digitalWrite(12, HIGH);
        }
      }
    }
  }
  delay(100); // Espere por 100 milisegundos
}
```