
ARDUINO

Sumário

1 – Introdução ao Arduino

Bem-vindo ao curso Kit Maker Arduino Iniciante!

Este curso possui 50 aulas divididas em 15 módulos para garantir uma experiência de aprendizado completa para quem está iniciando nos estudos. Além de um módulo de Projetos Extras que você vai aplicar os conhecimentos aprendidos no curso e combinar o conteúdo dos módulos abordando diferentes temas. No Kit Arduino estão incluídos diversos sensores, acessórios e até peças em impressão 3D, além de uma Placa Arduino Uno R3 SMD (item opcional do Kit).

Durante os módulos do curso você irá aprender como utilizar todos os componentes em conjunto com o Arduino, aprendendo conceitos básicos de programação e a utilizar a plataforma da IDE. Aprenda mais sobre os componentes, sobre o kit, conceitos básicos de programação e monte projetos incríveis!

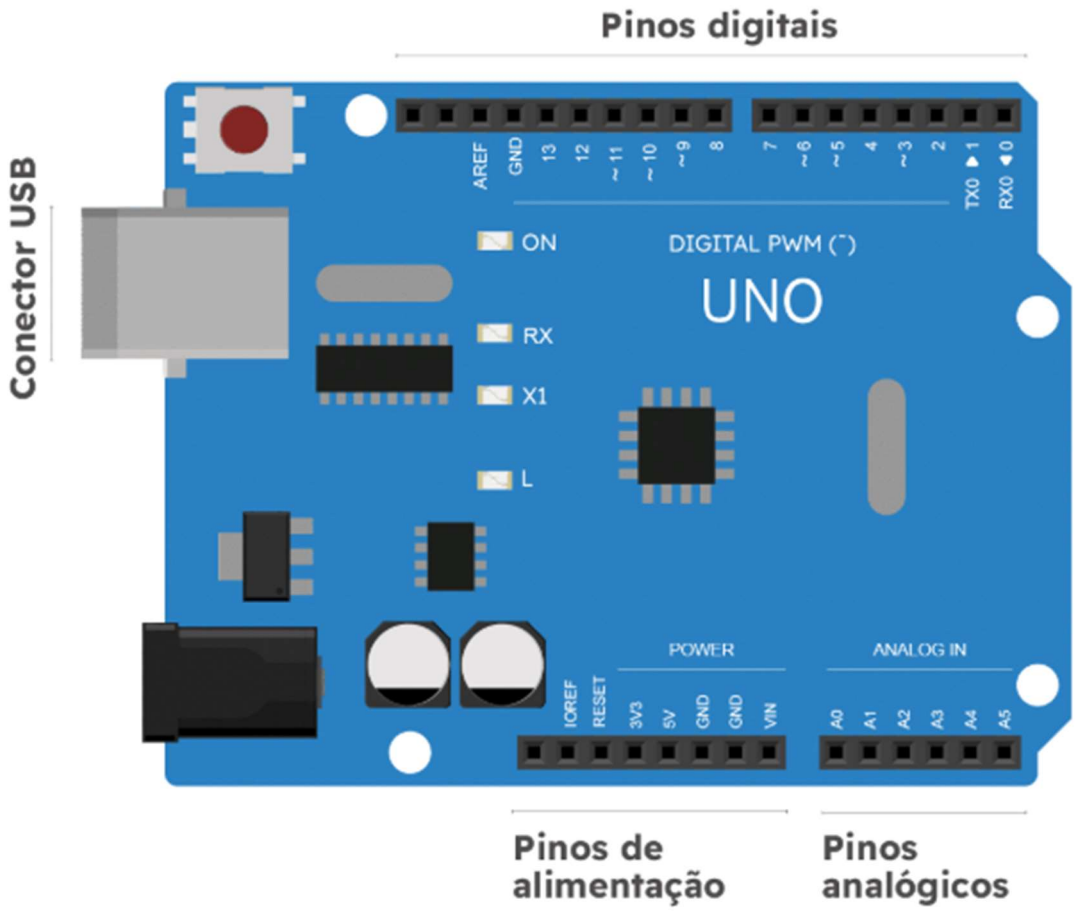
1.2 – O que é Arduino?

Você já deve estar se perguntando como tornar a eletrônica e a programação acessíveis e de fácil entendimento para pessoas sem conhecimento técnico ou sem formação em Engenharia? Foi aí que surgiu o Arduino e deixou tudo mais simples.

Arduino é a principal plataforma de prototipagem eletrônica e está amplamente difundida no mundo. Desenvolvedores, engenheiros, estudantes, empresas e makers em geral estão utilizando Arduino para inovar em seus projetos dos mais variados tipos. Existem diversas placas de Arduino, mas neste guia vamos trabalhar apenas com a mais simples delas, a Arduino Uno.

A ideia do nosso Kit Maker Arduino Iniciante é ajudar quem está começando no mundo da eletrônica. Neste kit você vai encontrar tudo o que precisa para dar seus primeiros passos e desenvolver projetos utilizando um dos hardwares mais versáteis do mundo da eletrônica!

O Arduino Uno tem um microcontrolador que é o “cérebro” da placa e é ele que programamos para realizar os nossos projetos. Além do microcontrolador, a placa possui diversas entradas e saídas e pode ser facilmente conectada ao computador através de um cabo USB. Para programá-la, utilizamos um programa chamado IDE (*Integrated Development Environment*, ou Ambiente de Desenvolvimento Integrado).



As entradas e saídas do Arduino são esses vários pinos, mais comumente chamados de portas. É através dos pinos que a placa interage com os outros componentes. O microcontrolador vai receber e enviar comandos através dessas portas de acordo com o que foi definido em sua programação.

Uma porta de saída controla um dispositivo e uma porta de entrada lê um sinal de um dispositivo externo. Assim, o Arduino pode ser ligado em conjunto com dispositivos externos para controlá-los ou para ler suas informações.

As portas do Arduino podem ser digitais ou analógicas. Mais para frente explicaremos o que são sinais analógicos e digitais, então agora só precisamos saber que o Arduino nos oferece as duas possibilidades.

Os pinos que estão marcados com um til “~” indicam que ele suporta a saída analógica.
Os pinos A0 a A5 suportam entradas analógicas.

Depois de programada, a placa Arduino pode ser usada de forma independente, sem a necessidade de um computador. O programa fica salvo na placa e basta você fornecer energia

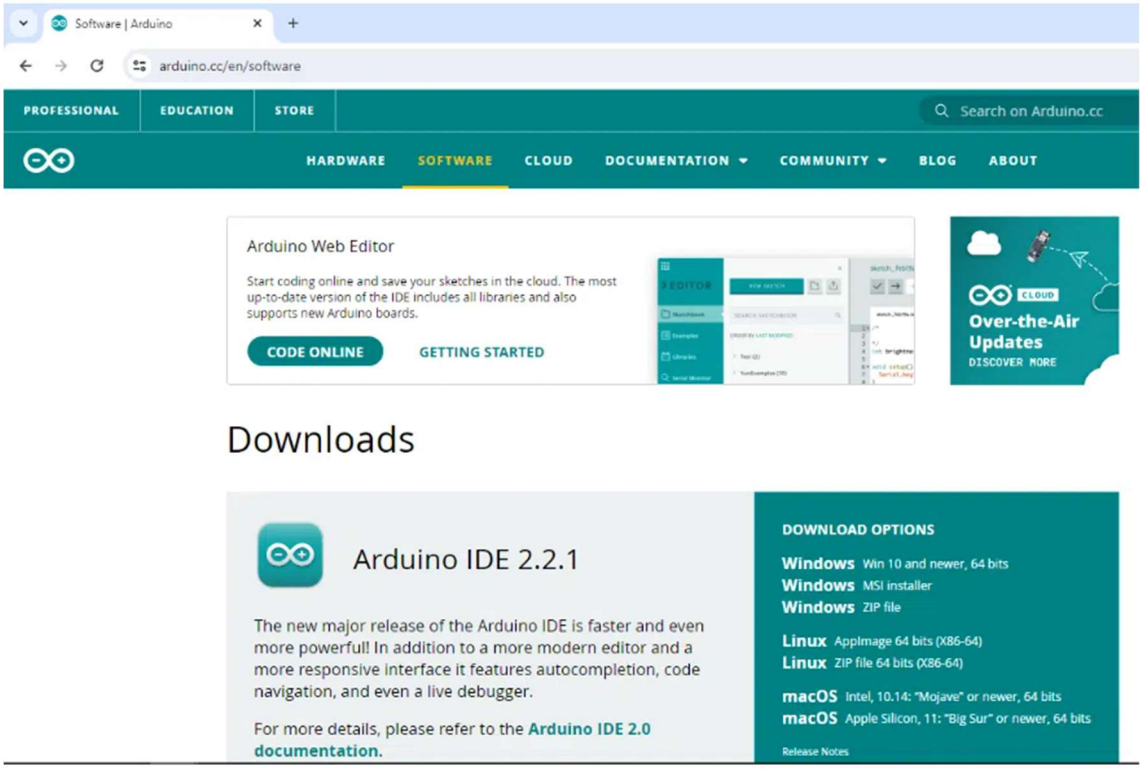
para ele funcionar, colocando o cabo USB do Arduino em um carregador de celular, por exemplo.

Ou seja, você pode colocá-la para controlar um robô, um ventilador, as luzes da sua casa, a temperatura do ar condicionado ou pode utilizá-la como um aparelho de medição. Existem diversos sensores e módulos que podem ser usados em conjunto com Arduino para expandir suas funções, assim você pode desenvolver qualquer projeto que vier à cabeça.

1.3 – Baixando e Instalando a IDE Arduino

Antes de começar os projetos, vamos precisar de um programa chamado IDE (*Integrated Development Environment*, ou Ambiente de Desenvolvimento Integrado). Com a IDE Arduino podemos dar os primeiros passos, desenvolver programas e realizar a gravação dos programas na placa.

Para fazer o download da IDE Arduino entre no site oficial do Arduino na seção *Software* -> *Downloads* e escolha sua versão de sistema operacional(Windows, Linux, MacOS).



1.4 – Estrutura de um Programa Arduino

Você não precisa ser um expert em programação para conseguir programar com Arduino, mesmo assim, é bastante interessante que você procure entender o programa. No começo vai ser mais complicado, algumas coisas podem não ficar tão claras, mas com experimentos e força de vontade você será capaz de entender cada linha desse guia. Esse entendimento é essencial para que você possa fazer seus próprios projetos no futuro.

Ao abrir a IDE Arduino, que instalamos na introdução, você se depara com um programa em branco, que não faz nada.

Cada programa é composto de uma série de funções e comandos, cada uma desempenhando um papel para o funcionamento geral do programa.

A função, no contexto de programação, é um conjunto de instruções que realizam uma determinada tarefa. Podem ou não receber valores (parâmetros) quando são chamadas para realizar a sua tarefa. Separamos o programa em funções para que fique mais claro o que cada parte do programa está fazendo e tornando mais simples caso seja preciso alterar algo no programa, além de facilitar o reaproveitamento quando precisamos daquela mesma tarefa em outro programa.

A função é construída da seguinte forma:

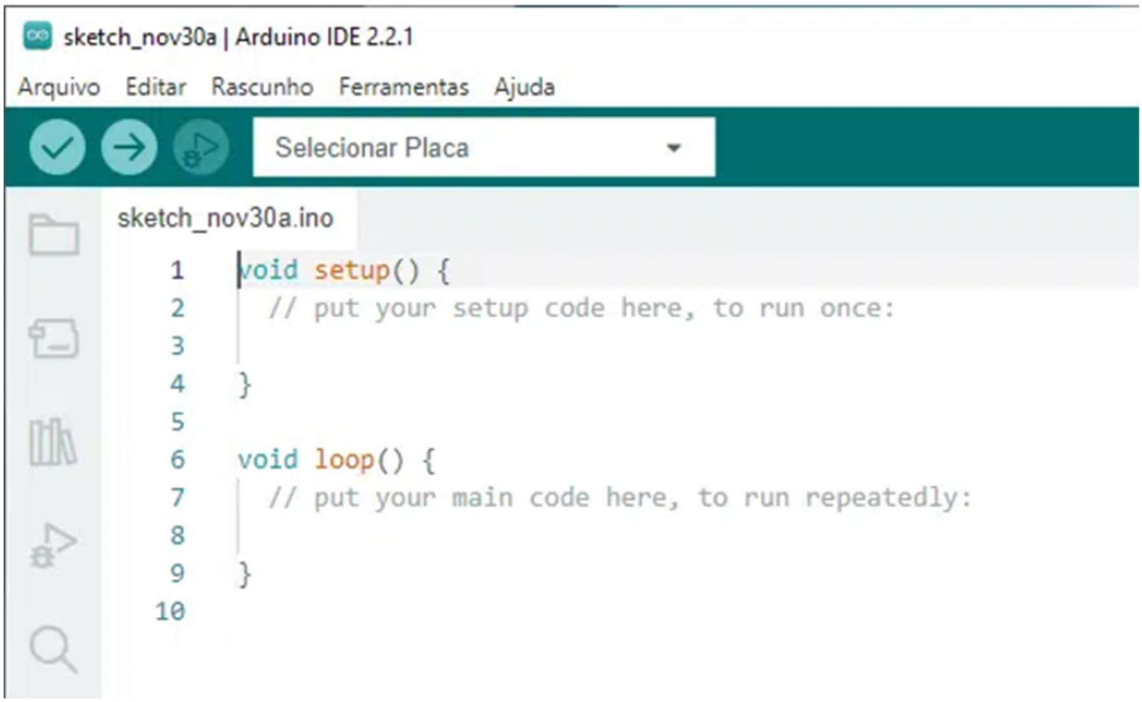
```
1  void minhaFuncao(int parametro1, int parametro2, … ) {
2
3  (...)
4
5  }
```

Primeiro vem o tipo da função, que variável ela devolve ou se não devolve nada, o nome da função, e entre parênteses os parâmetros que ela recebe ao ser chamada, se receber algum. Se não receber nenhum parâmetro, devemos apenas abrir e fechar os parênteses sem nada dentro.

Note que o conteúdo de cada função deve ir dentro das chaves “{ }”. O número de chaves que abrem “{” devem ser o mesmo das que fecham “}”, senão o programa dá um erro e não é possível passar para o Arduino. Se você tiver dificuldades em se entender com as chaves, a IDE do Arduino te dá uma ajudinha: Basta clicar em uma chave que abre “{” que ele irá apontar a que fecha “}”. Se você clica em uma chave que não mostra onde está o seu par, provavelmente ela está sobrando no código.

Não é o foco aprender tudo sobre funções, mas dar uma noção geral para que você reconheça a estrutura básica delas e consiga entender o que o programa está fazendo, ou porque determinado erro está acontecendo.

A estrutura padrão de programa contém as funções **setup()** e **loop()**.



setup() – É nessa parte do programa que você configura as opções iniciais do seu programa: os valores iniciais de uma variável, se uma porta será utilizada como entrada ou saída, mensagens para o usuário, etc. Essa função será executada apenas uma vez quando o Arduino for ligado.

loop() – Diferente da função setup(), essa parte do programa repete uma estrutura de comandos de forma contínua ou até que algum comando de "parar" seja enviado ao Arduino. Vamos ver exatamente como isso funciona ao decorrer dos projetos.

Você pode notar que existem coisas escritas após as barras duplas "//", Isso é chamado de comentário. Ele não altera seu funcionamento, mas serve para quem está programando explicar algo para alguém que vai lê-lo.

Outro formato de comentário é o "/*", que transformará tudo que estiver entre ele e o final "*/" em comentário. É útil para comentários maiores ou para fazer uma divisão de segmentos do código.

```
1  /* comentário de
2  várias linhas
3  para textos maiores
4  e explicações mais complexas
5  /*
```

Uma das boas práticas de programação é documentar o seu código por meio das linhas de comentário. Isso será útil não só para você, se precisar alterar o código depois de algum tempo, como também para outras pessoas que utilizarão o seu programa.

Atente-se aos comentários, eles vão ajudar!

O código também conta com variáveis, que são “apresentadas”(declaradas) para o arduino uma vez no código, para que depois possam ser chamadas e utilizadas ou alteradas. Quando declaramos uma variável, dizemos de que tipo ela é, seu nome e podemos ou não atribuir um valor inicial à ela.

```
1  int contador1;
2  int contador2 = 0;
```

A variável contador1 foi apenas declarada, enquanto a contador 2 foi declarada e foi atribuido o valor inicial 0 a ela.

As variáveis guardam valores para utilizarmos novamente em nosso código.

Quando fazemos a apresentação da variável, ela deve ser feita no local que será utilizada. Se ela só for utilizada em uma função, deve ser declarada dentro daquela função.
Já se será utilizada por todo o código, deve ser declarada fora das funções.

Em geral fazemos todas as declarações de variáveis que são usadas por todo o programa antes das funções do código.

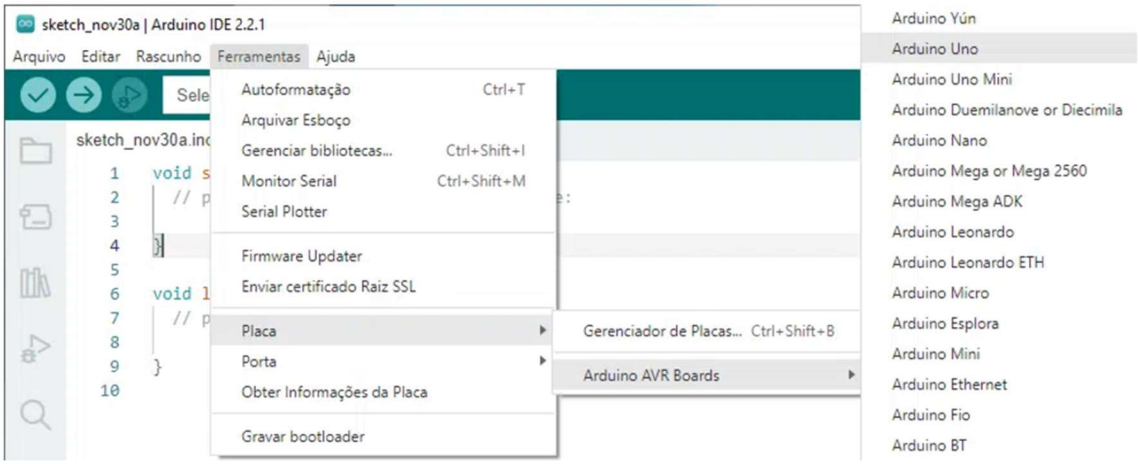
```
1  int contador1;
2  int contador2 = 0;
3  void setup() {
4      // put your setup code here, to run once:
5  }
6  void loop() {
7      // put your main code here, to run repeatedly:
8  }
```

1.5 – Carregando um Programa no Arduino

Agora vamos aprender o passo a passo da IDE para **quando o código estiver pronto para ser carregado na placa**, utilize o cabo usb para conectar a placa Uno R3 no seu computador.

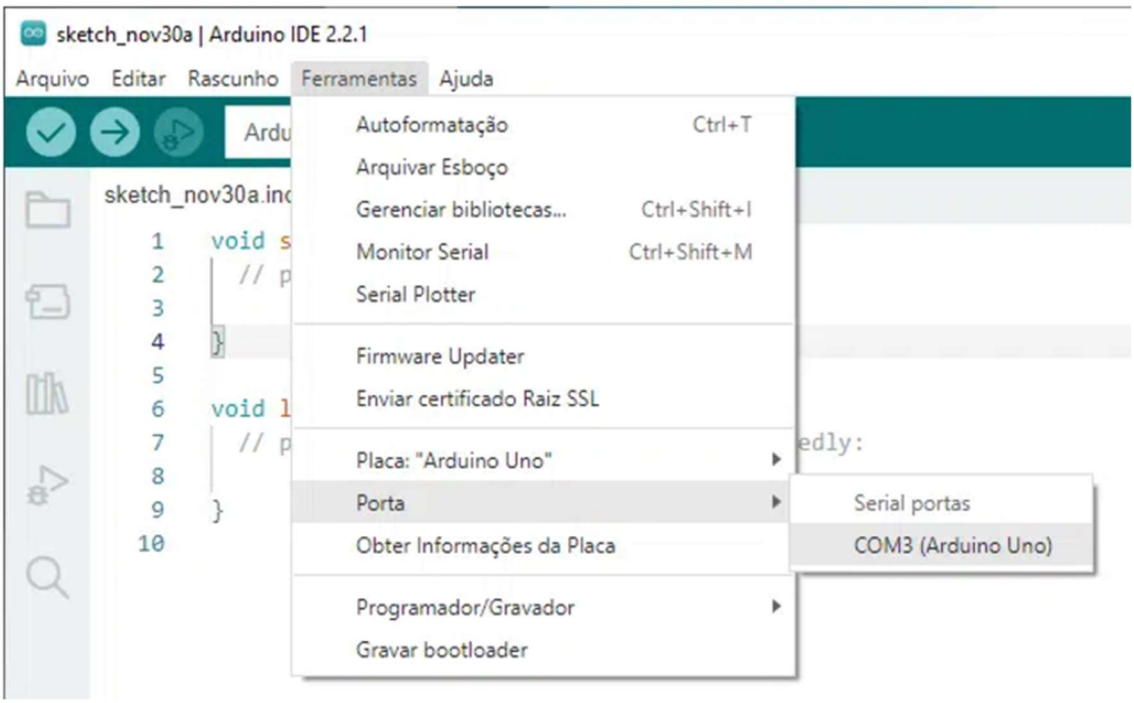
Com a placa conectada no computador siga os passos a seguir:

- entre no menu **Ferramentas**;
- em **Placa:** escolha "**Arduino AVR Boards**";
- e então escolha o modelo da placa como "**Arduino Uno**" ;



Em seguida entre no menu **Ferramentas** novamente e veja a porta na qual a placa está conectada, que no caso do sistema Windows, será identificada com a palavra COM seguido de algum número (ex: COM3, COM4 ou COM5, etc.).

Caso não apareça nenhuma porta, teste com outras entradas USB ou até mesmo em outro computador.



Então clique no botão de checagem. Isso irá verificar se existe algum erro no código e gerar o arquivo de programa necessário para carregar na placa.

Se estiver tudo OK, clique no botão carregar. Isso irá gravar o programa na placa.

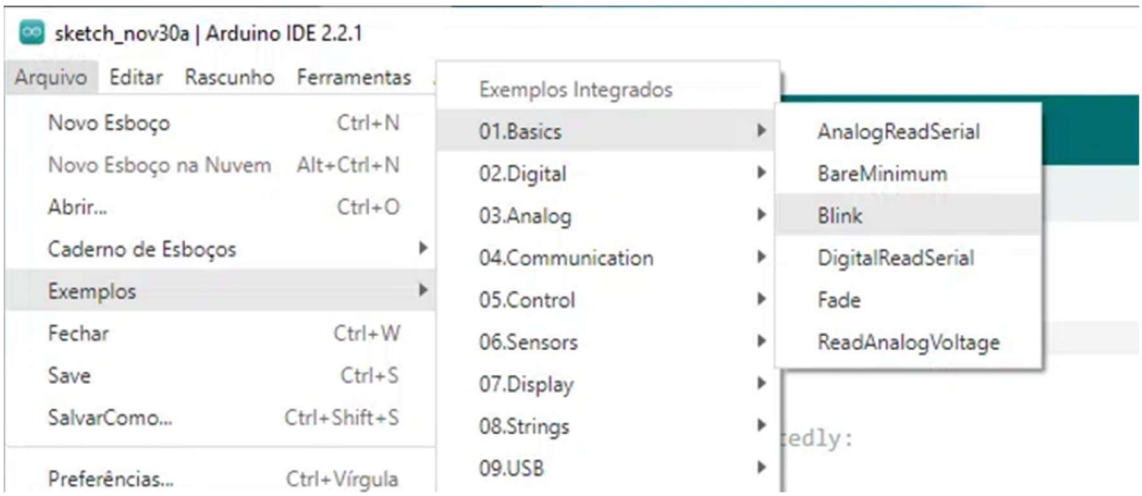
Caso não apareça nenhum erro, você deverá ver o LED da placa piscando em um intervalo de 1 segundo.

Pronto você aprendeu como carregar um código na placa! Portanto os passos apresentados acima você deve seguir quando for carregar um código de projeto das aulas que veremos ao longo do curso.

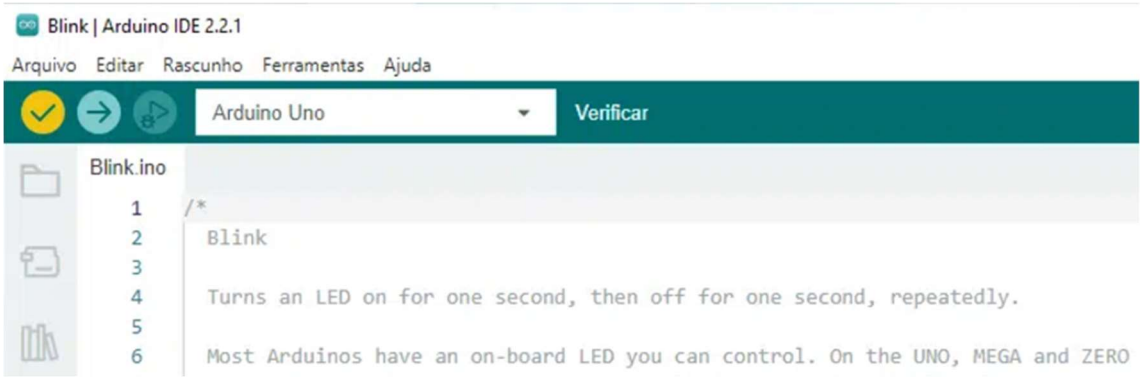
1.6 – Piscar o LED embarcado

Para garantir que está tudo funcionando corretamente, vamos carregar um exemplo de código que já está disponível na IDE, o código Blink. Isso fará o LED da própria placa piscar em intervalos de 1 segundo. O LED em questão está identificado com L na placa UNO.

Com a placa Arduino UNO conectada ao computador. Abra a IDE e vá no menu Arquivo > Exemplos > 01.Basics > Blink.



A IDE abrirá um código pronto, com diversos comentários, e que irá piscar o LED presente na placa em intervalos de um segundo. Verifique o código para se acostumar com o procedimento, mas como é um exemplo da própria Arduino, não deve conter erros.



Por fim, clique em Enviar usando Programador para carregar o código para a placa.

2 – Teoria básica de circuito e boas práticas

2.1 – Circuitos

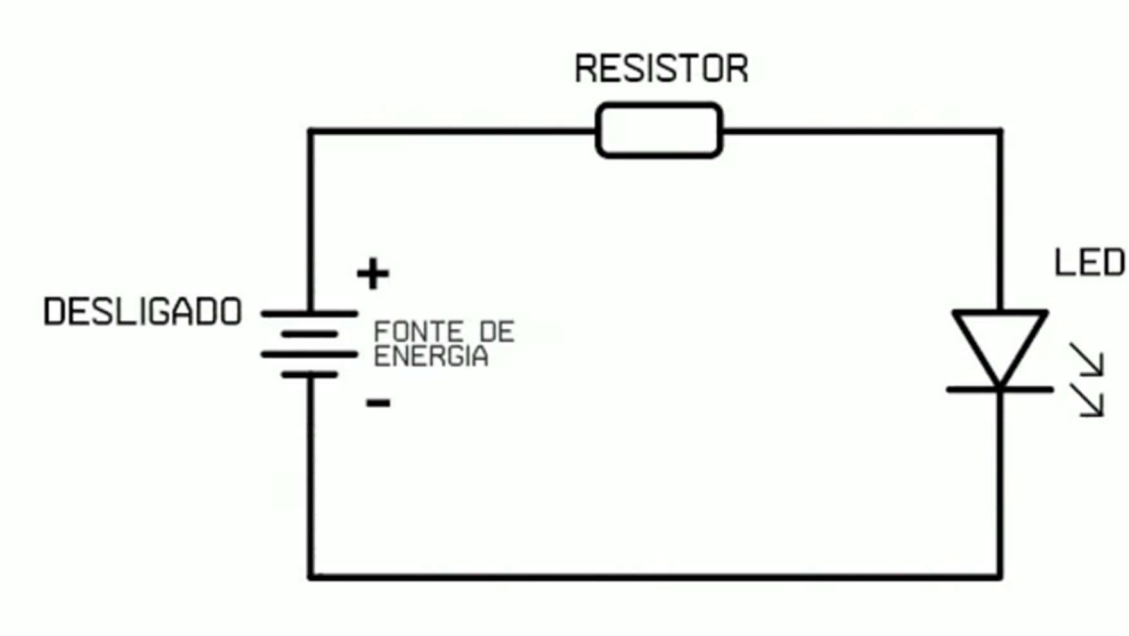
Como funciona um circuito eletrônico?

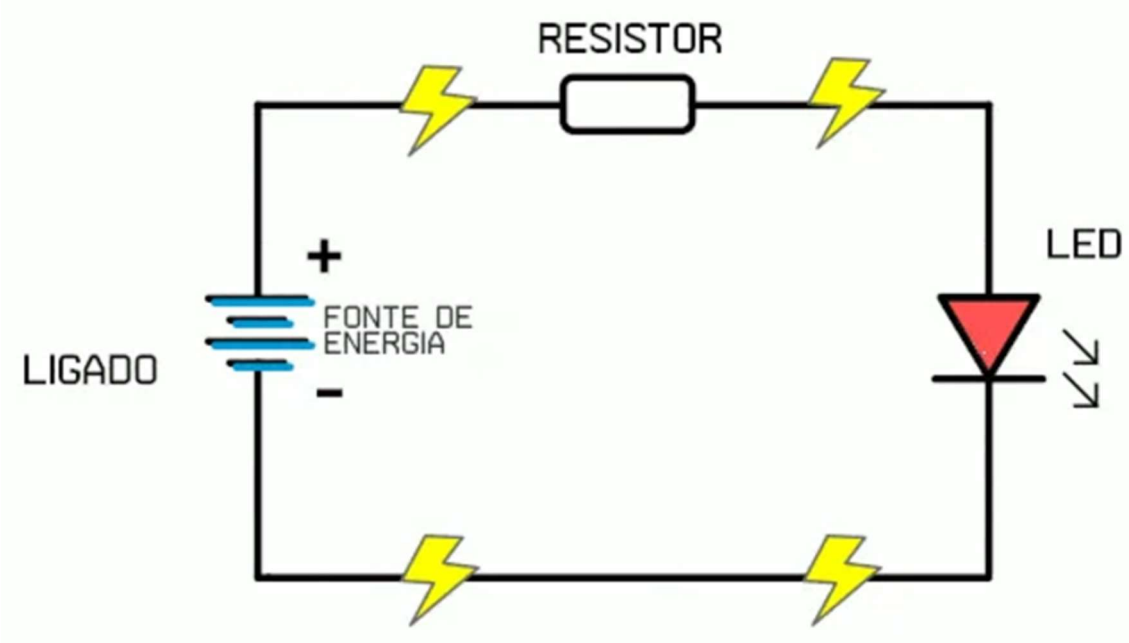
Antes de fazermos a montagem dos exemplos e projetos, é interessante aprender um pouco sobre o que é um circuito eletrônico e alguns conceitos básicos.

Um circuito eletrônico é um conjunto de componentes eletrônicos conectados entre si para realizar uma função específica. Os componentes eletrônicos mais comuns são:

- Diodos: Dispositivos que permitem a passagem da corrente elétrica em uma direção apenas.
- Resistores: Dispositivos que resistem à passagem da corrente elétrica.
- Capacitores: Dispositivos que armazenam energia elétrica.
- Indutores: Dispositivos que armazenam energia magnética.
- Transistores: Dispositivos que amplificam ou comutam a corrente elétrica.

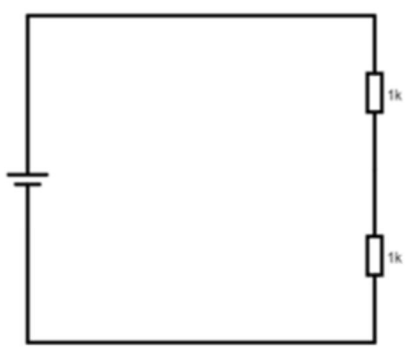
A função de um circuito eletrônico é determinada pela forma como os componentes são conectados entre si. Por exemplo, um circuito simples com um LED e um resistor pode ser usado para criar um indicador luminoso.



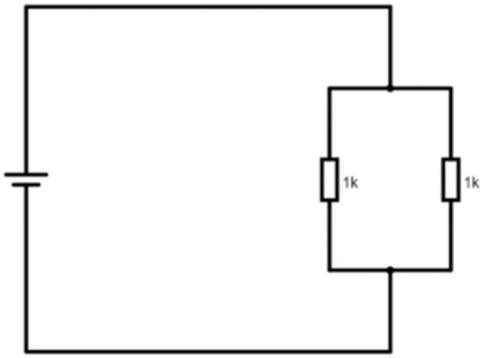


O lado **positivo** da fonte é chamado de **VCC (ou 5V no caso do Arduino)** e o lado **negativo** é chamado de **GND**. Uma fonte pode ser uma bateria ou pilha que quando ligada faz com que o circuito funcione. Também veremos que algumas portas do Arduino (os pinos onde você irá encaixar os jumpers) podem fazer um papel de fonte, assim podemos ligar e desligar o circuito utilizando programação.

Podemos ligar os componentes de duas maneiras diferentes nos circuitos, em série, onde ficam em fila, ou em paralelo, onde ficam um ao lado do outro.



Ligação em série



Ligação em paralelo

Essas ligações fornecem caminhos diferentes à energia, e fazem com que ela seja distribuída de forma diferente dependendo da ligação.

A energia vai sempre procurar o caminho mais fácil para passar, e cada componente tem uma característica de deixar a corrente passar com mais facilidade ou mais dificuldade. Pense em um cano de água, um cano maior deixa passar mais água enquanto o cano menor não deixa passar tanta água.É assim que conseguimos montar os circuitos para fazer a energia passar onde queremos e da forma que queremos.

Podemos calcular essa limitação de passagem da energia com a famosa lei de Ohm :

V=R*I

A ideia não é passar muita teoria ou cálculos nesse guia, mas essa é a fórmula mais básica de toda a eletrônica, então não podemos deixar de mencioná-la.

Vamos explicar aqui como é feito o cálculo, mas fique tranquilo que fizemos todas as contas dos projetos para você. Para quem quiser entender mais a fundo os projetos, essa conta será útil mais para frente.

Vimos nos circuitos que temos a fonte, que seria o V nessa fórmula. R é o valor da resistência e I a corrente que a resistência “deixará passar” através dele.

Imagine que você ligou uma fonte de 5 V num resistor de 220 ohms, podemos achar a corrente que passa pelo circuito.

Tensão = 5 V

Resistência = 220 ohms

V=R*I

Substituindo os valores, temos:

5 = 220 * I

Fazendo a álgebra necessária temos que I = 5 / 220, logo a corrente (I) é aproximadamente 0,023 A

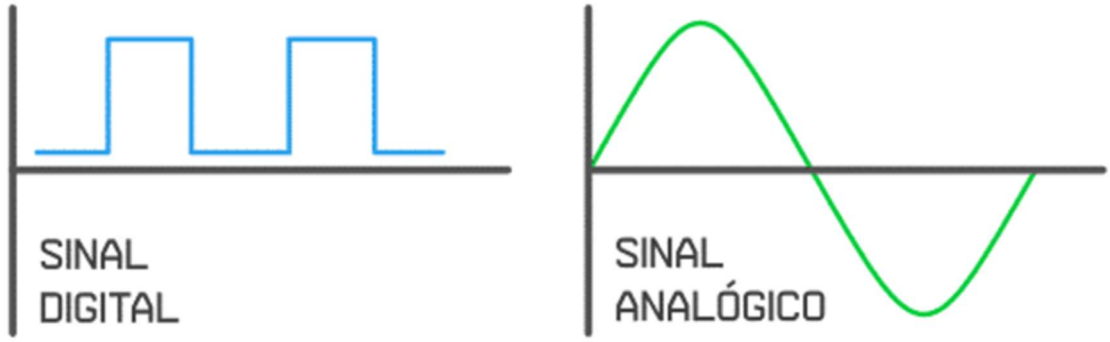
Você já percebeu que alguns valores do dia a dia possuem um “k” ou “M” em sua descrição? Como algumas grandezas possuem valores altos, como: 10000, 150000, 2000000000 (2 milhões), para não precisarmos escrever tantos zeros, utilizamos os multiplicadores, que são esses símbolos. Exemplos: 1k (1000), 15k (15000), 1M (1000000). Abaixo temos uma tabela com os principais multiplicadores usados na eletrônica:

Prefixo	Símbolo	Multiplicador	Formato potencia de 10
micro	µ	0.000001	10-6
mili	m	0.001	10-3
–	–	1	100
quilo	k	1000	103
mega	M	1000000	106

Para exemplificar o uso dos multiplicadores, vamos pegar o resultado da conta que fizemos acima. A corrente que encontramos na conta anterior é 0,023 A. Utilizando o multiplicador mili(m), podemos escrever que a corrente é igual a 23 mA.

2.2 – Sinais digitais e analógicos

Nos circuitos podemos ter normalmente dois tipos de sinais, os sinais digitais e os sinais analógicos.



Os sinais digitais são sinais que tem dois valores bem definidos, podendo ter alguns nomes diferentes, mas significando a mesma coisa:

1	0
5 V	0 V
HIGH	LOW
SIM	NÃO
VERDADEIRO	FALSO
LIGADO	DESLIGADO

As entradas e saídas digitais do Arduino recebem ou enviam esse tipo de sinal. Pode parecer pouco, apenas ligado ou desligado, mas se combinarmos uma sequência de sinais ligados e desligados podemos construir mensagens elaboradas (como uma onda PWM, comunicação serial, etc), fique tranquilo que veremos esses exemplos mais adiante.

Diferente do sinal digital, o sinal analógico pode ter diversos valores dentro de uma faixa de tensão (ou corrente), exemplos:

a) 0 a 5 V

b) 0 a 12 V

c) 0 a 3,3 V

Um exemplo de aplicação de um sinal analógico é o volume de um aparelho de som, o mesmo pode ser: silencioso, muito baixo, baixo, médio, meio alto, alto e máximo. Essa alteração é feita pelo usuário, que ajusta a tensão do sistema que gera o som.

As portas marcadas com “~” no Arduino são capazes de gerar saídas analógicas, podendo variar o brilho de LEDs ou a velocidade de motores, entre outras aplicações.

As portas de entrada analógicas (A0 a A5 no Arduino UNO) são capazes de ler sinais variados e converter em informações de intensidade que podem ser utilizadas na programação do Arduino.

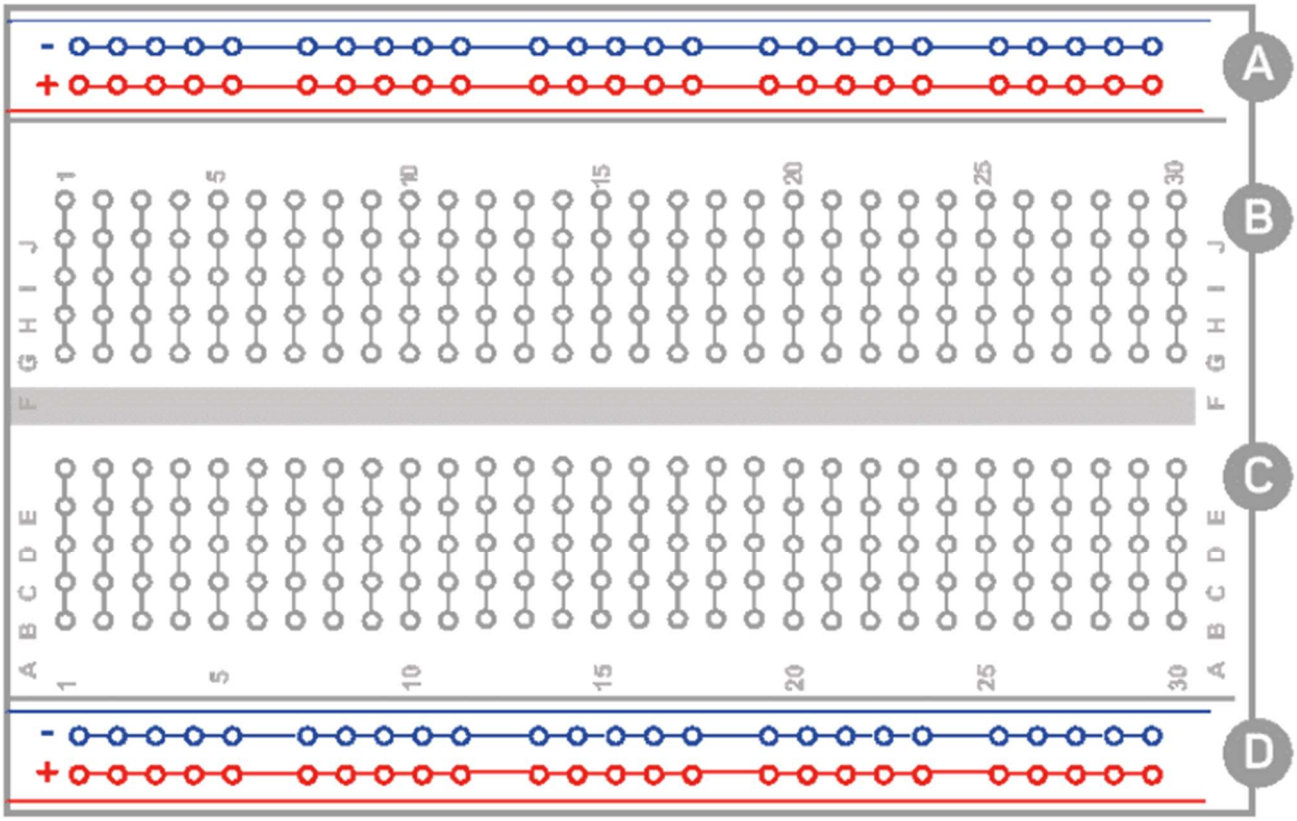
Entrada de sinal analógico: Pinos A0 a A5
b) Saída de sinal analógico: Pinos 3, 5, 6, 9, 10, 11 (que tem um ~) na frente.
c) Entrada e saída de sinal digital: Praticamente todos os pinos do Arduino UNO, incluindo os A0 a A5, com “~” e etc.

O diferencial das portas com “~” é que elas podem atuar como porta digital e como porta analógica. Já as que não tem esse símbolo, atuam apenas como porta digital.

2.3 – Como funciona uma protoboard

A protoboard é um componente utilizado em todos os projetos do nosso kit e em quase todos os projetos de eletrônica. É nela que se faz a montagem dos circuitos eletrônicos, ação também conhecida como **prototipagem**.
Uma protoboard é cheia de furos, onde você pode encaixar os pinos e terminais dos componentes eletrônicos, ou até mesmo um fio (jumper) diretamente, e tem uma lógica de conexão entre os furos que, depois que você entende, a montagem dos seus projetos fica bem intuitiva.

Com a protoboard podemos experimentar a montagem de diversos tipos de combinações de componentes eletrônicos e circuitos sem a necessidade de conectá-los permanentemente. Caso haja a necessidade de trocar um componente de posição ou mesmo substituí-lo, isso pode ser feito de maneira muito rápida e fácil.
Como você pode ver na figura abaixo, na protoboard existem dois blocos de colunas B e C, que são uma série de 30 colunas (1 a 30) lado a lado nomeada de “a” a “e” e “f” a “j”. Cada coluna possui 5 furos e esses estão interligados entre si como mostrado nas linhas cinzas. Uma coluna não possui conexão interna com a coluna ao lado.
Nos blocos B e C são montados a maior parte dos circuitos. Mas os dois blocos não são interligados entre si, sendo separados por uma cavidade central.

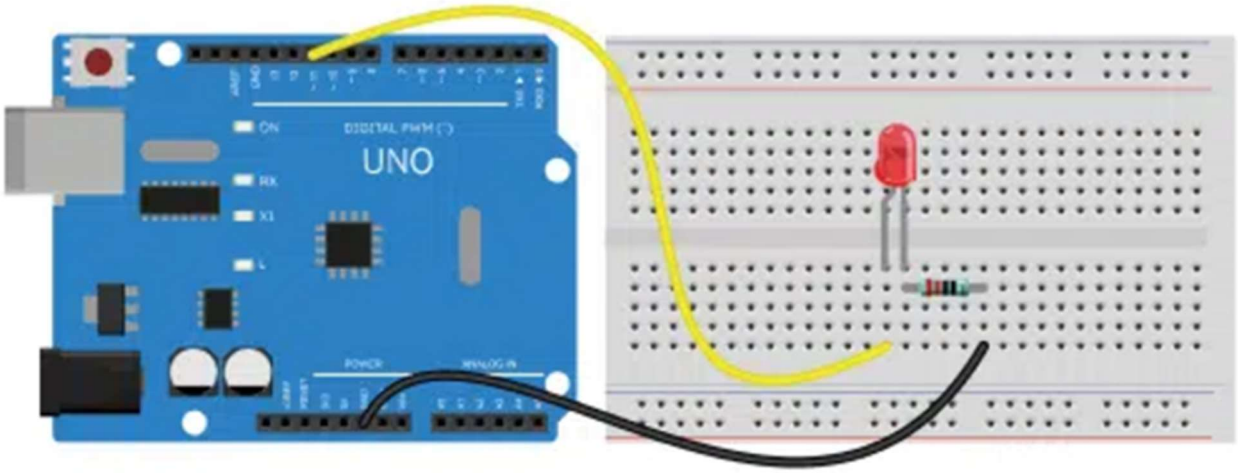


Já nos blocos A e D, nas extremidades superior e inferior da protoboard, temos as linhas em vermelho e azul. Todos os furos de uma mesma linha estão interligados entre si, mas os furos da linha vermelha não estão conectados aos furos da linha azul. Na linha vermelha existe um sinal de positivo “+” e na azul um sinal de negativo ”-”.
É nessas linhas que ligamos a energia do circuito que vamos montar. Nos projetos com Arduino, geralmente se coloca o 5 V no vermelho e o GND no azul, apenas uma questão de organização.
Nos nossos projetos sugerimos uma montagem através de um desenho auxiliar. Mas se você mantiver as conexões e respeitar a lógica da protoboard, pode montar o projeto em qualquer lugar dela.

2.4 – O que é um Jumper

Jumper é o fio que conecta os componentes de um circuito e permite a passagem da energia elétrica. Você pode ligá-los na protoboard, na placa Arduino ou mesmo direto no componente. Para ajudar na organização do seu circuito os jumpers possuem diversas cores, que não alteram em nada o seu funcionamento, servem apenas para que você crie uma organização dentro da sua montagem.

Nos esquemas de montagens de circuitos eletrônicos, os jumpers são representados apenas como traços coloridos que ligam os componentes, como você pode ver abaixo um dos esquemáticos que utilizamos ao longo das nossas aulas.



Uma característica dos jumpers é que eles já vem com os conectores certos para você encaixar na protoboard e na placa, enquanto um fio desencapado seria mais difícil de fazer um bom contato ou fazer entrar nos buracos da protoboard.
Nos nossos esquemáticos indicamos a quantidade de jumpers macho-macho e macho-fêmea que você deve usar. Note que para as conexões dos LEDS que não são conectados diretamente na protoboard usamos jumpers macho-fêmea, para permitir a maior mobilidade e encaixe nas peças de impressão 3D.

2.5 – Boas práticas

Ao trabalhar com circuitos e componentes eletrônicos, devemos ter certos cuidados, tanto para não causar algum acidente, quanto para não estragar os componentes com os quais estamos lidando.

- Crie um padrão de cores para as ligações do seu circuito, assim você conseguirá se organizar de forma mais fácil nas conexões do circuito.

- Apesar da protoboard ter o azul na linha “-” e o vermelho na linha “+”, convencionalmente utiliza-se o fio vermelho para o positivo e o fio preto para o negativo em circuitos.
  Com as demais cores, nem sempre temos o suficiente para todas as ligações necessárias, então procure deixar cores iguais afastadas para facilitar o entendimento das conexões.

- Evite deixar jumpers sem uso na protoboard. Nós sabemos, é tentador deixar aquele jumper do projeto anterior ali porque ele não vai atrapalhar. Mas crie o hábito de deixar apenas o necessário para aquele circuito, vai facilitar o entendimento, a visualização e evita conexões indevidas e comportamentos “estranhos” do circuito aparentemente sem explicação. Além disso, caso seja necessário, será mais fácil fazer qualquer alteração no circuito, seja de componente ou simplesmente no esquema de ligação.

- Sempre que for fazer alguma alteração na montagem, desligue o Arduino. Assim, caso você encoste algum componente, pino ou jumper em um lugar que não deveria, não causará problemas. Se o circuito estiver ligado, você corre o risco de estragar algum componente ou até mesmo o seu Arduino.

- Com LEDs e resistores, tome cuidado para que os terminais, que são bastante compridos, não estejam encostando em nenhum lugar além do pino da protoboard ou jumper em que devem estar conectados.

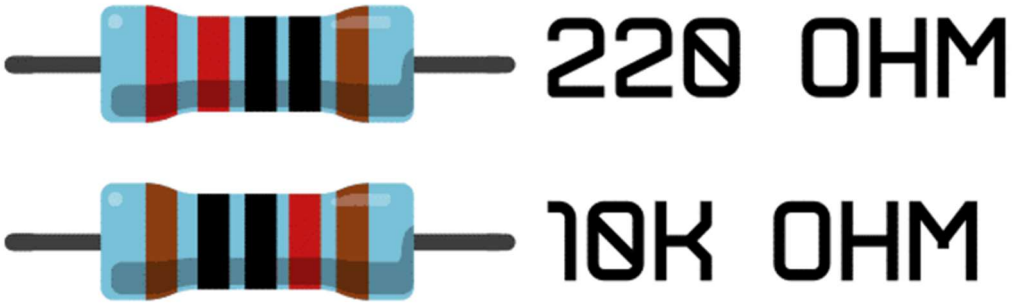
3 – Resistor e potenciômetro

Módulo 3 – Resistor e potenciômetro – Você será apresentado ao resistor e ao potenciômetro, dois componentes básicos em circuitos eletrônicos para a composição de projetos

3.1 – Resistor

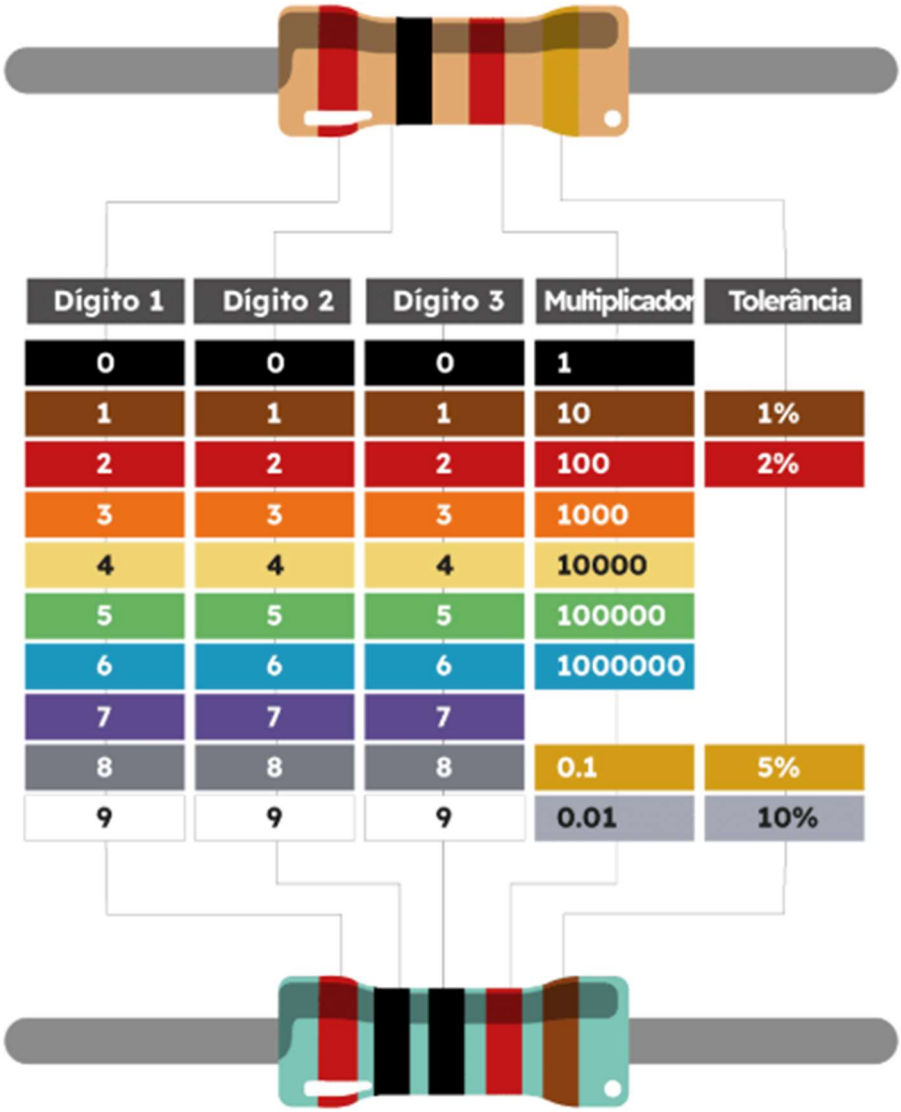
O resistor é um dos componentes eletrônicos mais comuns que existem. Uma das aplicações de um resistor, e a mais comum, é limitar o fluxo da corrente elétrica que passa em um circuito.

Os resistores têm diferentes valores de resistência, quanto mais alta a resistência, mais ele irá limitar a corrente que passa por ele. O valor do resistor é fixo e é indicado pelas faixas de cores pintadas nele. No kit temos dois tipos: 220 ohm e 10K ohm.



Aviso! O kit pode conter resistores de coloração ou faixa de cores diferentes da ilustração acima, mas não se preocupe, basta **olhar o valor escrito na etiqueta do pacote de resistores**. Caso venham na cor amarela e quatro faixas eles são resistores equivalentes aos azuis, só utilizam uma nomenclatura diferente, portanto identifique-os de acordo com a ilustração abaixo:

Cada valor de resistor tem o seu código de cores, que você pode ver na tabela abaixo como descobrir o valor do seu resistor pelas barrinhas coloridas dele:



Você pode ligar um resistor tranquilamente pois ele não tem lado positivo ou negativo, ou seja, tanto faz o lado que é conectado. Normalmente a última ou a penúltima faixa de cor é dourada, assim o usuário consegue perceber qual seria a ordem. Além disso a faixa de tolerância costuma ficar mais afastada das faixas que indicam valor.

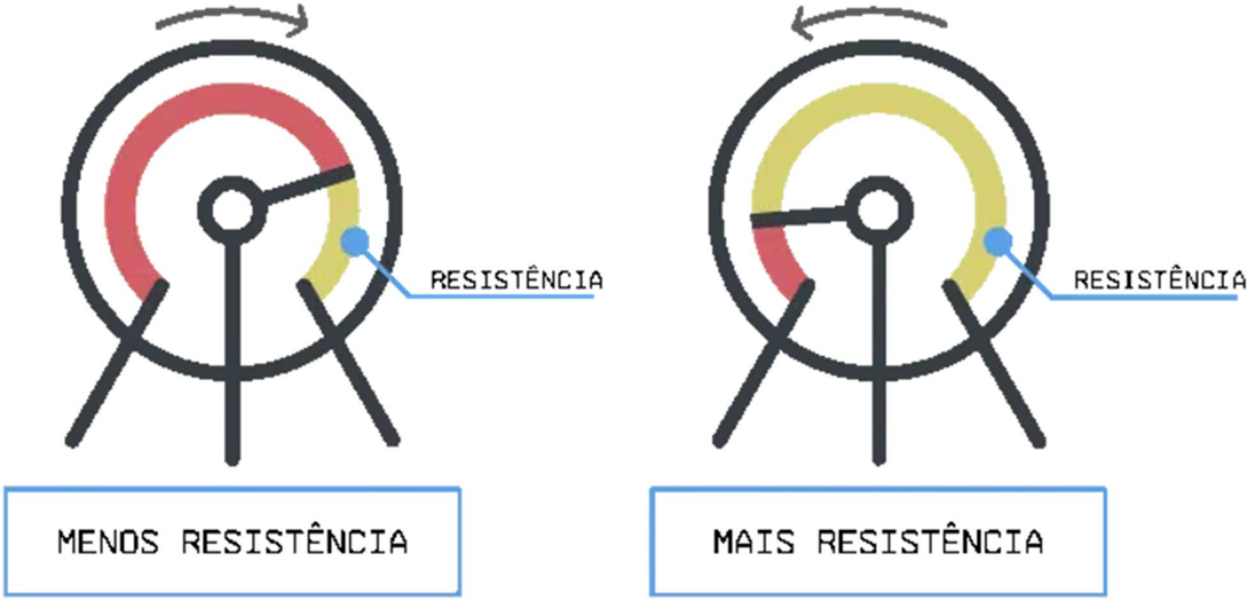
3.2 – Potenciômetro

Sabe o botão de girar do rádio, da guitarra ou de outros equipamentos que você gira para ajustar o volume, a frequência ou até a intensidade de uma lâmpada? Muito provavelmente ele é um potenciômetro.

Podemos resumir o potenciômetro como dois resistores ligados um em linha com o outro (em série), mas com uma característica especial, podemos alterar o valor desses dois resistores girando a sua haste.

O valor especificado do potenciômetro é a sua resistência total, somando os dois lados, ou a resistência medida entre os dois pinos mais externos. Exemplo, se o potenciômetro é de 10k Ohms, o mesmo pode ser ajustado de 0 a 10000 Ohms.

Veja na figura abaixo o seu funcionamento interno.



O terminal do meio do potenciômetro fica ligado bem no meio dos dois resistores, e a tensão ali será proporcional ao ajuste.

Normalmente usamos o potenciômetro com um de seus lados ligado ao 5V, o outro ao GND e o terminal do meio é onde temos o valor que nos interessa.

4 – LED e diodo laser

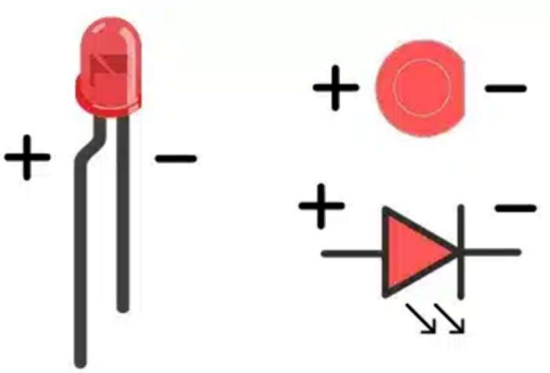
Módulo 4 – LED e Diodo Laser – Comece a entender o funcionamento do arduino com componentes externos conectados à ele, com os LEDs e Diodo laser, que são formas de informação e sinalização do funcionamento dos projetos.

4.1 – LED

Da luz de *standby* da sua tv, às lâmpadas mais modernas, letreiros de ônibus até faróis de carros mais modernos, com certeza você já viu um LED por aí.

LED (do inglês, *Light Emitting Diode*) é um diodo emissor de luz, é basicamente uma "lâmpada" que consome pouca energia. Por seu baixo consumo de energia, está se tornando cada vez mais comum o seu uso em casas (no caso, substituindo lâmpadas fluorescentes e incandescentes por lâmpadas/fitas de LED). Outra grande vantagem do LED é a facilidade de fabricação em diversas cores. Nesse kit você receberá LEDs das cores amarelo, verde e vermelho.

O LED, assim como outros componentes, possui um lado positivo "+" e um lado negativo "-", essa característica também é conhecida como **polaridade**. Para esses componentes, se ligar de modo invertido, ele não irá funcionar e pode até acabar queimando. A haste (perna) maior do LED é o lado positivo e a menor é o lado negativo. Você pode ver também pelo lado mais achatado na parte da cabeça do LED, que é o lado negativo enquanto o lado arredondado é o positivo.



Cada cor de LED tem uma tensão (às vezes chamada de voltagem) característica de operação, e os LEDs comuns de 5mm trabalham numa corrente máxima entre 10 e 20 mA.

Utilizando um resistor, de valor 220 ohm por exemplo, em série com o LED, reduzimos a corrente do circuito para trabalhar nos valores que o LED suporta. Sem o resistor, a corrente que passa pelo LED seria muito alta e acabaria queimando ele.

Se você quiser saber o porquê desse valor de 220, vamos usar de novo a lei de Ohm que já vimos, junto com a tabela a seguir:

Cor do LED	Tensão de operação típica
Amarelo	1,9 a 2,2 V
Vermelho	1,8 a 2,1 V
Verde	2 a 3,1 V

O LED vai “pegar para si” uma parcela da tensão da fonte, e o restante fica com o resistor. Dessa forma temos que a tensão no resistor é a tensão da fonte menos a tensão de operação do LED

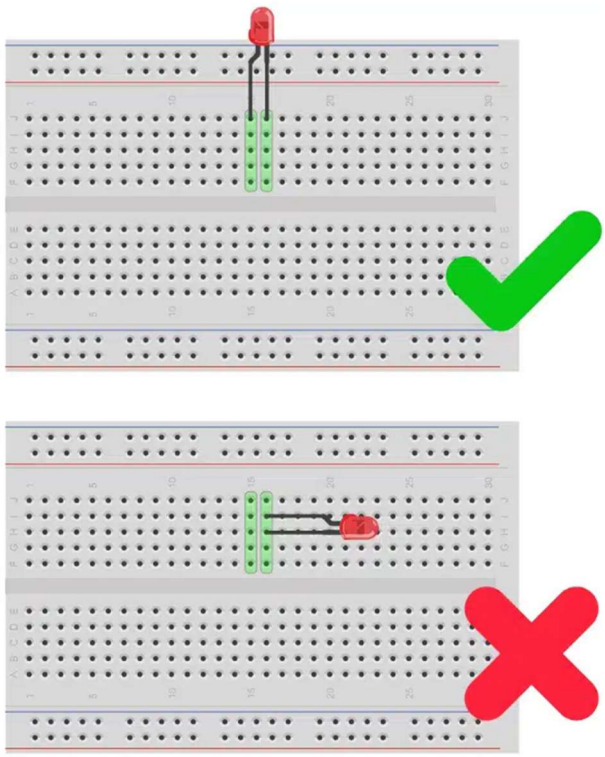
$$V_R = V_{Fonte} - V_{LED}$$

No caso do Arduino a tensão da fonte é 5V. Vamos usar o LED vermelho como exemplo e para facilitar a conta usaremos o valor de tensão de 2V.
Então a tensão no resistor vai ser 3V. Se queremos uma corrente de 10 mA (0,01 A, lembra dos multiplicadores?) temos com a lei de Ohm:

3(V) = R * 0,01(I)

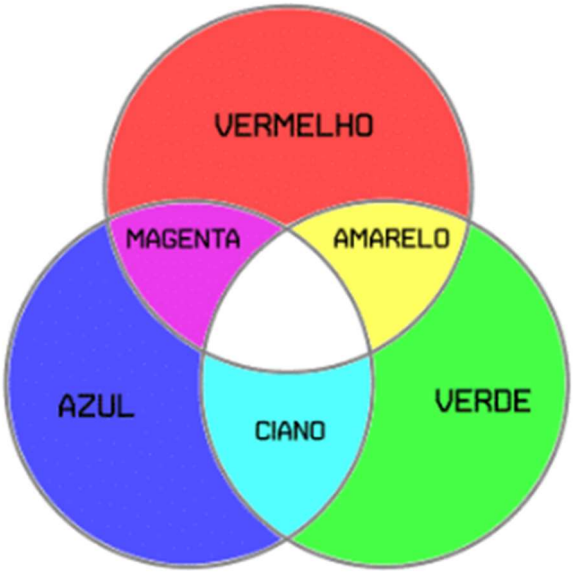
Fazendo os cálculos, temos que R = 300 ohms

A montagem correta de um LED em uma protoboard é feita como mostrado na figura abaixo. Os dois terminais do LED não podem ficar na mesma coluna da protoboard, lembrando que os furos das colunas são conectados entre si na vertical. Esse mesmo princípio de montagem se aplica também a outros componentes do kit.

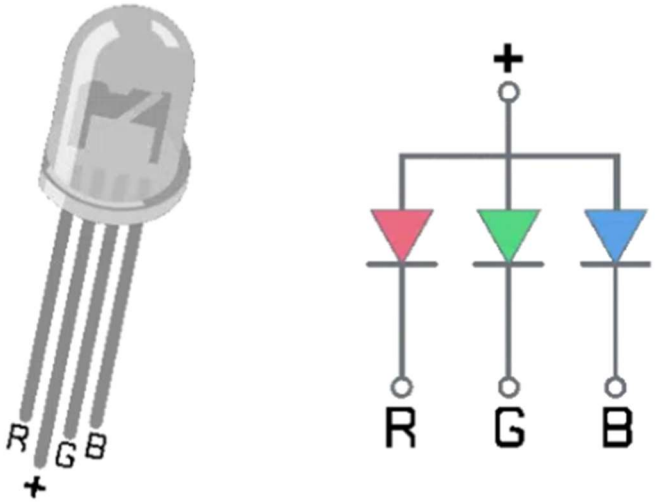


Na imagem acima o circuito não funcionaria pois ambas as pernas do LED estão em pontos interligados da protoboard, e para que funcione corretamente, cada perna tem que estar em um ponto que não tem conexão entre si.

Além disso, alguns LEDs são capazes de mostrar várias cores, chamamos eles de LED RGB. Ele tem o mesmo funcionamento básico de um LED comum porém possui três cores no mesmo componente, como se fossem três LEDs juntos: Red(R), Green(G) e Blue(B). Assim podemos trocar as cores via programação e até mesmo fazer misturas.



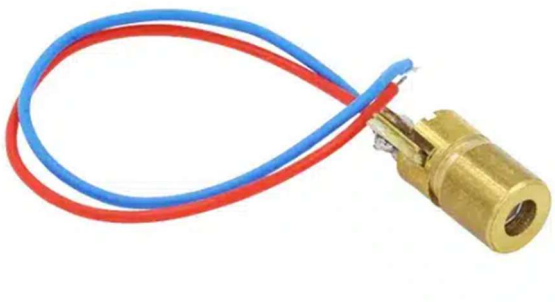
Como mostrado na figura abaixo, o LED RGB possui quatro pernas, sendo a perna maior o pino positivo comum para todos os LEDs internos.



4.2 – Diodo laser

Sabe o console de última geração que ainda usa discos, sejam eles Blu-Ray, DVD ou até os mais antigos CDs? Para ler esses discos ele com certeza usa um diodo laser. Eles também são usados em diversas outras aplicações, como:, ponteiros para apresentações (em lousa), máquinas de corte, etc.

O diodo laser tem uma construção parecida um LED, mas com algumas camadas especiais e uma lente que gera um feixe de luz em que os raios saem praticamente todos na mesma direção, fazendo com que tenha longo alcance e, normalmente, formato de ponto.



Diferente do LED que precisa de um resistor ligado junto, o diodo laser que você recebeu já tem o resistor montado nele, então ele é ligado diretamente no 5V, terminal positivo da protoboard.

Nos próximos exercícios vamos usar os LEDs comuns e vamos guardar o diodo laser para outro momento, em projetos mais avançados como o projeto da aula 9.4 – Alarme com sensor a laser.

4.3 – Pisca Pisca

Agora que conhecemos o funcionamento do resistor e do LED, em nosso primeiro exemplo prático iremos fazer uma luz piscar. Parece um exercício simples demais, mas isso exemplifica a utilização do Arduino para controle de dispositivos externos. Os conceitos aprendidos neste exemplo servem para acionamento de outros dispositivos como ventilador, lâmpadas, motores e etc.

É o primeiro passo que se dá quando se começa a trabalhar com uma placa, como fizemos na introdução ao piscar o LED da própria placa, porque ele mostra que você é capaz de programá-la. Agora vamos fazer com um LED externo. Esse primeiro exemplo será um pequeno grande passo para que você aprenda eletrônica e programação e possa fazer exercícios mais complexos.

Material necessário

Em todos os exercícios e projetos teremos uma seção que mostra os componentes que iremos utilizar. Se você tiver alguma dúvida sobre qual é o componente, você pode voltar na lista de materiais:

1x LED Vermelho 5 mm
1x Resistor 220 ohm
1x Protoboard 400 pontos
2x Jumper macho-macho
1x Cabo USB
1x Placa Uno

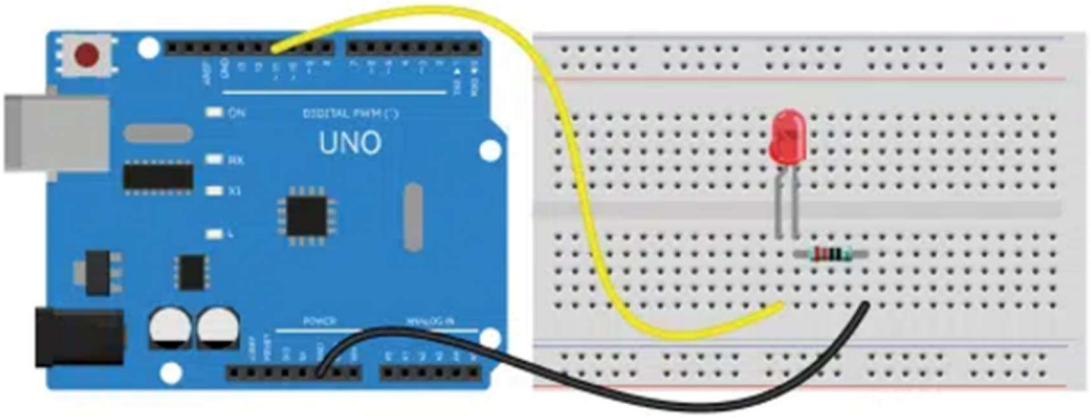
Nesse exemplo, o LED acende quando a fonte de energia é ligada e ao desligar a fonte de energia, o LED apagará. O resistor é colocado para reduzir a corrente que passa pelo circuito inteiro.

Montagem do circuito

No caso do nosso exercício, a fonte de energia vem da porta 11 do Arduino. Com a programação correta, é possível ligar e desligar a energia do pino 11, fazendo o LED acender e apagar.

No esquema da montagem do circuito, as linhas coloridas são a representação gráfica dos jumpers. Utilize-os para ligar os componentes entre si conforme a ilustração abaixo. Note que o pino 11 está conectado ao lado positivo do LED, enquanto o GND vai no pino negativo, passando antes pelo resistor.

A representação de circuito da montagem acima seria a seguinte e você pode encontrar também no livreto identificado como projeto 01- Pisca-Pisca:



Não é necessário conectar os componentes exatamente nos mesmos furos como indicado acima, basta apenas que os terminais de cada componente não estejam na mesma coluna. Também não importa se o resistor está antes ou depois do LED, ele vai reduzir a corrente do laço estando antes ou depois. Veja abaixo uma montagem alternativa do circuito acima, mas que funciona da mesma maneira. Note que o pino GND do Arduino ainda está conectado no negativo do LED e o pino 11 ainda está no positivo do LED.

Programa

```
// CODIGO: Pisca-Pisca
//----------------Funcao executada uma vez na inicializacao do sistema----------------
void setup() {
  pinMode(11, OUTPUT);          // Define a porta 11 como saida
}
//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------
void loop() {
  digitalWrite(11, HIGH);       // Acende o led
```

```
  delay(1000);              // Aguarda intervalo de tempo entre parenteses em milissegundos
  digitalWrite(11, LOW);    // Apaga o led
  delay(1000);              // Aguarda intervalo de tempo entre parenteses em milissegundos
}
```

Nesse primeiro código utilizamos duas funções que estão entre mais mais básicas do Arduino, **digitalWrite() e delay()**

A função **digitalWrite(11, HIGH)** está recebendo como parâmetro o número 11, que é o pino que vamos controlar e o valor HIGH, que indica para ligar o LED.

De maneira inversa, a função **digitalWrite(11, LOW);**, apaga o LED, desligando o pino 11.

O comando **delay(1000);** especifica o intervalo, em milésimos de segundos (milissegundos), no qual o programa fica parado antes de avançar para a próxima linha. Quanto maior esse número, mais tempo o programa vai esperar para passar para a próxima linha. O programa não irá fazer mais nada enquanto aguarda.

Ao chegar no final das instruções, o processo é então reiniciado, voltando para o início do **loop()**.

Você reparou que todo comando termina com um ponto e vírgula "**;**"? Ele é muito importante na programação porque define que o comando acabou ali. Se está faltando um "**;**" o programa não irá funcionar de jeito nenhum!

Possíveis erros

Caso o exercício não funcione, verifique alguns dos possíveis erros:

Na hora da checagem, apareceu alguma mensagem em vermelho? Verifique o código e caso não encontre o problema, copie e cole novamente o código na IDE Arduino e tente mais uma vez;

Verifique se os jumpers estão na mesma coluna dos terminais dos componentes, fazendo assim a conexão;

Verifique se os jumpers estão ligados nos pinos corretos no Arduino;

Verifique se o LED não está conectado invertido ou seja, terminal negativo no pino 11 e positivo no pino GND;

Verifique se o código carregou na placa através da IDE Arduino.

Desafios

Veja abaixo alguns desafios que você pode tentar!

Monte o circuito de uma maneira alternativa usando outros furos e posições na protoboard;

Mude o intervalo de tempo que o LED pisca mudando o valor na programação na linha **delay(1000);** por exemplo: **delay(250).**

Coloque um segundo conjunto de LED e resistor na protoboard, conecte a outra porta e altere o código para piscar os dois LEDs juntos (Lembre-se de colocar um segundo resistor para o novo LED)

Resposta do desafio

```
// CODIGO: Pisca-Pisca Desafio
// AUTOR: MAKERHERO



//----------------Funcao executada uma vez na inicializacao do sistema--------
--------
void setup() {
  pinMode(11, OUTPUT);            // Define a porta 11 como saida
  pinMode(10, OUTPUT);            // Define a porta 10 como saida
}
//----------------Funcao executada repetidamente enquanto o sistema estiver
ligado----------------

void loop() {


  digitalWrite(11, HIGH);         // Acende o led 1
  digitalWrite(10, HIGH);         // Acende o led 2
  delay(1000);                    // Aguarda intervalo de tempo entre
parenteses em milissegundos
  digitalWrite(11, LOW);          // Apaga o led 1
  digitalWrite(10, LOW);          // Apaga o led 2
```

```
  delay(1000);                    // Aguarda intervalo de tempo entre
parenteses em milissegundos
}
```

4.4 – SOS Luminoso

Você já deve ter visto nos filmes quando as pessoas enviam pedido de socorro SOS em código morse. Neste exercício iremos piscar um LED no padrão SOS. A letra **S** são 3 piscadas curtas indicando o ponto do código morse e a letra **O** são 3 piscadas um pouco mais longas indicando o traço do código morse.

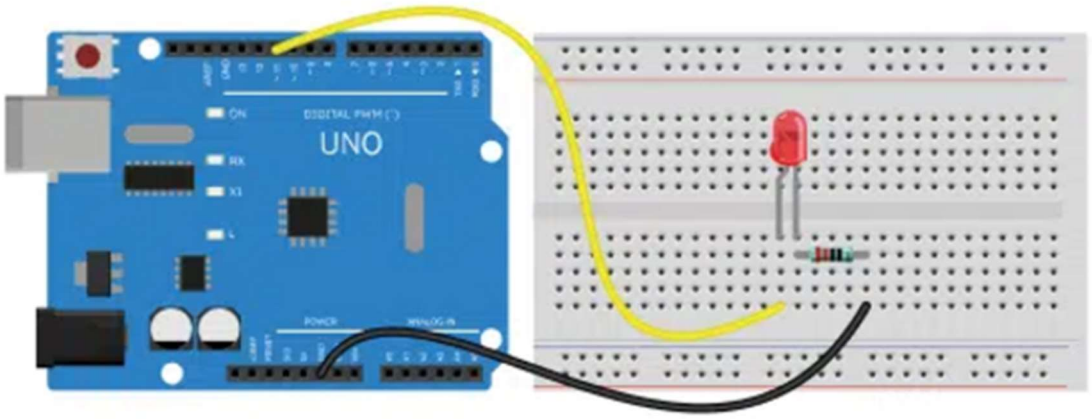
Este exercício mostra que, através da programação, podemos piscar o LED de diferentes maneiras e intervalos de tempo.

Material necessário

1x LED Vermelho 5mm
1x Resistor 220 ohm
1x Protoboard 400 pontos
2x Jumper macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

Nesse exercício vamos utilizar o mesmo circuito do pisca-pisca.



Programa

O programa, como feito com o Pisca-Pisca, acende e apaga a luz conforme um padrão. Como algumas coisas se repetem, foi acrescentada uma estrutura no código para repetição.

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: SOS Luminoso
// AUTOR: MAKERHERO
//----------------Declara as variaveis do codigo----------------
int pinoLed = 11;                // Declara a variável pinoLed e atribui o valor 11 a ela
//----------------Funcao executada uma vez na inicializacao do sistema----------------
void setup() {

 pinMode(pinoLed, OUTPUT);        // Configura a porta 11 (valor da variável pinoLed) como saida
}
//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------
void loop() {
 //----------------S(...) tres pontos----------------
 for(int x=0; x<3 ; x++) {        // Repete os comandos dentro das chaves 3 vezes
```

```
  digitalWrite(pinoLed,HIGH);       // Acende o LED
  delay(150);                  // Aguarda 150 milissegundos
  digitalWrite(pinoLed,LOW);       // Desliga o LED
  delay(150);                  // Aguarda 150 milissegundos
 }
 delay(200);                   // Aguarda 200 milissegundos entre as letras
 //----------------O(---) tres linhas----------------

 for(int x=0; x<3; x++) {          // Repete os comandos dentro das chaves 3 vezes
  digitalWrite(pinoLed,HIGH);       // Acende o LED
  delay(450);                  // Aguarda 450 milissegundos
  digitalWrite(pinoLed,LOW);       // Desliga o LED
  delay(150);                  // Aguarda 150 milissegundos
 }
 delay(200);                   // Aguarda 200 milissegundos entre as letras
 //----------------S(...) tres pontos----------------
 for(int x=0; x<3; x++) {          // Repete os comandos dentro das chaves 3 vezes
  digitalWrite(pinoLed,HIGH);       // Acende o LED
  delay(150);                  // Aguarda  150 milissegundos
  digitalWrite(pinoLed,LOW);       // Desliga o LED
  delay(150);                  // Aguarda 150 milissegundos
 }

 delay(5000);                   // Aguarda 5000 milissegundos (5 segundos)
}
```

A ideia não é entender a fundo como as estruturas de repetição funcionam, mas sim que elas existem e seu funcionamento básico.

Perceba também que declaramos a variável pinoLed antes do **setup()**, essa variável pode ser usada por todas as funções do programa. Caso tivéssemos declarado ela dentro da função **setup()**, a função **loop()** não "conheceria" essa variável.

Possíveis erros

Caso o exercício não funcione verifique alguns dos possíveis erros:

- Verifique se os jumpers estão na mesma coluna dos terminais dos componentes, fazendo assim a conexão;
- Verifique se os jumpers estão ligados nos pinos corretos no Arduino;
- Verifique se o LED não está conectado invertido ou seja, terminal negativo no pino 11 e positivo no pino GND;
- Verifique se o código carregou na placa através da IDE Arduino.

Essa estrutura é o **for()**, que é uma função especial.

```
for(int x=0; x<3 ; x++) {          // Repete os comandos dentro das chaves 3 vezes
  digitalWrite(pinoLed,HIGH);      // Acende o LED
  delay(150);                      // Aguarda 150 milissegundos
  digitalWrite(pinoLed,LOW);       // Desliga o LED
  delay(100);                      // Aguarda 100 milissegundos
}
```

Ela recebe três parâmetros, separados por "**;**".

O primeiro é a variável na qual vamos fazer as repetições, nesse caso, x, que começa em zero, o segundo é o intervalo das repetições, nesse caso enquanto x for menor que 3, e o terceiro indica o incremento ao final de cada repetição. **x++** na linguagem do Arduino significa aumentar o valor da variável em uma unidade.

Então vamos, ao chamar o for, acender o LED, aguardar 150 milissegundos, apagar o LED, aguardar 100 milissegundos e verificar se x ainda está dentro do intervalo. Enquanto x estiver dentro do intervalo, o programa irá repetir as instruções dentro do for.

4.5 – Brilho Oscilante

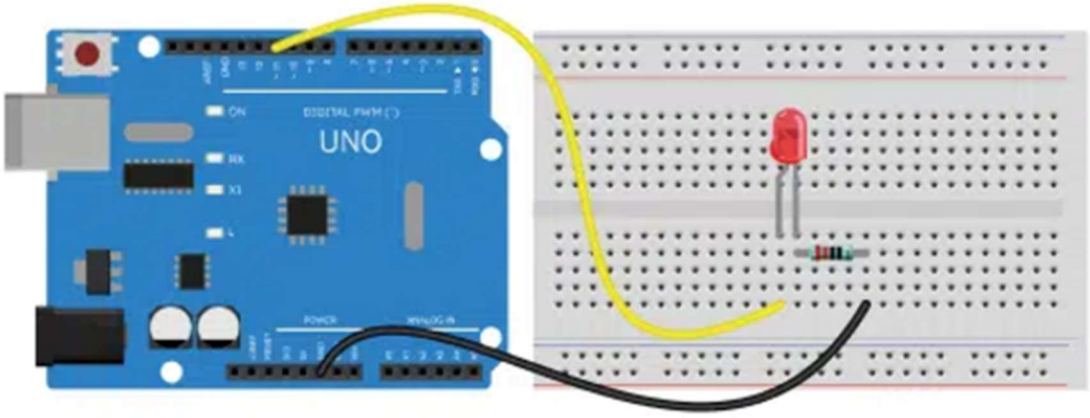
Nos exercícios anteriores vimos como ligar e desligar o nosso LED. Mas e se quiséssemos variar o brilho dessa luz? Com diversos componentes, às vezes queremos variar a sua intensidade, e não apenas ligar ou desligar. Com um motor, por exemplo, podemos querer variar a sua velocidade. Para isso usaremos uma função diferente do Arduino, a sua saída analógica.

Material necessário

1x LED Vermelho 5mm
1x Resistor 220 ohm
1x Protoboard 400 pontos
2x Jumper macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

Neste exercício continuaremos com o mesmo circuito dos exercícios anteriores.



Programa

Nesse programa utilizaremos a função **analogWrite()**, que escreve valores analógicos na saída do Arduino.

```
31    analogWrite(pinoLed, valor);
```

A função recebe como parâmetros o pino que deve ser acionado e o valor, que fica entre **0 e 255**. Esse valor que pode parecer estranho e aleatório é devido às características do Arduino,

mas a explicação do porquê é um pouco mais avançada, e basta você saber que o valor pode ser de 0 a 255 para conseguir utilizar a função, ok?

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Brilho Oscilante

// AUTOR: MAKERHERO

  //----------------Declara as variaveis do codigo----------------

 int pinoLed = 11;                              // Declara a variável pinoLed e atribui o valor 11 a ela

 int incremento = 5;                            // Declara a variável incremento e atribui o valor 5 a ela

 int decremento = 5;                            // Declara a variável decremento e atribui o valor 5 a ela

 //----------------Funcao executada uma vez na inicializacao do sistema----------------

 void setup() {

   pinMode(pinoLed,OUTPUT);                     // Configura a porta 11 (valor da variável pinoLed) como saida

 }


//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------

 void loop() {

    //----------------Aumenta o brilho usando o laco for----------------

   for (byte valor = 0; valor < 255; valor+=incremento) {       // Repete os comandos dentro das chaves 255 vezes

     analogWrite(pinoLed, valor);               // Controla o brilho no pino do LED

       delay(30);                               // Aguarda 30 milissegundos

   }


  //----------------Diminui o brilho usando o laco for----------------

   for (byte valor = 255; valor >0; valor-=decremento) {    // Repete os comandos dentro das chaves 255 vezes

       analogWrite(pinoLed, valor);             // Controla o brilho no pino do LED
```

```
  delay(30);                          // Aguarda 30 milissegundos

 }
}
```

Dica: Percebeu o **+=** e **-=** nas funções for? Isso é uma forma encurtada de dizer que estamos somando ou subtraindo o valor depois do igual à primeira variável e atualizando ela com esse valor. Fazendo por extenso a estrutura completa seria valor = valor + incremento

Possíveis erros

Caso o exercício não funcione, verifique alguns dos possíveis erros:

- Verifique se os jumpers estão na mesma coluna dos terminais dos componentes, fazendo assim a conexão;
- Verifique se os jumpers estão ligados nos pinos corretos no Arduino;
- Verifique se o **LED** não está conectado invertido, ou seja, terminal negativo no **pino 11** e positivo no pino **GND**;
- Verifique se o código carregou na placa através da IDE Arduino.

4.6 – Projeto: Semáforo

Já está sabendo tudo de LEDs e como ligá-los? Então vamos montar esse projeto maior para juntar várias coisas que aprendemos até agora.

Este projeto irá simular um semáforo de carros. Serão três LEDs para reproduzir as três cores. A sequência inicia com o LED verde dos carros aceso, depois de um determinado intervalo passa para cor amarela por alguns instantes e depois para cor vermelha e então a sequência volta ao início.

Material necessário

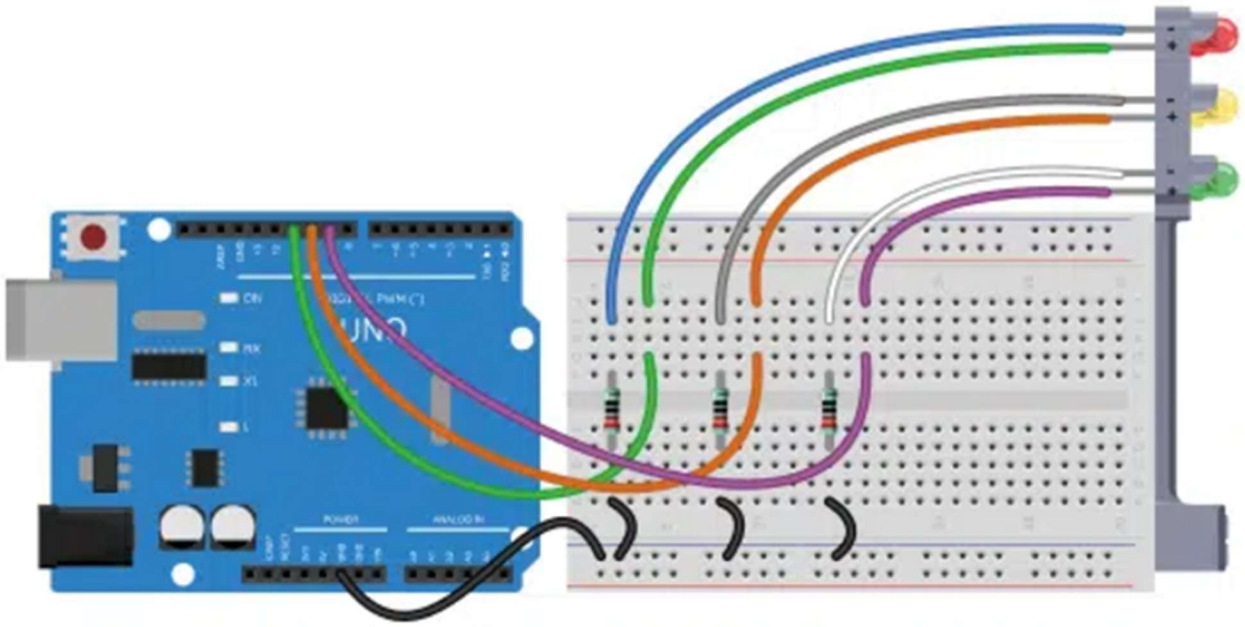
- 1x LED Vermelho 5mm
- 1x LED Verde 5mm
- 1x LED Amarelo 5mm

- 1x Semáforo 3D
- 3x Resistor 220 ohm
- 1x Protoboard 400 pontos
- 7x Jumper macho-macho
- 6x Jumper macho-fêmea
- 1x Cabo USB
- 1x Placa Uno

Montagem do circuito

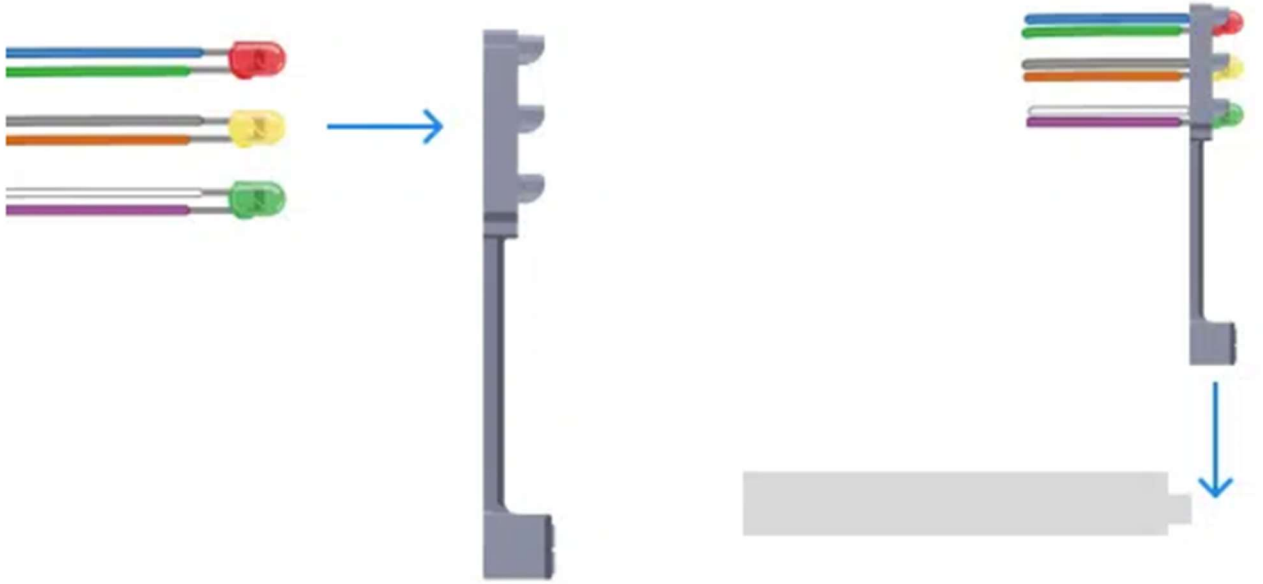
O grande desafio deste projeto está em montar os LEDs corretamente, pois o circuito consiste de mais LEDs, mais resistores e mais jumpers. Agora faremos uso de uma das linhas inferiores da protoboard, conectando o pino GND (negativo) do Arduino na linha azul da protoboard.

Lembre-se que o pino mais comprido do LED (terminal positivo) será conectado às saídas do Arduino e o pino mais curto (terminal negativo) será conectado ao resistor que por sua vez estará conectado ao GND. Procure deixar todos os LEDs na mesma posição ao encaixar no semáforo para facilitar as conexões



Montagem do semáforo

Para encaixar os LEDs na peça de impressão 3D do semáforo siga as seguintes instruções. Conecte os jumpers nos LEDs e depois os encaixe no semáforo no sentido das costas do semáforo para a frente. Para encaixar o semáforo na protoboard, encaixe os rebaixos da peça nos pinos de encaixe da protoboard.



A maioria das pessoas, mesmo makers experientes, tem que voltar para esta parte do projeto e refazer, não se preocupe. Se você conseguir fazer a montagem funcionar na primeira tentativa, parabéns!

Programa

Esse código, apesar de parecer extenso, é simples. Não tem nada que ainda não vimos neste guia. Procure entender o código e ver o que irá acontecer a partir dele.

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

// CODIGO: Semaforo

// AUTOR: MAKERHERO

//----------------Declara as variaveis do codigo----------------

```
int carroVerde = 9;                 // Declara a variável carroVerde e atribui o valor 9 a ela
int carroAmarelo = 10;              // Declara a variável carroAmarelo e atribui o valor 10 a ela
int carroVermelho = 11;             // Declara a variável carroVermelho e atribui o valor 11 a ela

//----------------Funcao executada uma vez na inicializacao do sistema----------------
void setup() {
 pinMode(carroVerde, OUTPUT);          // Configura a porta 9 (valor da variavel carroVerde) como saida
 pinMode(carroAmarelo, OUTPUT);        // Configura a porta 10 (valor da variavel carroAmarelo) como saida
 pinMode(carroVermelho, OUTPUT);       // Configura a porta 11 (valor da variavel carroVermelho) como saida
 // Coloca o semaforo na posição inicial.
 digitalWrite(carroVerde, LOW);        // Apaga o LED Verde do semaforo
 digitalWrite(carroAmarelo, LOW);      // Apaga o LED Amarelo do semaforo
 digitalWrite(carroVermelho, HIGH);    // Acende o LED Vermelho do semaforo
}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------

void loop() {
 digitalWrite(carroVerde, HIGH);       // Acende o LED Verde do semaforo
 digitalWrite(carroVermelho, LOW);     // Apaga o LED Vermelho do semaforo
 delay(5000);                          // Aguarda 5000 milissegundos (5 segundos)
 digitalWrite(carroVerde, LOW);        // Apaga o LED Verde do semaforo
 digitalWrite(carroAmarelo, HIGH);     // Acende o LED Amarelo do semaforo
 delay(3000);                          // Aguarda 3000 milissegundos (3 segundos)
 digitalWrite(carroAmarelo, LOW);      // Apaga o LED Amarelo do semaforo
```

```
 digitalWrite(carroVermelho, HIGH);       // Acende o LED Vermelho do semaforo
 delay(5000);                    // Aguarda 5000 milissegundos (5 segundos)
}
```

Como todo código, começamos definindo as variáveis, os pinos utilizados e, por fim, o **loop()** que fica repetindo continuamente.

Se estiver com problemas para entender, você pode ler os comentários dentro do programa e/ou voltar nos exercícios Pisca-Pisca e SOS Luminoso.

Conseguiu pegar a lógica do programa? Você pode alterá-la conforme bem entender e fazer a sua própria lógica de semáforo!

Possíveis erros

Caso o projeto não funcione verifique alguns dos possíveis erros:

- Verifique se os jumpers estão na mesma coluna dos terminais dos componentes, fazendo assim a conexão;
- Verifique se os jumpers estão ligados nos pinos corretos no Arduino;
- Verifique se o LED não está conectado invertido;
- Verifique se o código carregou na placa através da IDE Arduino.

4.7 – Luzes Coloridas

Agora vamos usar o LED RGB, que é um 3 em 1, temos as três cores primárias em um único LED.

Este exercício consiste em alternar entre as 3 cores do LED RGB através do Arduino em um intervalo de 1 segundo.

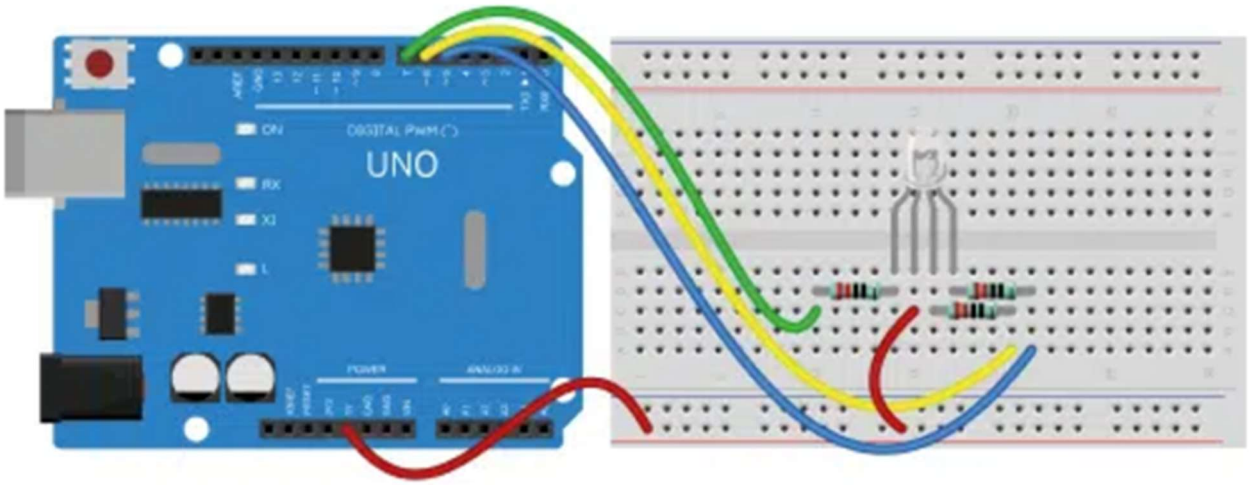
Material necessário

- 1x LED RGB 5mm
- 3x Resistor 220 ohm
- 5x Jumper Macho-macho

- 1x Protoboard 400 pontos
- 1x Cabo USB
- 1x Placa Uno

Montagem do circuito

A montagem deste circuito é feita como se fossem três LEDs colocados no mesmo exercício. Adicione um resistor para cada perna de cor do LED RGB e conecte o pino **5V** do Arduino à perna positiva do LED RGB. Cada pino de cor do LED é conectado a uma porta digital do Arduino.



Importante! Note que o fio vermelho (**5V**) deve ser conectado à perna mais comprida do LED.

Programa

O programa consiste em alternar cada uma das cores do LED em um intervalo de 1 segundo.

Nesse programa vamos ter várias linhas muito parecidas, e que podem dificultar para visualizar o que cada linha faz, então vamos separar elas em funções para entender mais fácil o que o programa está fazendo

```
void acendeVermelho(){
  digitalWrite(led_R, LOW);      // Acende a cor Vermelha do LED
  digitalWrite(led_G, HIGH);     // Apaga a cor Verde do LED
```

```
  digitalWrite(led_B, HIGH);     // Apaga a cor Azul do LED
}
```

A função **acendeVermelho()** não recebe nenhum parâmetro e quando é chamada acende o LED vermelho, usando o **digitalWrite()** que já vimos para deixar a porta do LED vermelho em LOW (lembre que o LED RGB o positivo é comum entre os três LEDs, então precisamos ligar cada pino ao GND para que a respectiva cor acenda) e os demais em HIGH.

Repetimos o mesmo para as outras cores e então no **loop()** precisamos chamar apenas a função de acender as cores, ao invés de termos um monte de **digitalWrite()** juntos.

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Luzes Coloridas
// AUTOR: MAKERHERO
//----------------Declara as variaveis do codigo----------------
int led_R = 7;              // Declara a variável led_R e atribui o valor 7 a ela (R)
int led_G = 6;              // Declara a variável led_G e atribui o valor 6 a ela (G)
int led_B = 5;              // Declara a variável led_B e atribui o valor 5 a ela (B)
//----------------Declaracao da funcao que acende a cor Vermelha do LED----------------
void acendeVermelho(){
  digitalWrite(led_R, LOW);     // Acende a cor Vermelha do LED
  digitalWrite(led_G, HIGH);    // Apaga a cor Verde do LED
  digitalWrite(led_B, HIGH);    // Apaga a cor Azul do LED
}

//----------------Declaracao da funcao que acende a cor Verde do LED----------------
void acendeVerde() {
  digitalWrite(led_R, HIGH);    // Apaga a cor Vermelha do LED
  digitalWrite(led_G, LOW);     // Acende a cor Verde do LED
```

```
  digitalWrite(led_B, HIGH);    // Apaga a cor Azul do LED
}

//----------------Declaracao da funcao que acende a cor Azul do LED----------------

void acendeAzul() {
  digitalWrite(led_R, HIGH);    // Apaga a cor Vermelha do LED
  digitalWrite(led_G, HIGH);    // Apaga a cor Verde do LED
  digitalWrite(led_B, LOW);     // Acende a cor Azul do LED
}

//----------------Declaracao da funcao que apaga todas as cores do LED----------------
void apagaLed() {
  digitalWrite(led_R, HIGH);    // Apaga a cor Vermelha do LED
  digitalWrite(led_G, HIGH);    // Apaga a cor Verde do LED
  digitalWrite(led_B, HIGH);    // Apaga a cor Azul do LED
}

//----------------Funcao executada uma vez na inicializacao do sistema----------------

void setup() {
  pinMode(led_R, OUTPUT);       // Configura a porta 7 (valor da variável led_R) como saida
  pinMode(led_G, OUTPUT);       // Configura a porta 6 (valor da variável led_G) como saida
  pinMode(led_B, OUTPUT);       // Configura a porta 5 (valor da variável led_B) como saida
  apagaLed();                   // Chama a funcao de apagar todas as cores do LED
}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------
```

```
void loop() {
  acendeVermelho();          // Chama a funcao de acender a cor Vermelha do LED e apagar as demais
  delay(1000);               // Aguarda 1000 milissegundos (1 segundo)

  acendeVerde();             // Chama a funcao de acender a cor Verde do LED e apagar as demais
  delay(1000);               // Aguarda 1000 milissegundos (1 segundo)
  acendeAzul();              // Chama a funcao de acender a cor Azul do LED e apagar as demais
  delay(1000);               // Aguarda 1000 milissegundos (1 segundo)
  apagaLed();                // Chama a funcao de apagar todas as cores do LED
  delay(1000);               // Aguarda 1000 milissegundos (1 segundo)
}
```

Possíveis erros

Caso o exercício não funcione verifique alguns dos possíveis erros:

- Verifique se os jumpers estão na mesma coluna dos terminais dos componentes, fazendo assim a conexão;
- Verifique se os jumpers estão ligados nos pinos corretos no Arduino;
- Verifique se o terminal positivo do LED RGB está na posição correta ligado ao 5 V do Arduino;
- Verifique se o código carregou na placa através da IDE Arduino.

4.8 – Troque a Cor das Luzes

Neste exercício você irá comandar como o LED irá variar a sua cor. Para dar os comandos para a placa Arduino, iremos utilizar um potenciômetro.

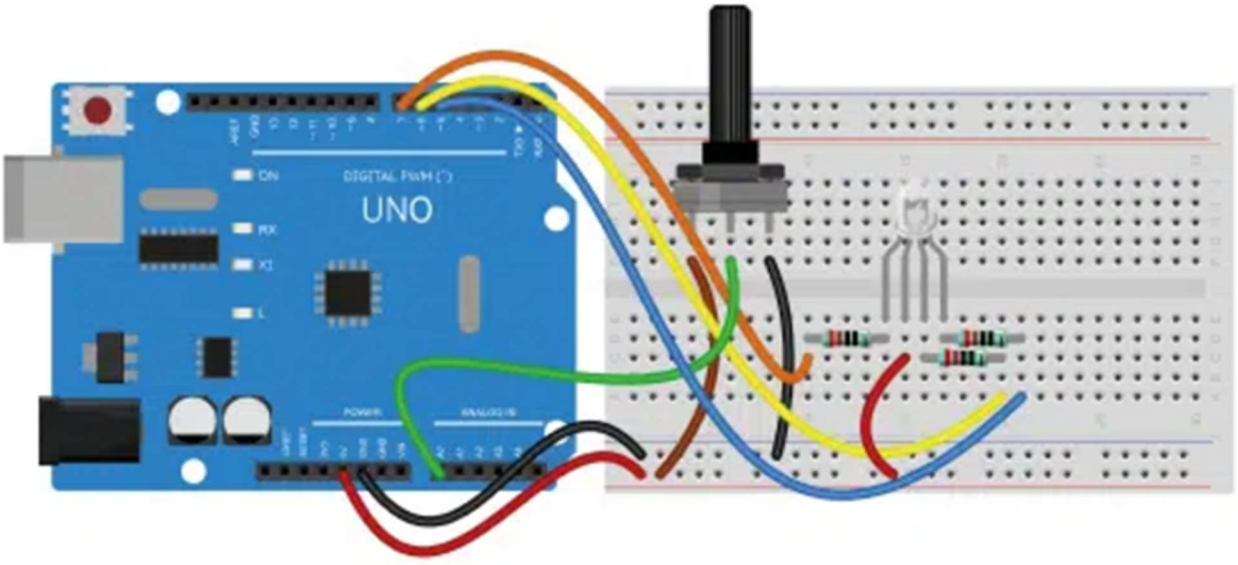
Esse exemplo consiste em controlar as cores do LED RGB ao rotacionar o potenciômetro.

Material necessário

1x LED RGB 5mm
1x Potenciômetro 10K ohm
3x Resistor 220 ohm
1x Protoboard 400 pontos
9x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

A montagem do LED RGB é a mesma do exercício anterior. Adicione o potenciômetro ligando um pino em 5 V, outro em GND e seu pino central em uma porta analógica do Arduino. Não se preocupe com o lado, assim como um resistor comum, o potenciômetro não tem lado positivo ou negativo, ou seja, tanto faz o lado que é conectado ao circuito. Mas diferente de um resistor que tem apenas 2 terminais, dependendo do sentido que colocarmos o potenciômetro, os valores da leitura irão crescer em sentidos diferentes. O único cuidado que se deve ter ao colocar o potenciômetro na protoboard é o encaixe, que tem que apertar um pouco.



Programa

Para ler o valor de um potenciômetro ou qualquer componente que tenha um sinal variável, fazemos uso da função **analogRead()** que transforma a rotação do potenciômetro em valores de 0 a 1023. Esse valor é diferente de quando escrevemos com o **analogWrite()** também devido a construção do Arduino, e a explicação também é um pouco mais avançada do que iremos ver nesse curso.

```
81  valorPot = analogRead(pot); // lê o valor do potenciômetro (de 0 a 1023)
```

A função **analogRead()** irá ler o valor que recebe no pino passado como parâmetro, que está salvo na variável pot e salvará o valor da leitura na variável valorPot.

Para decidir qual cor será acesa com qual valor, utilizaremos a função **if()**. Ela verificará se uma determinada condição foi atendida, e se foi, irá executar as instruções que estão entre chaves

```
if(valorPot > 256 && valorPot <= 512) // entre 256 e 512, acende vermelho
 {
   acendeVermelho();
 }
```

Caso a condição não seja atendida, o programa irá ignorar as instruções dentro do **if()** e seguirá com as próximas linhas do programa.

Dentro do **if()** podemos usar as condições **==**, para comparar se uma coisa é igual a outra, **>** ou **<** para comparar se é maior ou menor, **>=** ou **<=** para comparar se algo é menor ou igual.

Também podemos, como acima, verificar mais de uma condição, com **&&** ou **||**. **&&** somente atenderá a condição e executará as instruções do **if()** se as duas condições forem verdadeiras. Já o **||** basta que uma das duas condições sejam verdadeiras para executar as instruções do **if()**

O programa consiste em alternar as cores do LED RGB conforme o potenciômetro é rotacionado, lendo o valor do mesmo através da função **analogRead()** e decidindo qual cor do LED acender com a função **if()**.

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Troque a Cor das Luzes
// AUTOR: MAKERHERO


//----------------Declara as variaveis do codigo----------------
int led_R = 7;                  // Declara a variável led_R e atribui o valor 7 a ela (R)
int led_G = 6;                  // Declara a variável led_G e atribui o valor 6 a ela (G)
```

```
int led_B = 5;                    // Declara a variável led_B e atribui o valor 5 a ela (G)
int pot = A0;                     // Declara a variável pot e atribui o valor A0 a ela
int valorPot;                     // Declara a variável valorPot sem atribuir valor a ela


//----------------Declaracao da funcao que acende a cor Vermelha do LED----------------

void acendeVermelho() {
  digitalWrite(led_R, LOW);       // Acende a cor Vermelha do LED
  digitalWrite(led_G, HIGH);      // Apaga a cor Verde do LED
  digitalWrite(led_B, HIGH);      // Apaga a cor Azul do LED

}
//----------------Declaracao da funcao que acende a cor Verde do LED----------------
void acendeVerde() {
  digitalWrite(led_R, HIGH);      // Apaga a cor Vermelha do LED
  digitalWrite(led_G, LOW);       // Acende a cor Verde do LED

  digitalWrite(led_B, HIGH);      // Apaga a cor Azul do LED
}
//----------------Declaracao da funcao que acende a cor Azul do LED----------------
void acendeAzul() {
  digitalWrite(led_R, HIGH);      // Apaga a cor Vermelha do LED
  digitalWrite(led_G, HIGH);      // Apaga a cor Verde do LED
  digitalWrite(led_B, LOW);       // Acende a cor Azul do LED

}
//----------------Declaracao da funcao que apaga todas as cores do LED----------------
```

```
void apagaLed() {
  digitalWrite(led_R, HIGH);                    // Apaga a cor Vermelha do LED
  digitalWrite(led_G, HIGH);                    // Apaga a cor Verde do LED
  digitalWrite(led_B, HIGH);                    // Apaga a cor Azul do LED
 }
//----------------Funcao executada uma vez na inicializacao do sistema----------------

void setup() {
  pinMode(led_R, OUTPUT);                    // Configura a porta 7 (valor da variável led_R) como saida
  pinMode(led_G, OUTPUT);                    // Configura a porta 6 (valor da variável led_G) como saida
  pinMode(led_B, OUTPUT);                    // Configura a porta 5 (valor da variável led_B) como saida
  apagaLed();                              // Chama a funcao de apagar todas as cores do LED
}
//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------
void loop() {
  valorPot = analogRead(pot);                    // Lê o valor do potenciômetro (de 0 a 1023) e atribui o valor lido
a variavel valorPot
  if(valorPot >= 0 && valorPot <= 256) {          // Se o valor de valorPot for menor ou igual a 256, executa as
instrucoes entre chaves
    apagaLed();                              // Chama a funcao de apagar todas as cores do LED

  }

  if(valorPot > 256 && valorPot <= 512) {          // Se o valor estiver entre 257 e 512, executa as instrucoes
entre chaves
    acendeVermelho();                        // Chama a funcao de acender a cor Vermelha do LED e apagar as
demais
  }
```

```
  if(valorPot > 512 && valorPot <= 768) {          // Se o valor estiver entre 513 e 768, executa as instrucoes entre chaves
    acendeVerde();                          // Chama a funcao de acender a cor Verde do LED e apagar as demais
  }
  if(valorPot > 768 && valorPot <= 1023) {          // Se o valor estiver entre 769 e 1023, executa as instrucoes entre chaves
    acendeAzul();                          // Chama a funcao de acender a cor Azul do LED e apagar as demais
  }
}
```

Possíveis erros

Caso o exercício não funcione verifique alguns dos possíveis erros:

- Verifique se os jumpers estão na mesma coluna dos terminais dos componentes, fazendo assim a conexão;
- Verifique se os jumpers estão ligados nos pinos corretos no Arduino;
- Verifique se o terminal positivo do LED RGB está na posição correta, ligado ao 5 V do Arduino;
- Verifique se os pinos do potenciômetro estão totalmente inseridos nos furos da protoboard;
- Verifique se o código carregou na placa através da IDE Arduino.

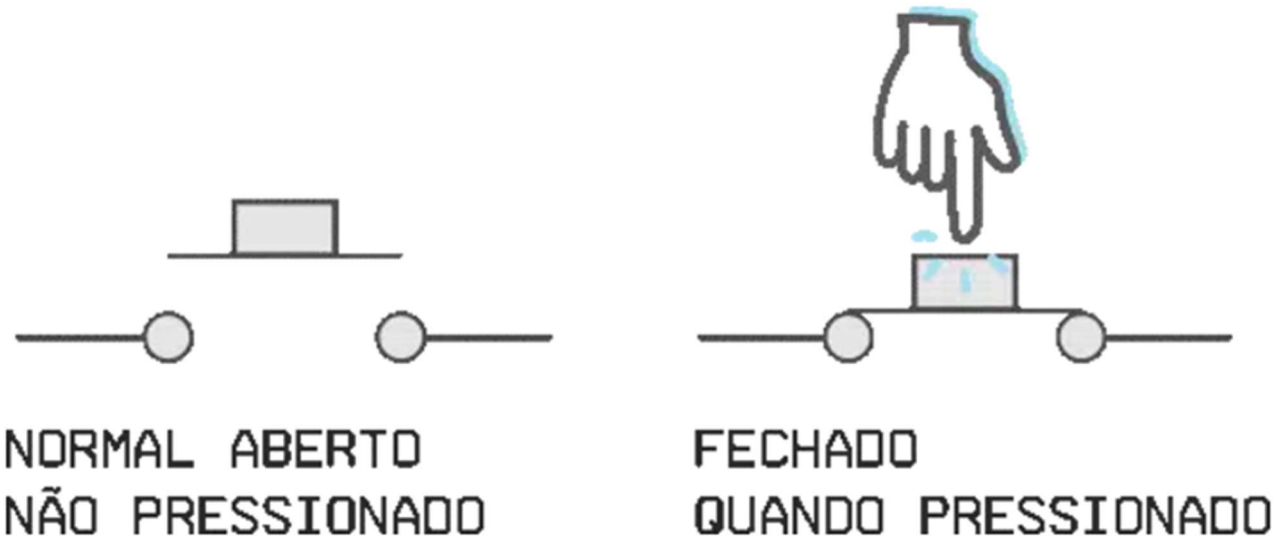
5 – Botão

Aprenda a interagir com o Arduino através de botões, de forma a atuar sobre a programação e controlar os projetos sem ser apenas trocando a programação.

5.1 – Conhecendo o botão

Praticamente todos os equipamentos eletrônicos terão pelo menos um botão para ligar. O botão é usado para permitir interação do usuário com a máquina, seja para a operação direta ou para realizar ajustes.

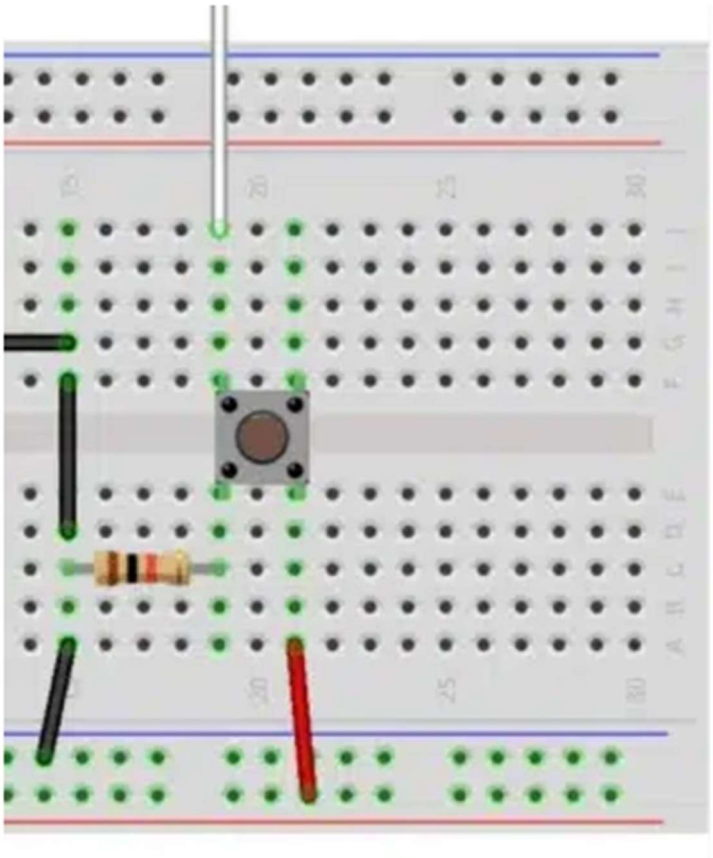
Também conhecido como chave push-button, funciona como um contato que abre e fecha, sendo assim, uma chave possui dois valores, 0 ou 1, aberto ou fechado. Conectando uma chave a uma porta do Arduino podemos ler o valor 0 ou 1 da chave e assim tomar uma ação, que no caso do nosso exemplo será acionar o LED. Por só ter dois valores possíveis, utilizaremos uma das portas digitais da placa.



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, decidiremos o que fazer com essa informação.

O botão que vem com o seu kit tem 4 terminais, ligados dois a dois. Então quando não pressionamos o botão, haverá um par de pinos de cada lado conectados entre si. Quando apertamos o botão, os dois pares são conectados, de forma que qualquer um dos quatro terminais do botão estará conectado no mesmo ponto, como ilustrado na figura abaixo.

Por isso é importante prestar atenção na posição que o botão precisa ser encaixado na protoboard.



5.2 – Interruptor de Luz Intermitente

Neste exercício, veremos que também é possível usar botões para interagir com o arduino além de usar o potenciômetro. Faremos um LED acender enquanto você mantém um botão pressionado.

Mas você pode estar se perguntando “pra que eu preciso do Arduino para fazer isso? Só o botão ligado ao LED faria a mesma coisa!”. É verdade, poderíamos fazer o mesmo comportamento sem o Arduino ligando o botão diretamente ao LED, mas precisamos construir aos poucos o entendimento de como o Arduino lida com entradas e saídas e o que é possível fazer com elas.

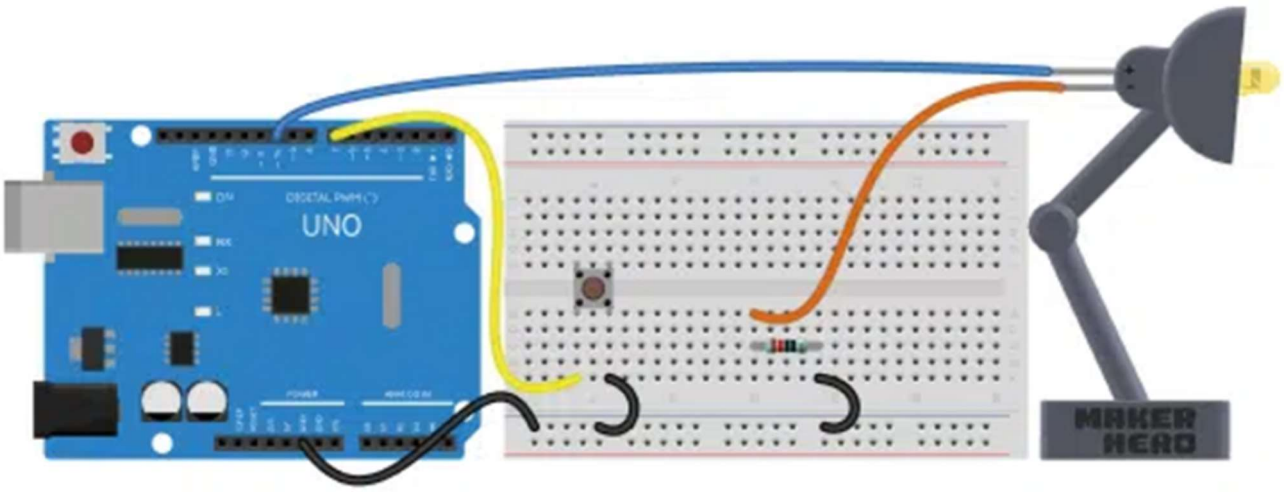
Imagine que no lugar do LED você coloque um motor, você não quer que seu carrinho continue andando quando você parar de dizer para ele ir pra frente, quer?

Material necessário

1x LED Amarelo 5mm
1x Luminária 3D
1x Resistor 220 ohm
1x Chave Táctil Push-button
1x Protoboard 400 pontos
4x Jumper Macho-macho
2x Jumper Macho-fêmea
1x Cabo USB
1x Placa Uno

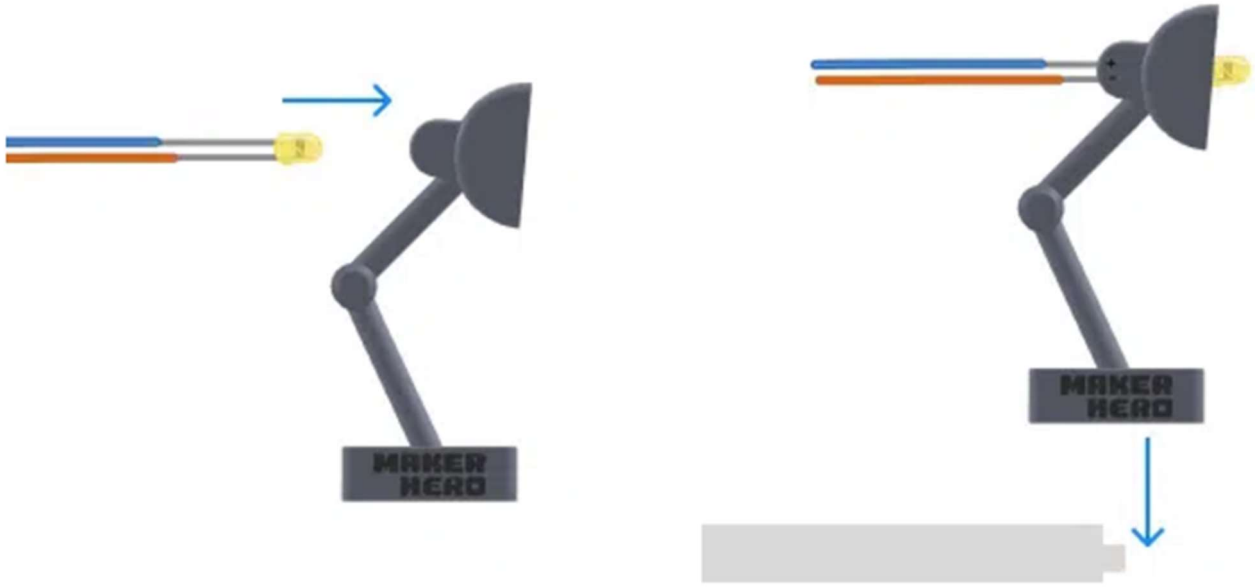
Montagem do circuito

A chave táctil push button deve ser montada com dois terminais acima da cavidade central da protoboard e os outros dois abaixo. Perceba que deixamos os terminais que saem da mesma face do botão para o mesmo lado da cavidade central. Um terminal da chave vai no pino 7 e outro no GND. No caso deste exercício o pino do LED é uma saída digital e o pino da chave é uma entrada digital.



Montagem da luminária

Para encaixar o LED na peça de impressão 3D da luminária siga as seguintes instruções. Conecte os jumpers no LED e depois encaixe-o na luminária no sentido das costas da lumiária para a frente. Para encaixar a luminária na protoboard, encaixe os rebaixos da peça nos pinos de encaixe da protoboard.



Programa

Para ler o valor digital de um botão, primeiro temos que configurar o pino como entrada utilizando **pinMode(botao, INPUT_PULLUP);** e então fazer a leitura com **digitalRead(botao);** Sendo o valor da leitura 0 ou 1 (LOW ou HIGH), controlamos o LED de acordo.

```
21  pinMode(pinoBotao, INPUT_PULLUP);          // Configura a porta 7 (valor da variável pinoBotao) como
```

Como o botão, quando não está pressionado, não conecta a porta a nada, passamos o parâmetro INPUT_PULLUP à função **pinMode()** para que o Arduino, internamente, ligue o pino onde ligamos o botão ao HIGH, dessa forma não deixando ocorrer variações indesejadas no sinal.

Diferente do **digitalWrite()**, onde passamos a porta e queremos que ela fique ligada ou desligada, o **digitalRead()** passamos apenas o número da porta, e vamos ler se o que está conectado a ela está ligado ou desligado.

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Interruptor de Luz Intermitente
// AUTOR: MAKERHERO

//----------------Declara as variaveis do codigo----------------
int botao = 7;                  // Declara a variável botao e atribui o valor 7 a ela
int led = 10;                   // Declara a variável led e atribui o valor 10 a ela
bool estadoLed = LOW;           // Declara a variável estadoLed e atribui o valor LOW a ela
bool estadoBotao = LOW;         // Declara a variável estadoBotao e atribui o valor LOW a ela
//----------------Funcao executada uma vez na inicializacao do sistema----------------
void setup() {
 pinMode(botao, INPUT_PULLUP);      // Configura a porta 7 (valor da variável botao) como entrada com resistor de Pull-Up integrado
 pinMode(led, OUTPUT);              // Configura a porta 10 (valor da variável led) como saida
}
//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------

void loop() {
 estadoBotao = digitalRead(botao);   // Le o estado do botao (HIGH ou LOW) e atribui o valor a variavel estadoBotao
 estadoLed =
!estadoBotao;          // Atribui o
valor da variavel estadoBotao a
variavel estadoLed
 digitalWrite(led, estadoLed);      // Apaga ou acende o LED de acordo com o valor da variavel estadoLed (HIGH faz o LED acender e LOW faz o LED apagar)
 delay(10);                         // Aguarda 10 milissegundos

}
```

Possíveis erros

- Verifique se os jumpers estão na mesma coluna dos terminais dos componentes, fazendo assim a conexão;
- Verifique se os jumpers estão ligados nos pinos corretos no Arduino;
- Verifique se o LED não está conectado invertido;
- Verifique se a posição do botão está correta;
- Verifique se os pinos do botão estão bem inseridos na protoboard;
- Verifique se o código carregou na placa através da IDE Arduino.

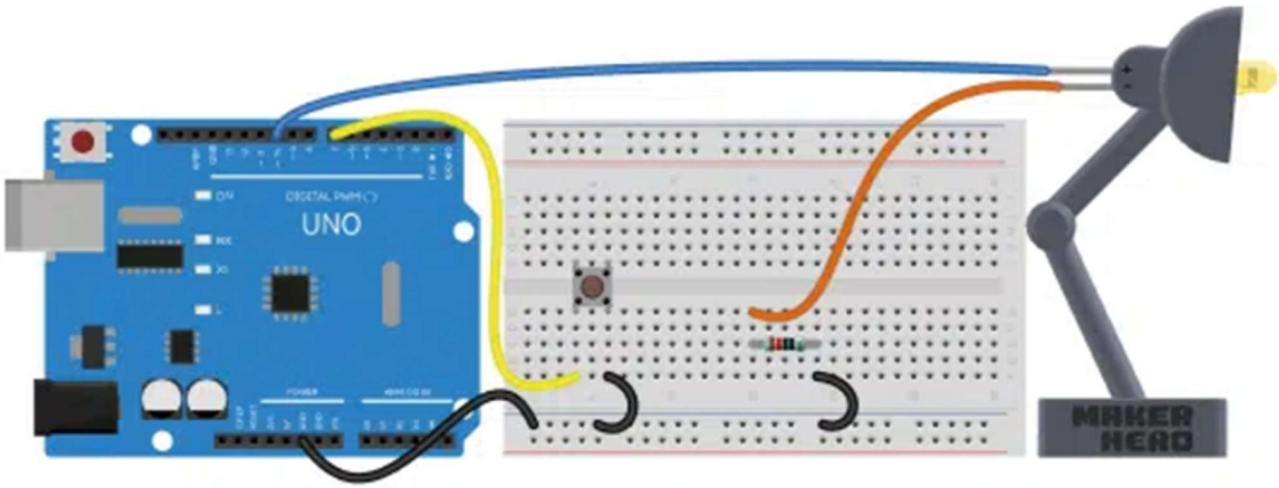
5.3 – Interruptor de Luz Liga-Desliga

Agora vamos manter o led aceso até pressionarmos novamente o botão, dessa vez algo que só esse botão ligado ao LED não conseguiria fazer.

Isso mostra que o Arduino pode “lembrar” do último estado do sistema, e mudar de estado conforme um sinal externo.

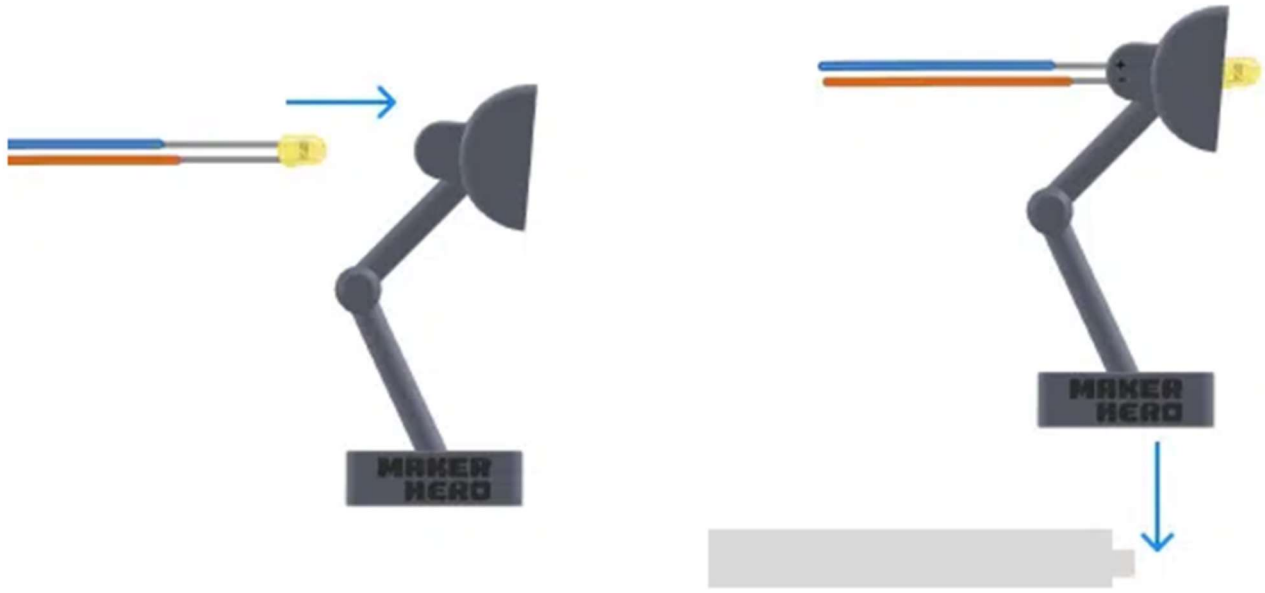
Material necessário
1x LED Amarelo 5mm
1x Luminária 3D
1x Resistor 220 ohm
1x Chave Táctil Push-button
1x Protoboard 400 pontos
5x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito
A montagem é igual ao do exercício anterior.



Montagem da luminária

Para encaixar o LED na peça de impressão 3D da luminária siga as seguintes instruções. Conecte os jumpers no LED e depois encaixe-o na luminária no sentido das costas da lumiária para a frente. Para encaixar a luminária na protoboard, encaixe os rebaixos da peça nos pinos de encaixe da protoboard.



Programa

Nesse exercício vamos usar a função **while**() para manter o arduino parado enquanto não soltarmos o botão, para que não haja trocas seguidas de estado se o botão continuar apertado.

```
33          while(digitalRead(pinoBotao) == LOW) {          // Enquanto o botao estiver apertado (leitura
34
35           }
```

Como não botamos nada entre as chaves, a função não fará nada enquanto o botão estiver apertado, mas podemos também colocar instruções dentro do while. É mais uma estrutura de laço como o **for**(), mas mais usada quando não sabemos de antemão quantas vezes precisaremos executá-la

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Interruptor de Luz Liga Desliga
// AUTOR: MAKERHERO
//----------------Declara as variaveis do codigo----------------

int pinoBotao = 7;                    // Declara a variável pinoBotao e atribui o valor 7 a ela

int pinoLed = 10;                     // Declara a variável pinoLed e atribui o valor 10 a ela

bool estadoLed = LOW;                 // Declara a variável estadoLed e atribui o valor LOW a
ela

//----------------Funcao executada uma vez na inicializacao do sistema----------------

void setup() {

 pinMode(pinoBotao, INPUT_PULLUP);          // Configura a porta 7 (valor da variável
pinoBotao) como entrada com resistor de Pull-Up integrado
 pinMode(pinoLed, OUTPUT);                  // Configura a porta 10 (valor da variável pinoLed)
como saida
}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------
--

void loop() {
```

```
  if(digitalRead(pinoBotao) == LOW) {          // Se o botao estiver apertado (leitura igual a LOW), executa as instrucoes entre chaves

    estadoLed = !estadoLed;                    // Troca o estado da variavel estadoLed (Se estiver HIGH passa a ser LOW e vice versa)

    digitalWrite(pinoLed, estadoLed);          // Apaga ou acende o LED de acordo com o valor da variavel estadoLed (HIGH faz o LED acender e LOW faz o LED apagar)

    while(digitalRead(pinoBotao) == LOW) {     // Enquanto o botao estiver apertado (leitura igual a LOW), nao faz nada, pois nao ha instrucoes entre as chaves

    }

    delay(100);                                // Aguarda 100 milissegundos

  }

}
```

Possíveis erros

Caso o exercício não funcione, verifique alguns dos possíveis erros:

- Verifique se os jumpers estão na mesma coluna dos terminais dos componentes, fazendo assim a conexão;
- Verifique se os jumpers estão ligados nos pinos corretos no Arduino;
- Verifique se o LED não está conectado invertido;
- Verifique se a posição do botão está correta;
- Verifique se os pinos do botão estão bem inseridos na protoboard;
- Verifique se o código carregou na placa através da IDE Arduino.

6 – Transistor

Módulo 6 – Transistor – Saiba como contornar a limitação das portas do arduino e controle cargas maiores com o transistor.

6.1 – O que é um transistor?

Praticamente todos os dispositivos eletrônicos modernos têm como base o transistor. Talvez você já tenha ouvido falar ao mencionar um processador de um computador ou celular que ele é da tecnologia de x nm (nanômetros). Esse valor se refere à dimensão de parte da construção dos transistores que compõem os processadores.

O transistor pode ser imaginado como uma torneira de eletricidade, quanto mais você abre a torneira, mais água vai passar. Com o transistor é semelhante, dependendo do sinal que você aplicar no pino de controle dele, mais eletricidade ele irá permitir que passe.

Transistores servem muitos propósitos na eletrônica, mas o que vamos aplicar no kit maker é a capacidade de transformar um sinal “fraco”, que não consegue acionar cargas muito fortes, como é o caso das saídas do arduino, que tem uma limitação de corrente, em sinais “fortes”, que conseguem acionar cargas maiores, como um motor, por exemplo.

7 – Motores

Módulo 7 – Motores – Adicione movimento ao leque de capacidades dos projetos com Arduino e aprenda novas formas de interagir seu projeto com o mundo

Luzes e entradas de comando são bastante úteis em projetos, mas muitas vezes queremos ir além e dar movimento aos nossos projetos, para isso, precisamos aprender a usar os motores. Nesse módulo iremos aprender sobre dois dos motores mais comuns para projetos, o motor dc e o servo motor

7.1 – Motor DC

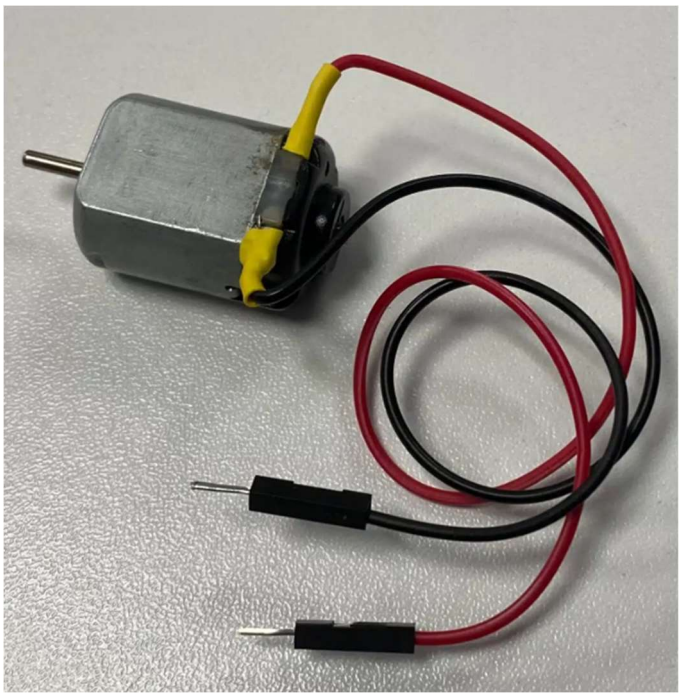
Praticamente qualquer máquina que se mexe tem um motor.

Existem alguns tipos de motores elétricos, mas o mais comum de encontrarmos em projetos que demandam movimentos sem precisão milimétrica são os motores DC. São os motores mais simples de serem acionados e também podem ter sua velocidade controlada facilmente.

Basta aplicar uma tensão no motor que ele irá girar, é assim simples. Dependendo da polaridade que você ligar os fios o motor irá girar no sentido horário ou anti-horário. Claro que o princípio de funcionamento é um pouco mais complicado, mas não é o objetivo deste guia ir à fundo na teoria. Mas caso você queira saber um pouco mais sobre o funcionamento, pode ver este vídeo.

Também é preciso estar atento aos limites de funcionamento, cada motor tem uma tensão máxima de operação, e se for ligado numa tensão maior poderá apresentar problemas muito mais cedo ou até mesmo queimar.

O motor que acompanha o kit maker foi selecionado para trabalhar sem problemas com o Arduino, então pode ficar tranquilo quanto a isso.



7.2 – Acionando um motor DC

Vamos fazer algo parecido com o LED, ligar e desligar o motor em um determinado intervalo.

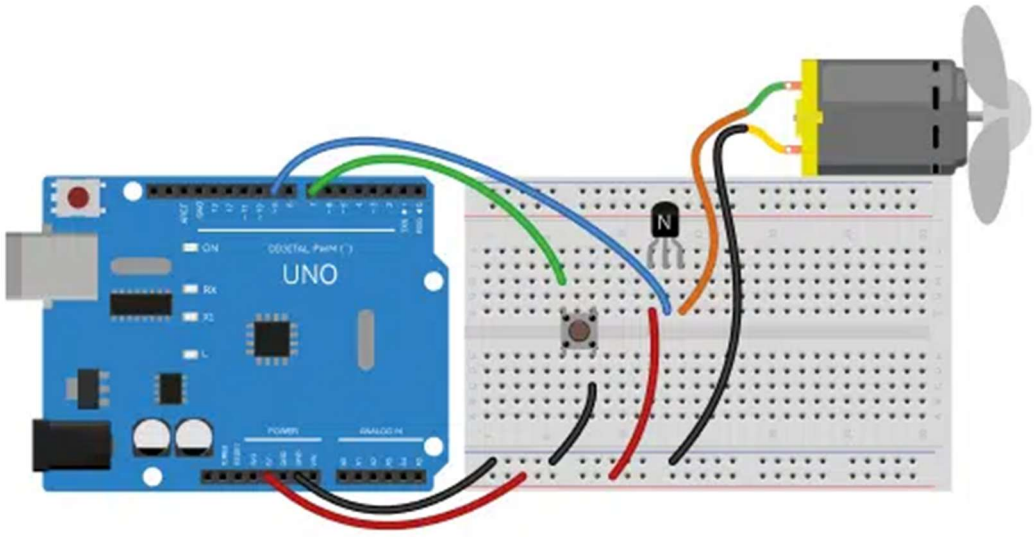
Vamos usar a mesma função digitalWrite que usamos nos LEDs para controlar o motor, o que nos mostra que o arduino não se importa com o que está ligado nas suas entradas (dentro das suas limitações técnicas). Dessa forma não precisamos dizer ao arduino se estamos ligando um motor, um LED ou qualquer outro dispositivo e podemos aproveitar os códigos que já utilizamos, desde que os componentes tenham comportamentos parecidos eletricamente.

Material necessário

1x Motor DC
1x Transistor BC338
1x Chave Táctil Push-button
1x Protoboard 400 pontos
5x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

Nesta montagem usaremos novamente um botão na cavidade central da protoboard. Vamos usar a linha azul como GND ligando tanto o botão quanto o motor, e o outro fio do motor será ligado no emissor do transistor (pino da direita olhando de frente para a parte reta do transistor). O pino do meio do transistor será ligado ao pino 9 do arduino e a perna mais a esquerda será ligada no 5 V.



Programa
Neste programa iremos usar uma nova função, a função else.
Ela serve quando queremos dizer para o programa que se certa condição não for atendida, deve executar uma ou uma série de instruções. Ela trabalha acoplada a função if, desta forma:

```
if(digitalRead(botao) == LOW) {          // Se o
botao estiver apertado (leitura igual a LOW),
executa as instrucoes entre chaves



  digitalWrite(motor,HIGH);              // Aciona o motor DC



  delay(10);                             // Aguarda 10 milissegundos

 } else {                                // Caso contrário (botao nao apertado), executa as instrucoes entre
chaves

  digitalWrite(motor,LOW);               // Desliga o motor DC

 }
```

Podemos ver que o else fica depois de fecharmos as chaves do if, e tem seu próprio conjunto de chaves. Como a condição foi verificada no if, o else não recebe nenhum parâmetro, e nem tem parênteses na sua estrutura. Ela serve para que determinado código seja executado somente se a condição do if não for atendida, e não toda vez após o if

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Acionando um Motor DC
// AUTOR: MAKERHERO


//----------------Declara as variaveis do codigo----------------

int botao = 7;                           // Declara a variável botao e atribui o valor 7 a ela
```

```
int motor = 9;                    // Declara a variável motor e atribui o valor 9 a ela

//----------------Funcao executada uma vez na inicializacao do sistema----------------

void setup() {

  pinMode(botao, INPUT_PULLUP);          // Configura a porta 7 (valor da variável
botao) como saida

  pinMode(motor, OUTPUT);              // Configura a porta 9 (valor da variável motor)
como saida

}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado-----
----------

void loop() {

  if(digitalRead(botao) == LOW) {          // Se o botao estiver apertado (leitura igual a
LOW), executa as instrucoes entre chaves

    digitalWrite(motor,HIGH);            // Aciona o motor DC



    delay(10);                    // Aguarda 10 milissegundos

  } else {                        // Caso contrário (botao nao apertado), executa as
instrucoes entre chaves

    digitalWrite(motor,LOW);             // Desliga o motor DC

  }

  delay(100);                     // Aguarda 100 milissegundos

}
```

Possíveis erros

- Verifique se os jumpers estão na mesma coluna dos terminais dos componentes, fazendo assim a conexão;
- Verifique se os jumpers estão ligados nos pinos corretos no Arduino;
- Verifique se a posição do botão está correta;
- Verifique se os pinos do botão estão bem inseridos na protoboard;
- Verifique se o código carregou na placa através da IDE Arduino.

O motor pode girar para os dois lados. Porém, por ser um ventilador, o ar deve soprar para frente.

Nesse caso, você pode ligar o motor, e caso ele gire ao contrário, basta inverter os terminais.

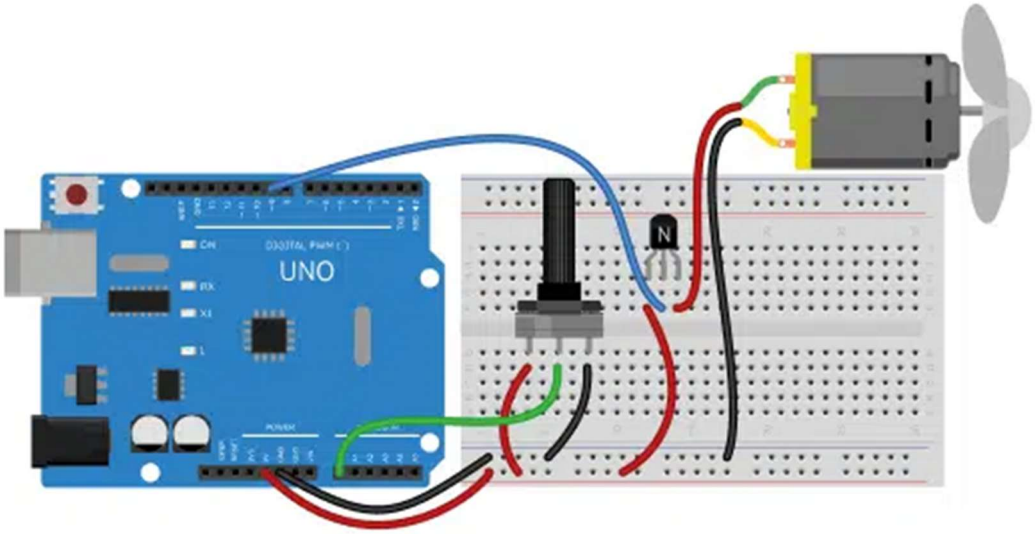
7.3 – Controlando um motor DC

Agora que sabemos ligar um motor DC no arduino, seria muito interessante controlar ele, poder alterar sua velocidade.

Podemos fazer isso da mesma forma que alteramos o brilho do LED

Material necessário
1x Motor DC
1x Transistor BC338
1x Potenciômetro 10K ohm
1x Protoboard 400 pontos
5x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito
Nessa montagem retiraremos o botão e trocaremos por um potenciômetro, que já vimos a conexão no exercício Troque a Cor das Luzes. A montagem do motor será igual ao exercício anterior



Programa

Neste programa iremos usar a função **map**(). Essa função ajuda a “traduzir” valores de uma faixa de valores para outra sem precisarmos fazer todas as contas e escrever o código das contas. Para acionar o motor com velocidade variável vamos usar a função **analogWrite**() que já vimos em outras aulas. Essa função sabemos que recebe valores entre 0 e 255.

Para fazer o controle da velocidade usaremos a função **analogRead**() que também já vimos, e que tem como saída valores entre 0 e 1023.

Aí podemos ver o valor da função **map**(), pois temos dois intervalos distintos e queremos que eles se comuniquem.

Vamos usar a função **map**() dessa forma:

```
valorMotor = map(valorPot, 0, 1023, 0, 255);
 // Mapeia os valores lidos do potenciômetro (entre 0 e 1023) para os valores de angulo do servo (entre 0 e 180)
```

37

Estamos então atribuindo à variável valorMotor o valor referente ao mapeamento da variável valorPot, que pode estar entre 0 e 1023 para o intervalo 0 a 255, valor de entrada da função **analorWrite**().

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Controlando um Motor DC
```

```
// AUTOR: MAKERHERO

//----------------Declara as variaveis do codigo----------------

int motor = 9;                          // Declara a variavel motor e atribui o valor 9 a ela



int pinoPot = A0;                       // Declara a variavel pinoPot e atribui o valor A0 a
ela

int valorPot;                           // Declara a variavel valorPot e não atribui nenhum
valor a ela

int valorMotor;                         // Declara a variavel valorMotor e não atribui
nenhum valor a ela

//----------------Funcao executada uma vez na inicializacao do sistema----------------

void setup() {
  pinMode(pinoPot, INPUT_PULLUP);               // Configura a porta A0 (valor da
variável pinoPot) como entrada

  pinMode(motor, OUTPUT);                       // Configura a porta 9 (valor da variável
motor) como saida

}


//----------------Funcao executada repetidamente enquanto o sistema estiver ligado-----
----------

void loop() {

  valorPot = analogRead(pinoPot);               // Lê o valor do potenciômetro (de 0 a
1023) e atribui o valor lido a variavel valorPot

  valorMotor = map(valorPot, 0, 1023, 0, 255);  // Mapeia os valores lidos do
potenciometro (entre 0 e 1023) para os valores de entrada do analogWrite (entre 0 e
255)
```

```
 analogWrite(motor, valorMotor);                    // Envia o comando para o servo mover
para o angulo informado

 delay(15);                                 // Aguarda 15 milissegundos

}
```

Possíveis erros

- Verifique se os jumpers estão na mesma coluna dos terminais dos componentes, fazendo assim a conexão;
- Verifique se os jumpers estão ligados nos pinos corretos no Arduino;
- Verifique se a posição do botão está correta;
- Verifique se os pinos do botão estão bem inseridos na protoboard;
- Verifique se o código carregou na placa através da IDE Arduino.

7.4 – Servo motor

Existem diversos tipos de motor que podem ser utilizados facilmente com o Arduino. Entre os mais fáceis de se utilizar está o servo motor, sendo bastante utilizado em aeromodelismo e outros projetos que precisam de movimentos mais curtos e com uma precisão razoável.

O servo motor é, basicamente, um motor onde é possível controlar sua posição. A maioria dos servos consegue fazer um movimento de rotação de até meia volta, não gira continuamente como outros motores. Porém, existem outros modelos de servo motor que conseguem dar voltas completas, sendo cada modelo indicado para um tipo de projeto diferente.

O servo tem um motor DC que faz a rotação, e uma lógica de controle embarcada nele, que recebe um sinal, além da alimentação, para determinar a posição em que ele deve ficar.

7.5 – Acionando um servo Motor

Agora vamos acionar o servo motor. Como vimos, a diferença dele para o motor DC está no fato de que conseguimos definir em que posição o servo vai ficar. Nesta aula vamos movimentar um servo ao pressionar um botão. Para isso, vamos precisar de uma entrada digital para o botão e uma saída analógica para o servo motor. Conforme variamos a saída analógica do servo, variamos a sua posição.

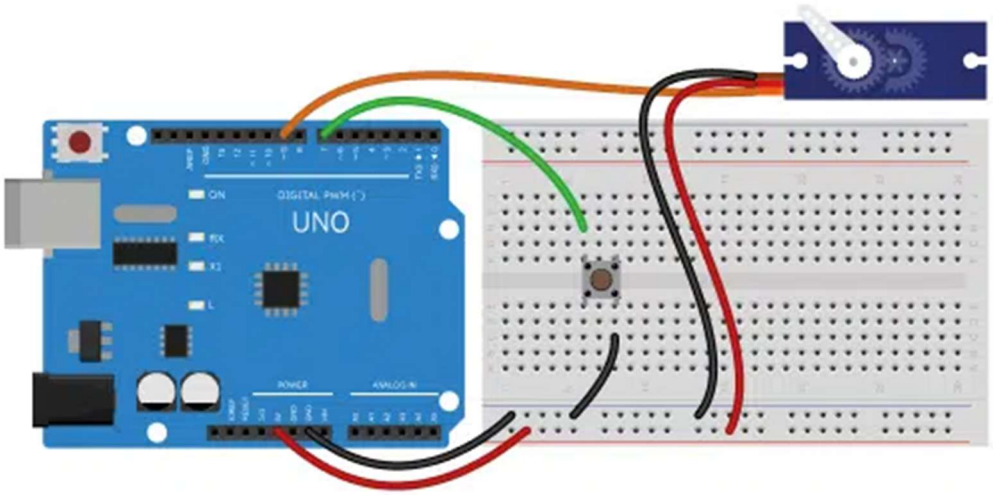
Material necessário

1x Servo 9g
1x Chave push-button
7x Jumper Macho-macho
1x Protoboard 400 pontos
1x Cabo USB
1x Placa Uno

Montagem do circuito

O servo possui três fios para sua utilização e na extremidade desses fios um conector de 3 pinos fêmea. O fio marrom do servo deve ser conectado ao GND, o fio vermelho ao 5 V e o fio laranja ao pino 9 do Arduino. **Na ilustração abaixo o fio preto corresponde ao marrom, o fio vermelho ao vermelho e o fio amarelo ao fio laranja.**

Para conectar o servo basta utilizar 3 jumpers macho-macho e ligar o conector do servo com a protoboard. Também acompanha algumas hastes e parafusos. **Não é necessário parafusar as hastes ao servo,** apenas um encaixe já basta. Você pode escolher uma das hastes para visualizar melhor o movimento do servo.



Programa

No exercício 8 o programa deve fazer o seguinte: ao apertar o botão, o servo gira até meia volta (180 graus) e retorna, ficando parado até pressionar o botão novamente.

Para esse exercício vamos utilizar uma ferramenta muito poderosa do Arduino e uma das principais razões para que ele tenha facilitado tanto a eletrônica, as bibliotecas.

Bibliotecas são códigos escritos pela própria Arduino ou por outras pessoas para realizar as mais diversas tarefas, comunicar com sensores, realizar funções mais complexas ou, no caso desse exemplo, fazer a comunicação com o servo motor.

No começo do código fazemos a inclusão da biblioteca que queremos utilizar em nosso código da seguinte forma:

```
7  #include <Servo.h>
```

#include diz que queremos incluir uma biblioteca e entre **<>** colocamos o nome da biblioteca que desejamos incluir. Os arquivos de biblioteca sempre terão a extensão **.h**

Importante lembrar que para incluirmos uma biblioteca, ela deve estar instalada na pasta libraries que fica dentro da pasta Arduino criada em Meus Documentos ao instalar a IDE, ou dentro da pasta do exercício, junto com o arquivo **.ino**

A maioria das bibliotecas funcionam declarando uma variável do tipo da biblioteca, como fazemos aqui:

```
11  Servo meuServo;
```

Criamos a variável meuServo do tipo Servo, e com essa variável iremos interagir com as funções da biblioteca.

```
27  meuServo.attach(9);
```

Aqui dizemos que estamos conectando um servo ao pino 9 do Arduino e a biblioteca dá início ao sinal de controle do servo

```
33  meuServo.write(angulo);
```

Aqui escrevemos o ângulo para o qual queremos que o servo se mova

```
49  meuServo.detach();
```

Por fim, desligamos o sinal de controle do servo.

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Acionando um servo Motor

// AUTOR: MAKERHERO

//----------------Inclui as bibliotecas do codigo----------------

#include <Servo.h>                    // Inclui a biblioteca Servo para facilitar o controle do servo motor

//----------------Declara as variaveis do codigo----------------
```

```
Servo meuServo;                         // Declara a variavel meuServo para utilizar as funcoes
de biblioteca Servo

int botao = 7;                          // Declara a variavel botao e atribui o valor 7 a ela

//----------------Funcao executada uma vez na inicializacao do sistema----------------

void setup() {


 pinMode(botao, INPUT_PULLUP);                  // Configura a porta 7 (valor da variável botao)
como saida

}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------
--

void loop() {

 meuServo.attach(9);                    // Inicializa o envio de comandos para o servo

 if(digitalRead(botao) == LOW) {                // Se o botao estiver apertado (leitura igual a
LOW), executa as instrucoes entre chaves

  for(int angulo=0; angulo<=180; angulo++) {        // Aumenta o angulo do servo ate chegar
em 180 graus

   meuServo.write(angulo);                  // Envia o comando para o servo mover para o
angulo informado

   delay(10);                       // Aguarda 10 milissegundos


  }

  for(int angulo=180; angulo>=0; angulo--) {        // Diminui o angulo do servo ate retornar a 0

   meuServo.write(angulo);                  // Envia o comando para o servo mover para o
angulo informado

   delay(10);                       // Aguarda 10 milissegundos


  }


 }
```

```
  meuServo.detach();                    // Encerra o envio de comandos para o servo
}
```

Possíveis erros

Caso o exercício não funcione, verifique alguns dos possíveis erros:

- Verifique se os jumpers estão na mesma coluna dos terminais dos componentes, fazendo assim a conexão;
- Verifique se os jumpers estão ligados nos pinos corretos no Arduino;
- Verifique se as conexões do servo estão corretas, fio marrom no GND, fio vermelho no 5 V e laranja no pino 9;
- Verifique se os pinos do botão estão bem inseridos na protoboard;
- Verifique se o código carregou na placa através da IDE Arduino.

7.6 – Controlando um servo Motor

Já vimos no exercício Troque a cor das Luzes como funciona um potenciômetro e no exemplo anterior como funciona um servo motor. Agora, veremos como utilizar esses dois componentes em conjunto controlando o servo através da rotação do potenciômetro.

É interessante observar como os dois componentes utilizam portas analógicas. Com isso podemos ler diversos valores no potenciômetro e mandar esses valores diversos para o motor. Não é apenas um comando de ligado e desligado, como no exercício Interruptor de luz Liga-Desliga.

Material necessário

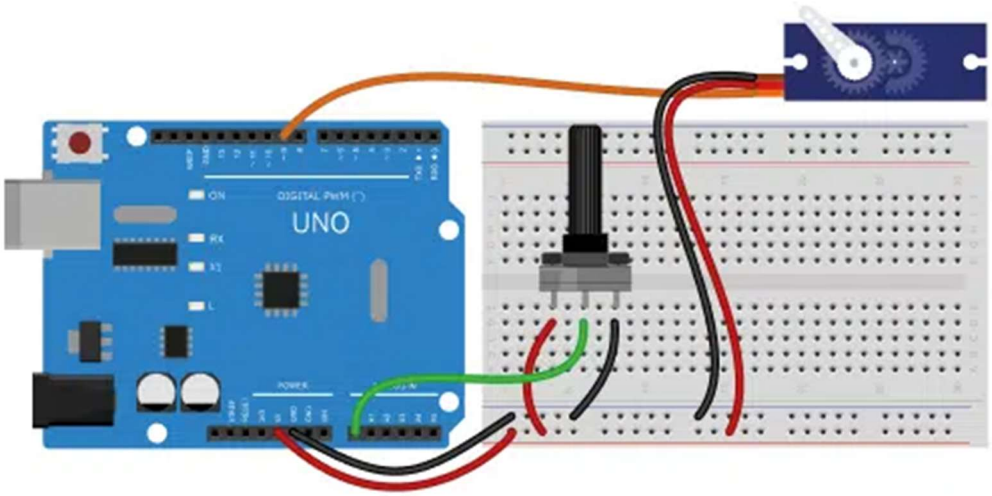
1x Servo 9g
1x Potenciômetro 10K ohm
8x Jumper macho-macho
1x Protoboard 400 pontos
1x Cabo USB
1x Placa Arduino Uno

Montagem do circuito

Como já visto no exercício anterior, o servo possui três fios para sua utilização e na extremidade desses fios um conector de 3 pinos fêmea. O fio marrom do servo deve ser conectado ao GND, o fio vermelho ao 5 V e o fio laranja ao pino 9 do Arduino.

Para conectar o servo basta utilizar 3 jumpers macho-macho. Também acompanha algumas hastes e parafusos. Não é necessário parafusar as hastes ao servo, apenas um encaixe já basta.

Adicione também na montagem o potenciômetro, como visto no exercício de controle do motor DC.



Programa

O programa consiste em ler o valor de rotação do potenciômetro e controlar a rotação do servo de acordo. Neste código não temos nenhuma novidade quanto às funções, então todos os comandos devem ser familiares à você.

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Controlando um Servo Motor

// AUTOR: MAKERHERO



//----------------Inclui as bibliotecas do codigo----------------
```

```
#include <Servo.h>                    // Inclui a biblioteca Servo para facilitar o controle do servo motor

//----------------Declara as variaveis do codigo----------------

Servo meuServo;                      // Declara a variavel meuServo para utilizar as funcoes de biblioteca Servo

int pinoPot = A0;                    // Declara a variavel pinoPot e atribui o valor A0 a ela

int valorPot;                        // Declara a variavel valorPot e não atribui nenhum valor a ela

int anguloServo;                     // Declara a variavel anguloServo e não atribui nenhum valor a ela


//----------------Funcao executada uma vez na inicializacao do sistema----------------

void setup() {

  meuServo.attach(9);                // Inicializa o envio de comandos para o servo

}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------


void loop() {

  valorPot = analogRead(pinoPot);    // Lê o valor do potenciômetro (de 0 a 1023) e atribui o valor lido a variavel valorPot

  anguloServo = map(valorPot, 0, 1023, 0, 180);   // Mapeia os valores lidos do potenciometro (entre 0 e 1023) para os valores de angulo do servo (entre 0 e 180)

  meuServo.write(anguloServo);       // Envia o comando para o servo mover para o angulo informado

  delay(15);                         // Aguarda 15 milissegundos


}
```

Possíveis erros

Caso o exercício não funcione, verifique alguns dos possíveis erros:

- Verifique se os jumpers estão na mesma coluna dos terminais dos componentes, fazendo assim a conexão;
- Verifique se os jumpers estão ligados nos pinos corretos no Arduino;
- Verifique se as conexões do servo estão corretas, fio marrom no GND, fio vermelho no 5V e laranja no pino 9;
- Verifique se os pinos do potenciômetro estão bem inseridos na protoboard;
- Verifique se o código carregou na placa através da IDE Arduino.

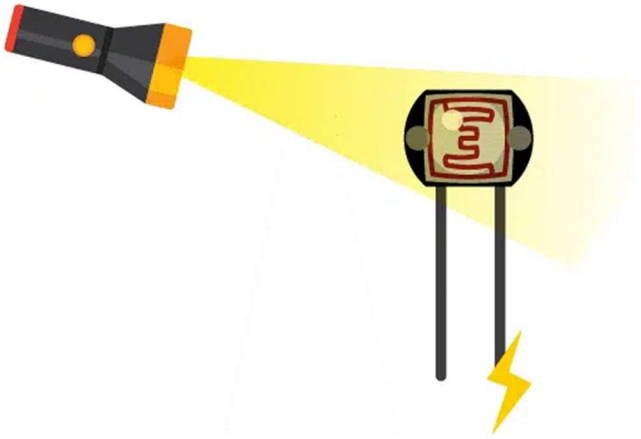
8 – LDR

Módulo 8 – LDR – Você descobrirá uma das muitas formas de “medir” o ambiente ao redor do projeto e interagir com base nas condições que você medir.

8.1 – O que é o LDR?

O LDR é bastante usado nos postes de luz na cidade, fazendo que quando anoitece as luzes da cidade acendam. Ele também é usado em lâmpadas de jardim que acendem ao anoitecer. Veja neste projeto como utilizar o sensor de luz LDR com Arduino.

Assim como um potenciômetro varia sua resistência conforme a rotação, o LDR é um resistor que varia sua resistência conforme a intensidade de luz no ambiente. Com isso conseguimos medir a quantidade de luz presente em um ambiente.



Da mesma forma que o potenciômetro é ligado em uma entrada analógica do Arduino, também é possível utilizar uma porta analógica para ler o valor de um LDR. Para o LED, iremos trabalhar apenas com uma porta digital.

8.2 – Sensor de Luz Ambiente

Neste exercício iremos mostrar que é possível identificar a quantidade de luz presente em um ambiente utilizando o Arduino e o sensor de luminosidade LDR (*Light Dependent Resistor*).

O exercício consiste em ligar ou desligar o LED de acordo com a intensidade de luz presente no ambiente. Pouca luz acende o LED, bastante luz apaga o LED. Você pode variar a luz do ambiente acendendo ou apagando as luzes ou colocando a mão em cima do sensor LDR como mostrado abaixo.

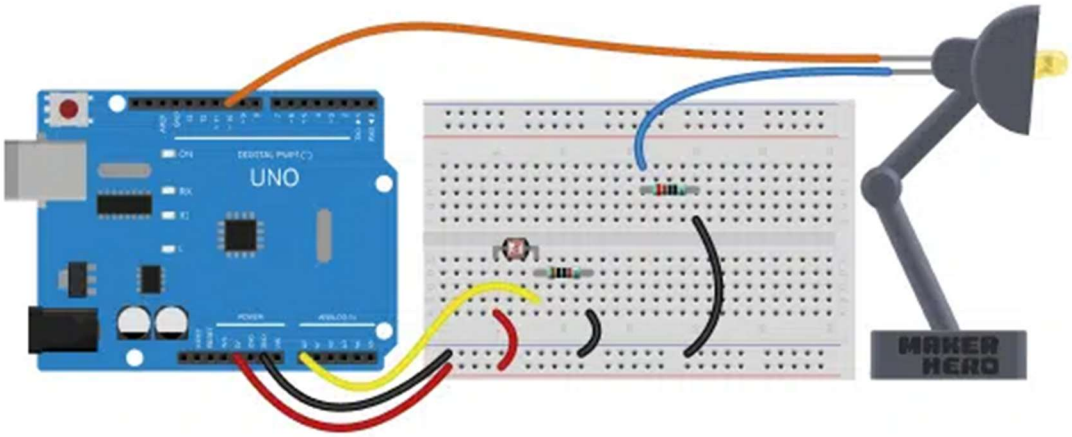
Material necessário

1x LED Amarelo 5 mm
1x Luminária 3D
1x Resistor 220 ohm
1x Resistor 10K ohm
1x Sensor de luminosidade LDR
1x Protoboard 400 pontos
7x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

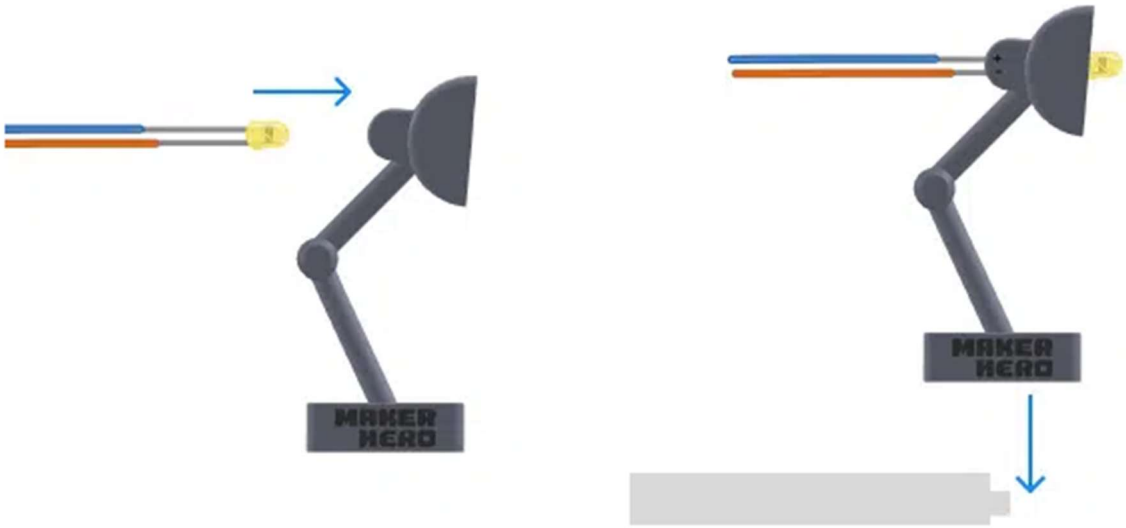
Na montagem deste exercício, atente-se à montagem do LED, pois, como já mostramos nos exercícios anteriores, ele tem lado positivo e negativo. Já o sensor de luminosidade LDR não

tem lado, ou seja apenas insira os seus pinos na protoboard e faça a ligação conforme indicado abaixo. Note que o LDR usa um resistor de 10k ohm e o LED um resistor de 220 ohm.



Montagem da luminária

Para encaixar o LED na peça de impressão 3D da luminária siga as seguintes instruções. Conecte os jumpers no LED e depois encaixe-o na luminária no sentido das costas da lumiária para a frente. Para encaixar a luminária na protoboard, encaixe os rebaixos da peça nos pinos de encaixe da protoboard.



Programa

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Sensor de luz ambiente

// AUTOR: MAKERHERO

//----------------Declara as variaveis do codigo----------------

int pinoLed = 10;                    // Declara a variavel pinoLed e atribui o valor 10 a ela

int pinoSensorLuz = A0;              // Declara a variavel pinoSensorLuz e atribui o valor A0 a ela

int valorLuz = 0;                    // Declara a variavel valorLuz e atribui o valor 0 a ela

//----------------Funcao executada uma vez na inicializacao do sistema----------------

void setup() {

 pinMode(pinoLed,OUTPUT);            // Configura a porta 10 (valor da variável pinoLed) como saida

}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------

void loop() {

 valorLuz = analogRead(pinoSensorLuz);     // Lê o valor do sensor de luminosidade (de 0 a 1023) e atribui o valor lido a variavel valorLuz


 if(valorLuz<750) {                  // Se o valor da variavel valorLuz for menor que 750, executa as instrucoes entre chaves

  digitalWrite(pinoLed,HIGH);        // Acende o LED

 } else {                            // Caso contrario (valorLuz maior que 750), executa as instrucoes entre chaves

  digitalWrite(pinoLed,LOW);         // Apaga o LED

 }

 delay(10);                          // Aguarda 10 milissegundos

}
```

Possíveis erros

Caso o exercício não funcione, verifique alguns dos possíveis erros:

- Verifique se os jumpers estão na mesma coluna dos terminais dos componentes, fazendo assim a conexão;
- Verifique se os jumpers estão ligados nos pinos corretos no Arduino;
- Verifique se o valor de resistores está correto como indicado na montagem do exercício;
- Verifique se o código carregou na placa através da IDE Arduino.

9 – Buzzer

Módulo 9 – Buzzer – Aqui você será apresentado ao buzzer, uma forma de alertar ou sinalizar com som o usuário dos seus projetos.

9.1 – Buzzer

Sabe quando a geladeira fica aberta e começa a apitar? Ou algum outro dispositivo que emite um “beep” ao interagir com ele. Muito provavelmente esse som é emitido por um buzzer.

O buzzer é um pequeno alto falante capaz de gerar tons em frequências determinadas, sendo possível, então, tocar pequenas músicas ou alarmes. Ele é bastante encontrado em brinquedos ou relógios digitais com alarme. No Arduino podemos utilizar a função chamada **tone();** que leva dois parâmetros: o pino em que o buzzer está conectado e a frequência do tom.

9.2 – Acionando um Buzzer

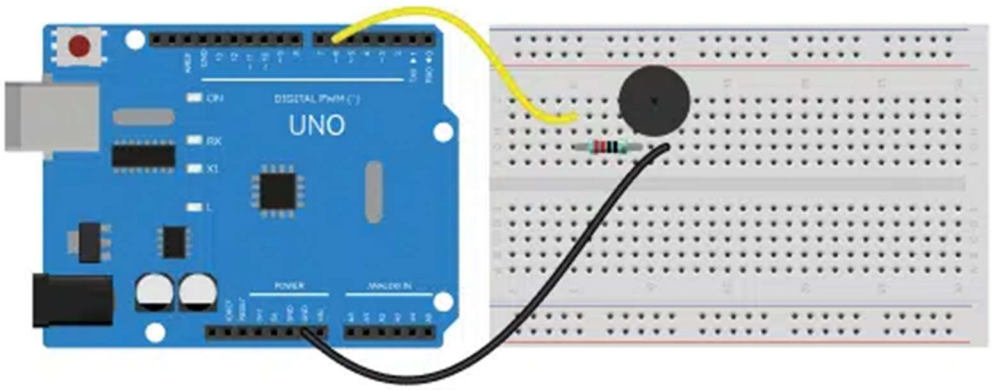
Mas e se precisarmos chamar ainda mais a atenção para o nosso projeto? Uma luz você precisa estar olhando para ela para que ela sirva de alerta. Agora um som será ouvido de qualquer forma. Então vamos aprender como acionar um buzzer para ampliar as possibilidades de nossos projetos.

Material necessário

1x Buzzer
1x Resistor 220 ohm
1x Protoboard 400 pontos
2x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

Atente-se que o buzzer tem lado positivo e negativo, assim como os LEDs. Ele está indicado por uma bolinha com mais no topo do buzzer.



Programa

Para controlar o buzzer iremos usar as funções **tone**() e **noTone**() para ligar e desligar o buzzer respectivamente. A função **tone**() recebe como parâmetro o pino que iremos controlar e a frequência do tom que se deseja emitir.

```
21 | tone(pinoBuzzer, 440);          // Aciona o Buzzer
```

Para o exemplo iremos usar um Lá (440 Hz). Devido a limites do microcontrolador, a frequência mais baixa que pode ser produzida é 31 Hz. Já a mais alta é 4 MHz, muito acima da frequência da audição humana, que em média é até 20 kHz (Mais uma vez os multiplicadores, lembra deles quando falamos de resistores?)

Caso não digamos para o arduino parar de enviar o tom, o arduino continuará tocando indefinidamente. Para parar usamos a função **noTone**(), que recebe como parâmetro apenas o pino que desejamos desativar o tom.

```
27  noTone(pinoBuzzer);                 // Desliga o Buzzer
```

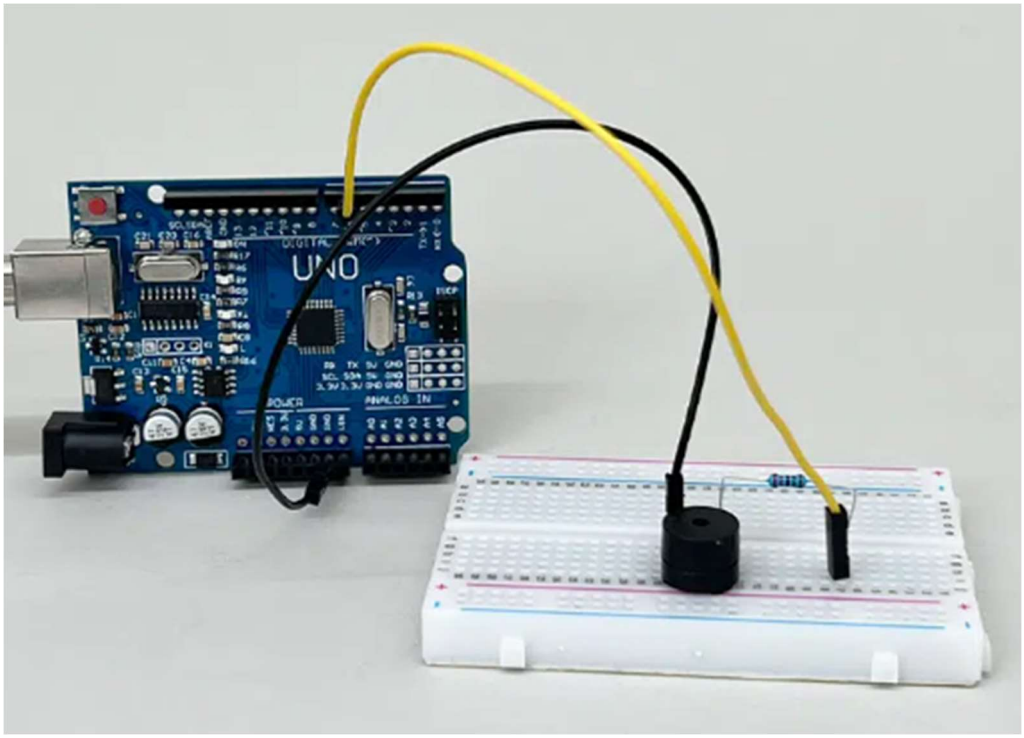
Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Acionando um buzzer

// AUTOR: MAKERHERO

//----------------Declara as variaveis do codigo----------------

int pinoBuzzer = 6;                 // Declara a variavel pinoBuzzer e atribui o valor 6 a ela

//----------------Funcao executada uma vez na inicializacao do sistema----------------

void setup() {

 pinMode(pinoBuzzer, OUTPUT);          // Configura a porta 6 (valor da variável pinoBuzzer) como saida

}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------

void loop() {

 tone(pinoBuzzer, 440);              // Aciona o Buzzer

 delay(30);                      // Aguarda 30 milissegundos

 noTone(pinoBuzzer);                 // Desliga o Buzzer

 delay(200);                     // Aguarda 200 milissegundos


}
```

9.3 – Dó Ré Mi

Bora fazer música com o Arduino? Agora que sabemos acionar um buzzer, conseguimos fazer ele tocar em diferentes frequências e fazer música com ele. Jogos como os de super nintendo tem o mesmo número de bits do timer que o arduino usa para a função tone, então teremos um som parecido usando o arduino.

Na música, cada nota musical possui uma frequência específica dada em hertz (Hz). Olhando as teclas de um piano é possível ver a frequência de cada uma das notas de uma escala de Dó por exemplo. Neste projeto iremos mostrar como é possível reproduzir notas musicais utilizando um buzzer com Arduino.

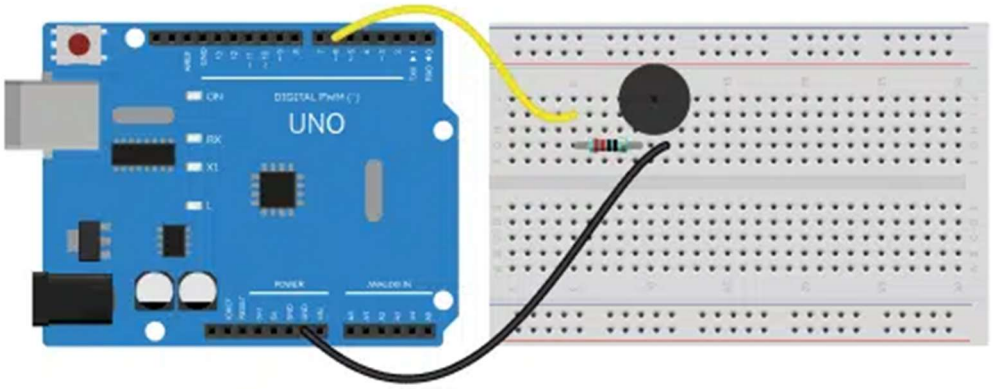


Material necessário

1x Buzzer
2x Jumper Macho-macho
1x Protoboard 400 pontos
1x Cabo USB
1x Placa Uno

Montagem do circuito

Utilizaremos a mesma montagem do exercício anterior.



Programa

Para este exercício vamos utilizar a estrutura das definições (**#define**) de constantes e a estrutura **array**.

A estrutura **#define**, como já falamos quando explicamos a estrutura básica de um programa arduino, serve para quando não iremos alterar um determinado valor e iremos usá-lo como referência.

```
//----------------Declara as constantes do codigo----------------

#define DO  262                                         // Define DO como o valor 262

#define RE  294                                         // Define RE como o valor 294

#define MI  330                                         // Define MI como o valor 330

#define FA  349                                         // Define FA como o valor 349

#define SOL  392                                        // Define SOL como o valor 392

#define LA  440                                         // Define LA como o valor 440

#define SI  494                                         // Define SI como o valor 494

#define DO_2  523                                       // Define DO_2 como o valor 523
```

A estrutura array é um tipo de lista de variáveis, quando temos vários valores relacionados. Ao invés de usar várias variáveis com nomes muito parecidos, usamos essa estrutura.

```
int melodia[] = {

  DO, RE, MI, FA, SOL, LA, SI, DO_2

};
```

A estrutura array pode ser percebida principalmente quando depois do nome da variável abrimos e fechamos colchetes. A declaração acima já coloca todos os valores no array, então não precisamos dizer seu tamanho. Caso só tivéssemos os valores a colocar depois, precisaríamos colocar um número, que é o número de elementos que queremos armazenar, dentro dos colchetes.

Com a estrutura de array conseguimos alterar facilmente entre elementos, usando laços como o for() por exemplo, que é o que fazemos no código deste exercício.

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Do-Re-Mi
// AUTOR: MAKERHERO
//----------------Declara as constantes do codigo----------------
#define DO  262                          // Define DO como o valor 262
#define RE  294                          // Define RE como o valor 294
#define MI  330                          // Define MI como o valor 330
#define FA  349                          // Define FA como o valor 349
#define SOL  392                         // Define SOL como o valor 392
#define LA  440                          // Define LA como o valor 440
#define SI  494                          // Define SI como o valor 494
#define DO_2  523                        // Define DO_2 como o valor 523

//----------------Declara as variáveis do codigo----------------
int pinoBuzzer = 6;                 // Declara a variavel pinoBuzzer e atribui o valor 6 a ela
//----------------Declara um array (tipo de lista) com os elementos entre chaves, na ordem em que são colocados----------------

int melodia[] = {
```

```
  DO, RE, MI, FA, SOL, LA, SI, DO_2
};
//----------------Funcao executada uma vez na inicializacao do sistema----------------
void setup() {
  pinMode(pinoBuzzer,OUTPUT);           // Configura a porta 6 (valor da variável pinoBuzzer) como saida
}
//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------
void loop() {
  for(int i=0; i<8; i++) {              // Repete os comandos dentro das chaves 8 vezes
    tone(pinoBuzzer, melodia[i]);       // Aciona o buzzer com a frequecia melodia da posicao i do array
    delay(500);                         // Aguarda 500 milissegundos
  }
}
```

9.4 – Projeto: Alarme com Sensor a Laser

Para este projeto, vamos combinar os exercícios em que vimos o sensor de luminosidade e como acionar um buzzer para criar um detector de presença a laser, igual àqueles que vemos em filmes de espionagem, mas com um laser somente. Claro que aprendendo o funcionamento deste projeto você pode juntar mais lasers e fazer um sensor mais parecido com os encontrados em filmes.

Neste projeto iremos fazer um protótipo de alarme por laser. O laser deve ser apontado para o LDR, para que o sensor detecte bastante luz, e quando algo cortar o feixe de luz, o buzzer irá soar como um alarme. Veja como o projeto irá funcionar:

Material necessário

1x Diodo Laser
1x Resistor 10K ohm
1x LDR
1x Buzzer
1x Protoboard 400 pontos
9x Jumper Macho-macho

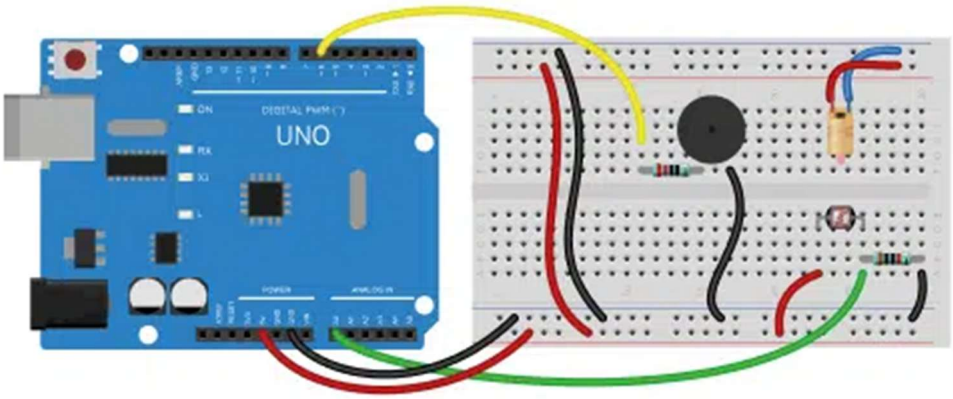
1x Cabo USB
1x Placa Uno

Montagem do circuito

Na montagem abaixo, note que o laser está com os fios conectados diretamente na protoboard. Para o funcionamento correto precisamos apontar o laser para o sensor LDR. Caso o laser não acenda, verifique se os fios foram colocados até o fundo dos furos da protoboard.

Atenção! Não aponte o laser diretamente no seu olho ou de outra pessoa.

Veja como deve ser feita a montagem do laser.



Programa

No programa abaixo, quando o laser não estiver apontado para o LDR, o buzzer irá soar. Sendo assim, aponte o laser para o LDR e corte o
feixe de luz com um objeto para que o alarme acione.

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Alarme com sensor a laser
```

```
// AUTOR: MAKERHERO

//----------------Declara as variaveis do codigo----------------

int pinoSensorLuz = A0;                    // Declara a variavel pinoSensorLuz e atribui o valor A0 a
ela

int pinoBuzzer = 6;                        // Declara a variavel pinoBuzzer e atribui o valor 6 a ela

int valorLuz = 0;                          // Declara a variavel valorLuz e atribui o valor 0 a ela


//----------------Funcao executada uma vez na inicializacao do sistema----------------

void setup() {

  pinMode(pinoBuzzer,OUTPUT);              // Configura a porta 6 (valor da variável pinoBuzzer)
como saida

}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------

void loop() {

 valorLuz = analogRead(pinoSensorLuz);     // Lê o valor do sensor de luminosidade (de 0 a
1023) e atribui o valor lido a variavel valorLuz


 if(valorLuz<950) {                        // Se o valor da variavel valorLuz for menor que 750, executa
as instrucoes entre chaves

  tone(pinoBuzzer, 440);                   // Aciona o Buzzer na frequência 440 Hz

 } else {                                  // Caso contrário (valorLuz maior que 950) , executa as instrucoes
entre chaves

  noTone(pinoBuzzer);                      // Desliga o Buzzer

 }

 delay(10);                                // Aguarda 10 milissegundos


}
```

10 – Display de 7 segmentos

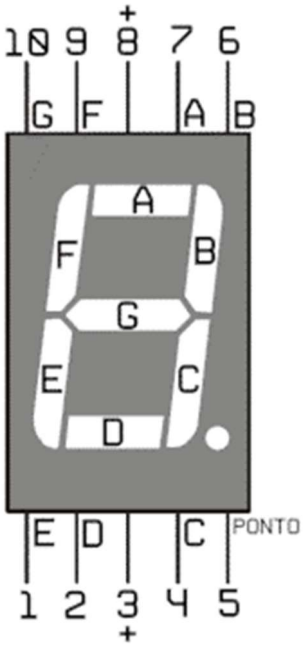
Módulo 10 – Display de 7 Segmentos – Aprenda a controlar um dos displays mais básicos da eletrônica, que está presente desde microondas até esteiras na academia

10.1 – O que é um display de 7 segmentos

Com certeza você já viu esse formato de números em algum lugar. Ele é feito com um componente chamado display de 7 segmentos.

O display de 7 segmentos é composto por alguns LEDs distribuídos em segmentos, para formação de caracteres e números. Todos os LEDs estão inseridos dentro de um mesmo componente, onde cada pino corresponde a um terminal de um LED e há um terminal comum correspondente ao positivo de todos os LEDs. Esse tipo de display pode ser encontrado em relógios digitais, rádios-relógio e etc.

Note que temos 10 pinos, sendo um LED para cada segmento e 2 comuns a todos os LEDs, sendo um de cada lado. Por mais que tenhamos 2 pinos comuns, precisamos ligar apenas 1 para que o display funcione.

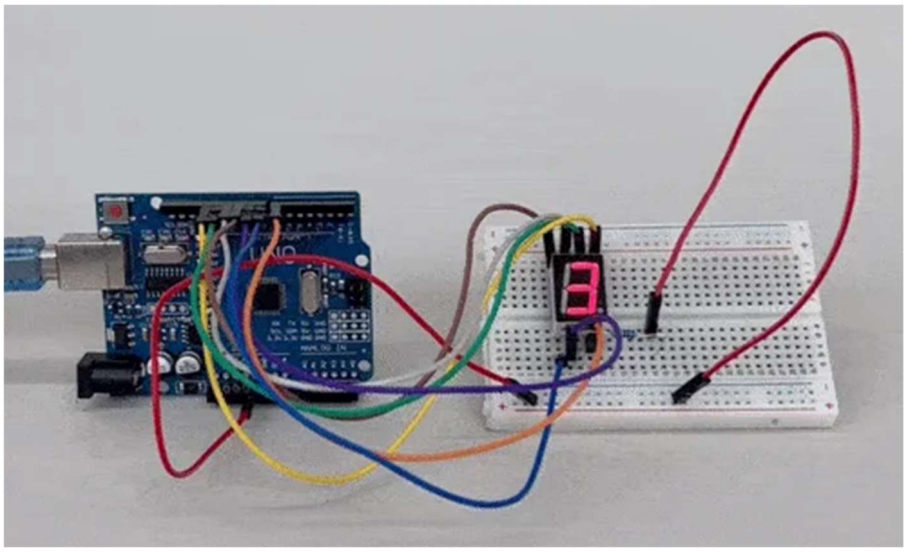


Para cada número se acende um conjunto de LEDs de forma a montar o dígito que se deseja. Por exemplo, para fazer o número 1, acendemos os LEDs B e C. Já para o número 8, acendemos todos.

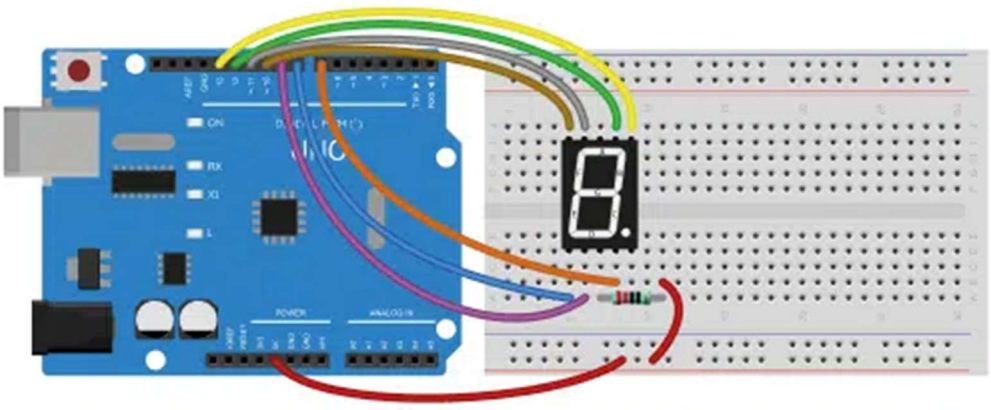
10.2 – Contador Digital

O contador digital é um display de um dígito que vai aumentando conforme o tempo. Como ele é um contador de um dígito apenas, começa em 0 e vai até 9. O exercício parece um pouco simples, mas ensina o funcionamento de um display bastante usado em projetos de eletrônica.

Ligando cada um dos terminais dos LEDs a uma porta do Arduino conseguimos controlá-los individualmente formando caracteres e números. Neste exercício iremos construir um contador digital com o display de 7 segmentos mostrando os números de 0 a 9 com um intervalo de 1 segundo.



Material necessário
1x Display 7 segmentos
1x Resistor 220 ohm
1x Protoboard 400 pontos
10x Jumper Macho-macho
1x Cabo USB
1x Placa Uno
Montagem do circuito
Como se fossem vários LEDs, precisamos colocar um pino do Arduino para cada pino do display. Como o lado positivo dos LEDs é comum a todos, precisamos de apenas 1 resistor.



Programa
O programa, apesar de comprido, é bastante simples. Apenas acendemos e apagamos os LEDs, conforme o número que desejamos.

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Contador digital

// AUTOR: MAKERHERO

//----------------Declara as variaveis do codigo----------------

int segE = 8;                  // Declara a variavel segE e atribui o valor 8 a ela
int segD = 9;                  // Declara a variavel segD e atribui o valor 9 a ela
int segC = 7;                  // Declara a variavel segC e atribui o valor 7 a ela
int segB = 13;                 // Declara a variavel segB e atribui o valor 13 a ela
int segA = 12;                 // Declara a variavel segA e atribui o valor 12 a ela
int segF = 11;                 // Declara a variavel segF e atribui o valor 11 a ela
int segG = 10;                 // Declara a variavel segG e atribui o valor 10 a ela

//----------------Funcao executada uma vez na inicializacao do sistema----------------
void setup() {
  pinMode(segE, OUTPUT);           // Configura a porta 7 (valor da variável segE) como saida
  pinMode(segD, OUTPUT);           // Configura a porta 8 (valor da variável segD) como saida
  pinMode(segC, OUTPUT);           // Configura a porta 9 (valor da variável segC) como saida
  pinMode(segB, OUTPUT);           // Configura a porta 13 (valor da variável segB) como saida
  pinMode(segA, OUTPUT);           // Configura a porta 12 (valor da variável segA) como saida
  pinMode(segF, OUTPUT);           // Configura a porta 11 (valor da variável segF) como saida
  pinMode(segG, OUTPUT);           // Configura a porta 10 (valor da variável segG) como saida

}
//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------

void loop() {

  acende0();                  // Chama a função que acende o digito 0 no display
  delay(1000);                // Aguarda 1000 milissegundos (1 segundo)
  acende1();                  // Chama a função que acende o digito 1 no display
  delay(1000);                // Aguarda 1000 milissegundos (1 segundo)
  acende2();                  // Chama a função que acende o digito 2 no display
```

```
  delay(1000);                  // Aguarda 1000 milissegundos (1 segundo)
  acende3();                   // Chama a função que acende o digito 3 no display
  delay(1000);                  // Aguarda 1000 milissegundos (1 segundo)
  acende4();                   // Chama a função que acende o digito 4 no display
  delay(1000);                  // Aguarda 1000 milissegundos (1 segundo)
  acende5();                   // Chama a função que acende o digito 5 no display
  delay(1000);                  // Aguarda 1000 milissegundos (1 segundo)
  acende6();                   // Chama a função que acende o digito 6 no display
  delay(1000);                  // Aguarda 1000 milissegundos (1 segundo)
  acende7();                   // Chama a função que acende o digito 7 no display
  delay(1000);                  // Aguarda 1000 milissegundos (1 segundo)
  acende8();                   // Chama a função que acende o digito 8 no display
  delay(1000);                  // Aguarda 1000 milissegundos (1 segundo)
  acende9();                   // Chama a função que acende o digito 9 no display
  delay(1000);                  // Aguarda 1000 milissegundos (1 segundo)
}

//----------------Declaracao da funcao que mostra o numero 0 no display de 7 segmentos-------

void acende0() {
  digitalWrite(segE, LOW);            // Acende o segmento E do display
  digitalWrite(segD, LOW);            // Acende o segmento D do display
  digitalWrite(segC, LOW);            // Acende o segmento C do display
  digitalWrite(segB, LOW);            // Acende o segmento B do display
  digitalWrite(segA, LOW);            // Acende o segmento A do display
  digitalWrite(segF, LOW);           // Acende o segmento F do display
  digitalWrite(segG, HIGH);           // Apaga o segmento G do display
}

//----------------Declaracao da funcao que mostra o numero 1 no display de 7 segmentos-----------
void acende1() {
  digitalWrite(segE, HIGH);           // Apaga o segmento E do display
  digitalWrite(segD, HIGH);           // Apaga o segmento D do display
  digitalWrite(segC, LOW);            // Acende o segmento C do display
  digitalWrite(segB, LOW);            // Acende o segmento B do display
  digitalWrite(segA, HIGH);           // Apaga o segmento A do display
  digitalWrite(segF, HIGH);           // Apaga o segmento F do display
  digitalWrite(segG, HIGH);           // Apaga o segmento G do display

}
```

```
//----------------Declaracao da funcao que mostra o numero 2 no display de 7 segmentos-----------

void acende2() {
 digitalWrite(segE, LOW);            // Acende o segmento E do display
 digitalWrite(segD, LOW);            // Acende o segmento D do display
digitalWrite(segC, HIGH);           // Apaga o segmento C do display
digitalWrite(segB, LOW);            // Acende o segmento B do display
 digitalWrite(segA, LOW);            // Acende o segmento A do display
digitalWrite(segF, HIGH);           // Apaga o segmento F do display
digitalWrite(segG, LOW);            // Acende o segmento G do display
}

//----------------Declaracao da funcao que mostra o numero 3 no display de 7 segmentos-----------

void acende3() {
 digitalWrite(segE, HIGH);           // Apaga o segmento E do display
 digitalWrite(segD, LOW);            // Acende o segmento D do display
 digitalWrite(segC, LOW);            // Acende o segmento C do display
 digitalWrite(segB, LOW);            // Acende o segmento B do display
 digitalWrite(segA, LOW);            // Acende o segmento A do display
 digitalWrite(segF, HIGH);           // Apaga o segmento F do display
 digitalWrite(segG, LOW);            // Acende o segmento G do display
}

//----------------Declaracao da funcao que mostra o numero 4 no display de 7 segmentos-----------

void acende4() {
 digitalWrite(segE, HIGH);           // Apaga o segmento E do display
 digitalWrite(segD, HIGH);           // Apaga o segmento D do display
 digitalWrite(segC, LOW);            // Acende o segmento C do display
 digitalWrite(segB, LOW);            // Acende o segmento B do display
 digitalWrite(segA, HIGH);           // Apaga o segmento A do display
 digitalWrite(segF, LOW);            // Acende o segmento F do display
 digitalWrite(segG, LOW);            // Acende o segmento G do display
}

//----------------Declaracao da funcao que mostra o numero 5 no display de 7 segmentos-----------

void acende5() {
```

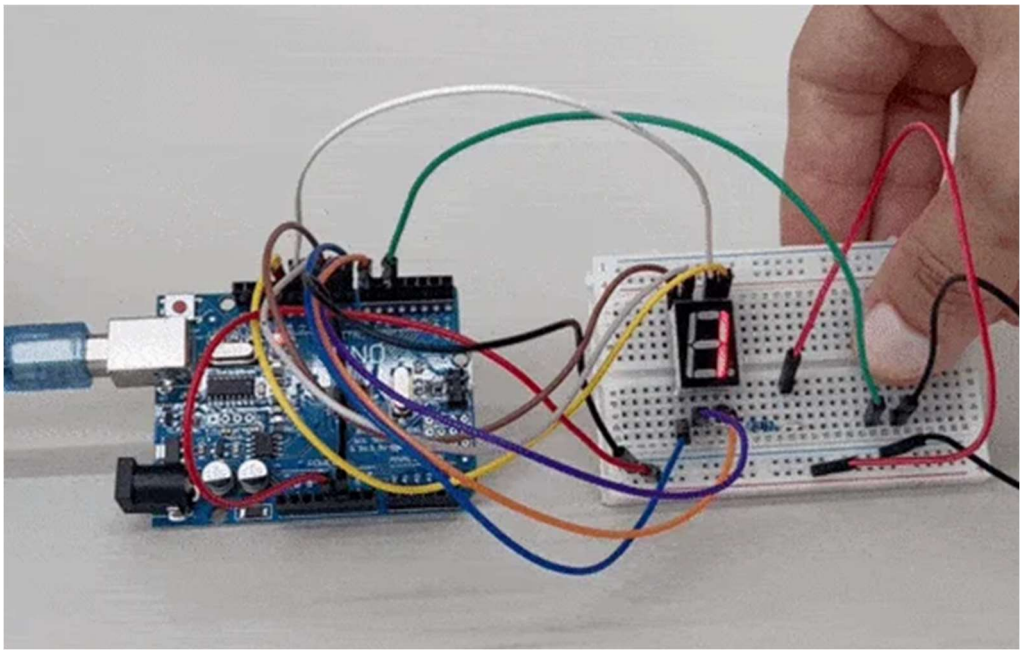
```
  digitalWrite(segE, HIGH);            // Apaga o segmento E do display
  digitalWrite(segD, LOW);             // Acende o segmento D do display
  digitalWrite(segC, LOW);             // Acende o segmento C do display
  digitalWrite(segB, HIGH);            // Apaga o segmento B do display
  digitalWrite(segA, LOW);             // Acende o segmento A do display
  digitalWrite(segF, LOW);            // Acende o segmento F do display
  digitalWrite(segG, LOW);             // Acende o segmento G do display
}

//----------------Declaracao da funcao que mostra o numero 6 no display de 7 segmentos-----------
void acende6() {
  digitalWrite(segE, LOW);             // Acende o segmento E do display
  digitalWrite(segD, LOW);             // Acende o segmento D do display
  digitalWrite(segC, LOW);             // Acende o segmento C do display
  digitalWrite(segB, HIGH);            // Apaga o segmento B do display
  digitalWrite(segA, LOW);             // Acende o segmento A do display
  digitalWrite(segF, LOW);            // Acende o segmento F do display
  digitalWrite(segG, LOW);             // Acende o segmento G do display

}

//----------------Declaracao da funcao que mostra o numero 7 no display de 7 segmentos-----------

void acende7() {
  digitalWrite(segE, HIGH);            // Apaga o segmento E do display
  digitalWrite(segD, HIGH);            // Apaga o segmento D do display
  digitalWrite(segC, LOW);             // Acende o segmento C do display
  digitalWrite(segB, LOW);             // Acende o segmento B do display
  digitalWrite(segA, LOW);             // Acende o segmento A do display
  digitalWrite(segF, HIGH);            // Apaga o segmento F do display
  digitalWrite(segG, HIGH);            // Apaga o segmento G do display
}

//----------------Declaracao da funcao que mostra o numero 8 no display de 7 segmentos-------

void acende8() {
  digitalWrite(segE, LOW);             // Acende o segmento E do display
```

```
  digitalWrite(segD, LOW);            // Acende o segmento D do display
  digitalWrite(segC, LOW);            // Acende o segmento C do display
  digitalWrite(segB, LOW);            // Acende o segmento B do display
  digitalWrite(segA, LOW);            // Acende o segmento A do display
  digitalWrite(segF, LOW);           // Acende o segmento F do display
  digitalWrite(segG, LOW);            // Acende o segmento G do display
}

//----------------Declaracao da funcao que mostra o numero 9 no display de 7 segmentos----------

void acende9() {
  digitalWrite(segE, HIGH);           // Apaga o segmento E do display
  digitalWrite(segD, LOW);            // Acende o segmento D do display
  digitalWrite(segC, LOW);            // Acende o segmento C do display
  digitalWrite(segB, LOW);            // Acende o segmento B do display
  digitalWrite(segA, LOW);            // Acende o segmento A do display
  digitalWrite(segF, LOW);           // Acende o segmento F do display
  digitalWrite(segG, LOW);            // Acende o segmento G do display

}
```

10.3 – Dado Eletrônico

Vamos construir um dado eletrônico que rola ao apertar um botão. Para este projeto iremos juntar dois componentes que já vimos anteriormente, um display de 7 segmentos e um botão.

Ao apertar o botão, o display deverá mostrar números aleatórios até parar, simulando o funcionamento de um dado de 10 lados, com valores de 0 a 9.

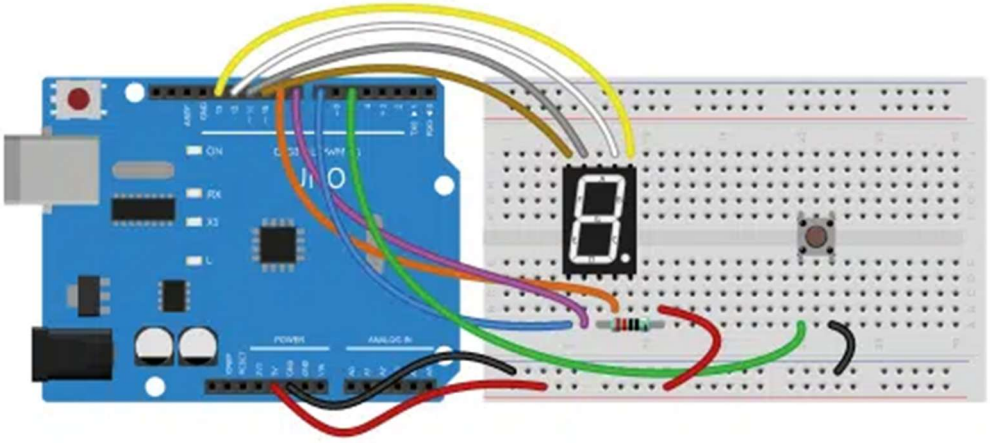


Material necessário

1x Display 7 segmentos
1x Resistor 220 ohm
1x Chave push-button
1x Protoboard 400 pontos
13x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

A montagem vai seguir a conexão do contador digital para o display e acrescentamos um push button para comandar a rolagem do dado.



Programa

Para o programa do dado eletrônico vamos usar a função **switch**() e a função **random**(). A função random() é bastante simples de entender, ela vai ter como saída um número de forma pseudo-aleatória (é muito difícil fazer um número realmente aleatório em microcontroladores ou computadores) no intervalo indicado

random(0, 10)

Nesse caso a função retornará um valor entre 0 e 9. O limite superior será sempre excluído das possibilidades, então você precisa se atentar a isso e sempre colocar um número a mais do que o maior que você espera receber

Para ajudar a fazer um número "mais aleatório" usamos a função **randomSeed**() junto com a função **millis**() dessa forma:

```
79 | randomSeed(millis());
```

A função millis() retorna o valor de milissegundos decorridos desde que o programa iniciou. Dessa forma, usamos como parâmetro de início do algoritmo que gera o número aleatório um número que está sempre mudando, e assim conseguimos sequências diferentes, tendo um pouco mais de imprevisibilidade nos resultados.

Já a função **switch**() serve quando temos várias condições possíveis, assim não precisamos encadear várias vezes a função **if**() e perder tempo testando todas as condições. Criamos os vários casos com a estrutura **case** e assim fazemos uma verificação só para ver em qual condição se encaixa

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Dado eletrônico
// AUTOR: MAKERHERO

//----------------Declara as variaveis do codigo----------------
int segE = 7;                   // Declara a variavel segE e atribui o valor 7 a ela
int segD = 8;                   // Declara a variavel segD e atribui o valor 8 a ela
int segC = 9;                   // Declara a variavel segC e atribui o valor 9 a ela
int segB = 13;                  // Declara a variavel segB e atribui o valor 13 a ela
int segA = 12;                  // Declara a variavel segA e atribui o valor 12 a ela
int segF = 11;                  // Declara a variavel segF e atribui o valor 11 a ela
int segG = 10;                  // Declara a variavel segG e atribui o valor 10 a ela
int pinoBotao = 5;              // Declara a variavel pinoBotao e atribui o valor 5 a ela
//----------------Funcao executada uma vez na inicializacao do sistema----------------
void setup() {
  pinMode(segE, OUTPUT);            // Configura a porta 7 (valor da variável segE) como saida
  pinMode(segD, OUTPUT);            // Configura a porta 8 (valor da variável segD) como saida
  pinMode(segC, OUTPUT);            // Configura a porta 9 (valor da variável segC) como saida
  pinMode(segB, OUTPUT);            // Configura a porta 13 (valor da variável segB) como saida
  pinMode(segA, OUTPUT);            // Configura a porta 12 (valor da variável segA) como saida
  pinMode(segF, OUTPUT);            // Configura a porta 11 (valor da variável segF) como saida
  pinMode(segG, OUTPUT);            // Configura a porta 10 (valor da variável segG) como saida
  pinMode(pinoBotao, INPUT_PULLUP);     // Configura a porta 7 (valor da variável pinoBotao) como
entrada com resistor de Pull-Up integrado

}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------
```

```
void loop() {
 if(digitalRead(pinoBotao) == LOW)       // Se o botao estiver apertado (leitura igual a LOW), executa as instrucoes entre chaves

 {
  jogaDado();                    // Chama a funcao jogaDado
 }
}
void jogaDado() {
 randomSeed(millis());
 for(int i=0; i<25; i++) {            // Repete os comandos dentro das chaves 24 vezes
  switch(random(0, 10)) {             // Escolhe um numero aleatorio entre 0 e 9 e chama o case referente a esse número
//----------------Se o numero sorteado for zero----------------
    case 0 :
     acende0();                  // Chama a função que acende o digito 0 no display
     break;
//----------------Se o numero sorteado for um----------------

    case 1 :

     acende1();                  // Chama a função que acende o digito 1 no display

     break;

//----------------Se o numero sorteado for dois----------------

    case 2 :
```

```
      acende2();                // Chama a função que acende o digito 2 no display
      break;
//----------------Se o numero sorteado for tres----------------

    case 3 :
      acende3();                // Chama a função que acende o digito 3 no display
      break;

//----------------Se o numero sorteado for quatro----------------
    case 4 :
      acende4();                // Chama a função que acende o digito 4 no display
      break;

//----------------Se o numero sorteado for cinco----------------

    case 5 :
      acende5();                // Chama a função que acende o digito 5 no display
      break;

//----------------Se o numero sorteado for seis----------------

    case 6 :
      acende6();                // Chama a função que acende o digito 6 no display
      break;

//----------------Se o numero sorteado for sete----------------
```

```
    case 7 :
     acende7();                  // Chama a função que acende o digito 7 no display
     break;

//----------------Se o numero sorteado for oito----------------

    case 8 :
     acende8();                  // Chama a função que acende o digito 8 no display
     break;

//----------------Se o numero sorteado for nove----------------

    case 9 :
     acende9();                  // Chama a função que acende o digito 9 no display
     break;
   }
   delay(4*i);                   // Aguarda 4*i milissegundos (onde i é cada iteracao do laco for, variando de 0 a
24)
 }
}

//----------------Declaracao da funcao que mostra o numero 0 no display de 7 segmentos----------------
void acende0() {
  digitalWrite(segE, LOW);             // Acende o segmento E do display
  digitalWrite(segD, LOW);             // Acende o segmento D do display
  digitalWrite(segC, LOW);             // Acende o segmento C do display
  digitalWrite(segB, LOW);             // Acende o segmento B do display
```

```
digitalWrite(segA, LOW);              // Acende o segmento A do display
 digitalWrite(segF, LOW);             // Acende o segmento F do display
 digitalWrite(segG, HIGH);            // Apaga o segmento G do display
}

//----------------Declaracao da funcao que mostra o numero 1 no display de 7 segmentos----------------

void acende1() {
 digitalWrite(segE, HIGH);            // Apaga o segmento E do display
 digitalWrite(segD, HIGH);            // Apaga o segmento D do display
 digitalWrite(segC, LOW);             // Acende o segmento C do display
 digitalWrite(segB, LOW);             // Acende o segmento B do display
 digitalWrite(segA, HIGH);            // Apaga o segmento A do display
 digitalWrite(segF, HIGH);            // Apaga o segmento F do display
 digitalWrite(segG, HIGH);            // Apaga o segmento G do display
}

//----------------Declaracao da funcao que mostra o numero 2 no display de 7 segmentos----------------

void acende2() {
 digitalWrite(segE, LOW);             // Acende o segmento E do display
 digitalWrite(segD, LOW);             // Acende o segmento D do display
 digitalWrite(segC, HIGH);            // Apaga o segmento C do display
 digitalWrite(segB, LOW);             // Acende o segmento B do display
 digitalWrite(segA, LOW);             // Acende o segmento A do display
```

```
  digitalWrite(segF, HIGH);             // Apaga o segmento F do display
  digitalWrite(segG, LOW);              // Acende o segmento G do display

}

//----------------Declaracao da funcao que mostra o numero 3 no display de 7 segmentos----------------

void acende3() {
  digitalWrite(segE, HIGH);             // Apaga o segmento E do display
  digitalWrite(segD, LOW);              // Acende o segmento D do display
  digitalWrite(segC, LOW);              // Acende o segmento C do display
  digitalWrite(segB, LOW);              // Acende o segmento B do display
  digitalWrite(segA, LOW);              // Acende o segmento A do display
  digitalWrite(segF, HIGH);             // Apaga o segmento F do display
  digitalWrite(segG, LOW);              // Acende o segmento G do display
}

//----------------Declaracao da funcao que mostra o numero 4 no display de 7 segmentos----------------

void acende4() {
digitalWrite(segE, HIGH);             // Apaga o segmento E do display
  digitalWrite(segD, HIGH);             // Apaga o segmento D do display
digitalWrite(segC, LOW);              // Acende o segmento C do display
digitalWrite(segB, LOW);              // Acende o segmento B do display
```

```
 digitalWrite(segA, HIGH);            // Apaga o segmento A do display
digitalWrite(segF, LOW);             // Acende o segmento F do display
digitalWrite(segG, LOW);             // Acende o segmento G do display
}
//----------------Declaracao da funcao que mostra o numero 5 no display de 7 segmentos----------------

void acende5() {
digitalWrite(segE, HIGH);            // Apaga o segmento E do display
 digitalWrite(segD, LOW);             // Acende o segmento D do display
digitalWrite(segC, LOW);             // Acende o segmento C do display
 digitalWrite(segB, HIGH);            // Apaga o segmento B do display
 digitalWrite(segA, LOW);             // Acende o segmento A do display
 digitalWrite(segF, LOW);             // Acende o segmento F do display
 digitalWrite(segG, LOW);             // Acende o segmento G do display

}

//----------------Declaracao da funcao que mostra o numero 6 no display de 7 segmentos----------------

void acende6() {
 digitalWrite(segE, LOW);             // Acende o segmento E do display
digitalWrite(segD, LOW);             // Acende o segmento D do display
 digitalWrite(segC, LOW);             // Acende o segmento C do display
 digitalWrite(segB, HIGH);            // Apaga o segmento B do display
digitalWrite(segA, LOW);             // Acende o segmento A do display
```

```
 digitalWrite(segF, LOW);              // Acende o segmento F do display
 digitalWrite(segG, LOW);              // Acende o segmento G do display
}
//----------------Declaracao da funcao que mostra o numero 7 no display de 7 segmentos----------------
void acende7() {
digitalWrite(segE, HIGH);             // Apaga o segmento E do display
digitalWrite(segD, HIGH);             // Apaga o segmento D do display
digitalWrite(segC, LOW);              // Acende o segmento C do display
 digitalWrite(segB, LOW);              // Acende o segmento B do display

digitalWrite(segA, LOW);              // Acende o segmento A do display
digitalWrite(segF, HIGH);             // Apaga o segmento F do display
digitalWrite(segG, HIGH);             // Apaga o segmento G do display
}
//----------------Declaracao da funcao que mostra o numero 8 no display de 7 segmentos----------------
void acende8() {
 digitalWrite(segE, LOW);              // Acende o segmento E do display
 digitalWrite(segD, LOW);              // Acende o segmento D do display
 digitalWrite(segC, LOW);              // Acende o segmento C do display
digitalWrite(segB, LOW);              // Acende o segmento B do display
 digitalWrite(segA, LOW);              // Acende o segmento A do display
 digitalWrite(segF, LOW);              // Acende o segmento F do display
digitalWrite(segG, LOW);              // Acende o segmento G do display
}

//----------------Declaracao da funcao que mostra o numero 9 no display de 7 segmentos----------------

void acende9() {
```

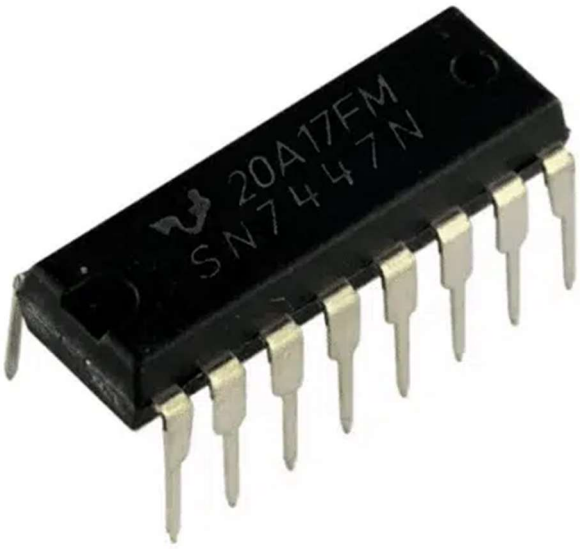
```
digitalWrite(segE, HIGH);          // Apaga o segmento E do display
digitalWrite(segD, LOW);           // Acende o segmento D do display
 digitalWrite(segC, LOW);           // Acende o segmento C do display
 digitalWrite(segB, LOW);           // Acende o segmento B do display
 digitalWrite(segA, LOW);           // Acende o segmento A do display
 digitalWrite(segF, LOW);           // Acende o segmento F do display
digitalWrite(segG, LOW);           // Acende o segmento G do display
}
```

11 – CI 7447

Módulo 11 – CI 7447 – Você irá aprender a usar menos portas para acionar o display de 7 segmentos e libere portas para outras funções do seu projeto com o CI que é feito especificamente para acionar displays de 7 segmentos.

11.1 – O que é o CI 7447?

O CI 7447 é um componente eletrônico que recebe um sinal de 4 bits. Opa, mas espera aí, o que são bits? Bits são cada unidade do sistema binário, onde só temos dois valores, 0 ou 1, como o sinal digital. Com mais bits formamos números maiores, só que dessa vez ao invés de dezenas ou centenas, temos potências de dois. Em binário, 1 ainda é 1, mas 10 equivale a 2 no sistema decimal que estamos acostumados, 100 equivale a 4 e assim por diante.



Então voltando ao CI7447, ele recebe um sinal de 4 bits, e então converte nas suas saídas para os sinais necessários para mostrar aquele número no display de 7 segmentos.

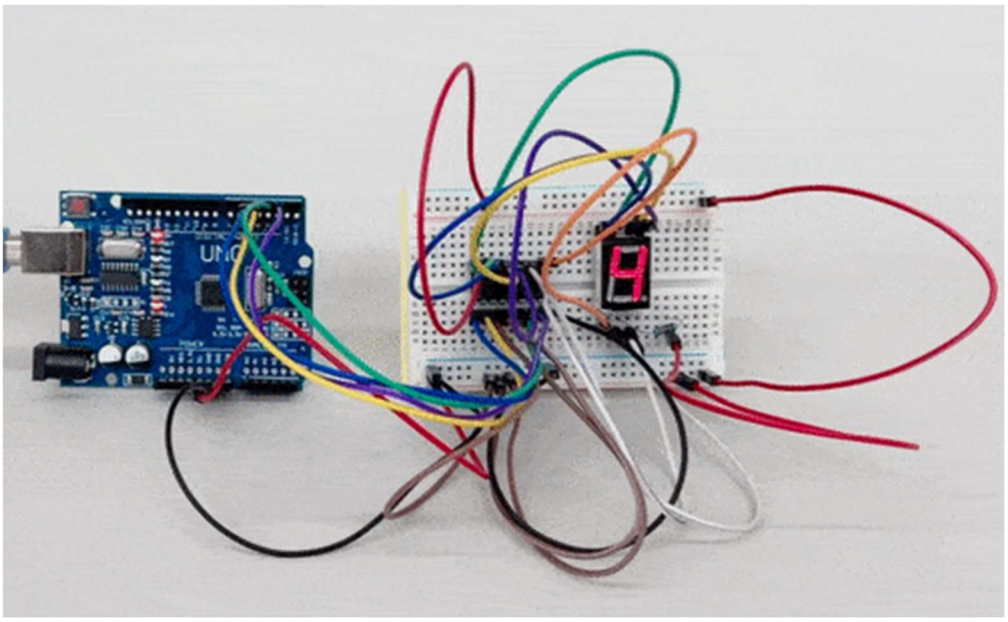
Então com ele conseguimos economizar 3 portas do Arduino para acionarmos outras coisas ou recebermos informações de algum outro sensor.

Se mandarmos, por exemplo, o número 0101 (você consegue dizer qual seria o número em decimal?) para o CI com as portas do Arduino, queremos que apareça o número 5 no display, e o CI vai ligar ou desligar suas saídas de acordo, assim como fizemos no programa para controlar o display direto com o Arduino.

Esse CI é muito usado em conjunto com o display de 7 segmentos exatamente porque com ele podemos poupar portas e construir projetos mais complexos sem precisar necessariamente de um microcontrolador maior ou com mais portas.

11.2 – Contador Digital 2.0

Percebeu que usamos muitas portas para controlar o display?



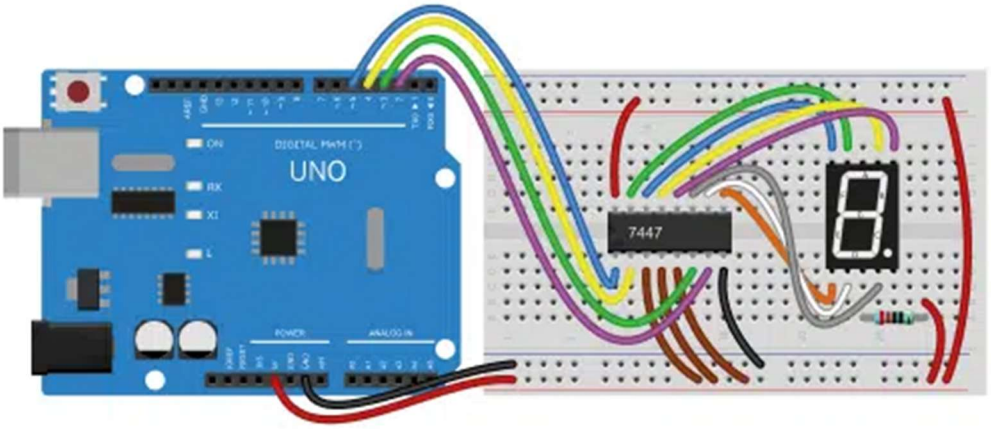
Vamos revisitar o exercício do contador, mas dessa vez vamos usar o CI 7447 para economizar algumas portas do arduino. Componentes externos mais complexos como CIs e módulos podem ajudar a usar menos portas para uma tarefa, como é o caso desse exemplo, ou até mesmo para acrescentar funcionalidades que o arduino não possui. Essa é a grande ideia e poder do arduino, conseguimos combinar outros componentes quase como um lego eletrônico para fazer projetos e funções mais complexas, mas ainda com a simplicidade do código do Arduino

Material necessário

1x Display 7 segmentos
1x CI 7447
1x Resistor 220 ohm
1x Protoboard 400 pontos
16x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

O CI deve ser encaixado na protoboard com um lado dos pinos para cima da cavidade central e o outro lado dos pinos para baixo do friso central. Caso esteja difícil de encaixar, aperte levemente os pinos para que eles fechem um pouco e encaixem mais facilmente. Atente-se também para a posição do CI, que deve ficar com a bolinha ou U em baixo relevo em um dos lados para a esquerda.



Programa

Neste exercício não teremos nenhuma nova função na programação.

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Contador digital 2.0

// AUTOR: MAKERHERO

//----------------Declara as variaveis do codigo----------------

int pinoA0 = 2;                    // Declara a variavel pinoA0 e atribui o valor 2 a ela

int pinoA1 = 5;                    // Declara a variavel pinoA1 e atribui o valor 3 a ela

int pinoA2 = 4;                    // Declara a variavel pinoA2 e atribui o valor 4 a ela

int pinoA3 = 3;                   // Declara a variavel pinoA3 e atribui o valor 5 a ela


//----------------Funcao executada uma vez na inicializacao do sistema----------------


void setup()


{

 pinMode(pinoA0, OUTPUT);              // Configura a porta 2 (valor da variável pinoA0) como saida
```

```
 pinMode(pinoA1, OUTPUT);            // Configura a porta 3 (valor da variável pinoA1) como saida
 pinMode(pinoA2, OUTPUT);            // Configura a porta 4 (valor da variável pinoA2) como saida
 pinMode(pinoA3, OUTPUT);            // Configura a porta 5 (valor da variável pinoA3) como saida
}
//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------
void loop()
{

 acende0();                 // Chama a função que acende o digito 0 no display
 delay(1000);                // Aguarda 1000 milissegundos (1 segundo)
 acende1();                 // Chama a função que acende o digito 1 no display
 delay(1000);                // Aguarda 1000 milissegundos (1 segundo)
 acende2();                 // Chama a função que acende o digito 2 no display
 delay(1000);                // Aguarda 1000 milissegundos (1 segundo)
 acende3();                 // Chama a função que acende o digito 3 no display
 delay(1000);                // Aguarda 1000 milissegundos (1 segundo)
 acende4();                 // Chama a função que acende o digito 4 no display
 delay(1000);                // Aguarda 1000 milissegundos (1 segundo)
 acende5();                 // Chama a função que acende o digito 5 no display
 delay(1000);                // Aguarda 1000 milissegundos (1 segundo)
 acende6();                 // Chama a função que acende o digito 6 no display
 delay(1000);                // Aguarda 1000 milissegundos (1 segundo)



 acende7();                 // Chama a função que acende o digito 7 no display
 delay(1000);                // Aguarda 1000 milissegundos (1 segundo)
 acende8();                 // Chama a função que acende o digito 8 no display
```

```
  delay(1000);                  // Aguarda 1000 milissegundos (1 segundo)
  acende9();                    // Chama a função que acende o digito 9 no display
  delay(1000);                  // Aguarda 1000 milissegundos (1 segundo)

}

//----------------Declaracao da funcao que mostra o numero 0 no display de 7 segmentos----------------
void acende0(){
  digitalWrite(pinoA0, LOW);          // Envia o valor 0 (LOW) ao pino A0 do CI 7447
  digitalWrite(pinoA1, LOW);          // Envia o valor 0 (LOW) ao pino A1 do CI 7447
  digitalWrite(pinoA2, LOW);          // Envia o valor 0 (LOW) ao pino A2 do CI 7447
  digitalWrite(pinoA3, LOW);          // Envia o valor 0 (LOW) ao pino A3 do CI 7447
}

//----------------Declaracao da funcao que mostra o numero 1 no display de 7 segmentos----------------

void acende1(){
  digitalWrite(pinoA0, HIGH);         // Envia o valor 1 (HIGH) ao pino A0 do CI 7447

  digitalWrite(pinoA1, LOW);          // Envia o valor 0 (LOW) ao pino A1 do CI 7447
  digitalWrite(pinoA2, LOW);          // Envia o valor 0 (LOW) ao pino A2 do CI 7447
  digitalWrite(pinoA3, LOW);          // Envia o valor 0 (LOW) ao pino A3 do CI 7447
}
//----------------Declaracao da funcao que mostra o numero 2 no display de 7 segmentos----------------

void acende2(){
  digitalWrite(pinoA0, LOW);          // Envia o valor 0 (LOW) ao pino A0 do CI 7447
  digitalWrite(pinoA1, HIGH);         // Envia o valor 0 (HIGH) ao pino A1 do CI 7447
```

```
  digitalWrite(pinoA2, LOW);          // Envia o valor 0 (LOW) ao pino A2 do CI 7447
  digitalWrite(pinoA3, LOW);          // Envia o valor 0 (LOW) ao pino A3 do CI 7447
}

//----------------Declaracao da funcao que mostra o numero 3 no display de 7 segmentos----------------

void acende3() {
  digitalWrite(pinoA0, HIGH);         // Envia o valor 0 (HIGH) ao pino A0 do CI 7447
  digitalWrite(pinoA1, HIGH);         // Envia o valor 0 (HIGH) ao pino A1 do CI 7447
  digitalWrite(pinoA2, LOW);          // Envia o valor 0 (LOW) ao pino A2 do CI 7447
  digitalWrite(pinoA3, LOW);          // Envia o valor 0 (LOW) ao pino A3 do CI 7447
}

//----------------Declaracao da funcao que mostra o numero 4 no display de 7 segmentos----------------

void acende4() {
  digitalWrite(pinoA0, LOW);          // Envia o valor 0 (LOW) ao pino A0 do CI 7447
  digitalWrite(pinoA1, LOW);          // Envia o valor 0 (LOW) ao pino A1 do CI 7447

  digitalWrite(pinoA2, HIGH);         // Envia o valor 1 (HIGH) ao pino A2 do CI 7447
  digitalWrite(pinoA3, LOW);          // Envia o valor 0 (LOW) ao pino A3 do CI 7447
}
//----------------Declaracao da funcao que mostra o numero 5 no display de 7 segmentos----------------

void acende5() {
  digitalWrite(pinoA0, HIGH);         // Envia o valor 1 (HIGH) ao pino A0 do CI 7447
  digitalWrite(pinoA1, LOW);          // Envia o valor 0 (LOW) ao pino A1 do CI 7447
  digitalWrite(pinoA2, HIGH);         // Envia o valor 1 (HIGH) ao pino A2 do CI 7447
```

```
  digitalWrite(pinoA3, LOW);          // Envia o valor 0 (LOW) ao pino A3 do CI 7447
}
//----------------Declaracao da funcao que mostra o numero 6 no display de 7 segmentos----------------

void acende6() {
  digitalWrite(pinoA0, LOW);          // Envia o valor 0 (LOW) ao pino A0 do CI 7447
  digitalWrite(pinoA1, HIGH);         // Envia o valor 1 (HIGH) ao pino A1 do CI 7447
  digitalWrite(pinoA2, HIGH);         // Envia o valor 1 (HIGH) ao pino A2 do CI 7447
  digitalWrite(pinoA3, LOW);          // Envia o valor 0 (LOW) ao pino A3 do CI 7447
}
//----------------Declaracao da funcao que mostra o numero 7 no display de 7 segmentos----------------

void acende7() {
  digitalWrite(pinoA0, HIGH);         // Envia o valor 1 (HIGH) ao pino A0 do CI 7447
  digitalWrite(pinoA1, HIGH);         // Envia o valor 1 (HIGH) ao pino A1 do CI 7447
  digitalWrite(pinoA2, HIGH);         // Envia o valor 1 (HIGH) ao pino A2 do CI 7447
  digitalWrite(pinoA3, LOW);          // Envia o valor 0 (LOW) ao pino A3 do CI 7447

}

//----------------Declaracao da funcao que mostra o numero 8 no display de 7 segmentos----------------

void acende8() {
  digitalWrite(pinoA0, LOW);          // Envia o valor 0 (LOW) ao pino A0 do CI 7447
  digitalWrite(pinoA1, LOW);          // Envia o valor 0 (LOW) ao pino A1 do CI 7447
  digitalWrite(pinoA2, LOW);          // Envia o valor 0 (LOW) ao pino A2 do CI 7447
  digitalWrite(pinoA3, HIGH);         // Envia o valor 1 (HIGH) ao pino A3 do CI 7447
}
```

```
//----------------Declaracao da funcao que mostra o numero 9 no display de 7 segmentos----------------
void acende9() {


  digitalWrite(pinoA0, HIGH);          // Envia o valor 1 (HIGH) ao pino A0 do CI 7447
  digitalWrite(pinoA1, LOW);           // Envia o valor 0 (LOW) ao pino A1 do CI 7447
  digitalWrite(pinoA2, LOW);           // Envia o valor 0 (LOW) ao pino A2 do CI 7447
  digitalWrite(pinoA3, HIGH);          // Envia o valor 1 (HIGH) ao pino A3 do CI 7447
}
```

12 – Sensor ultrassônico

Módulo 12 – Sensor Ultrassonico – Saiba como medir distâncias e detectar movimentos com o sensor ultrassônico, um sensor com o mesmo princípio de funcionamento de sensores como os que detectam obstáculos ao engatar a ré nos carros que contam com essa funcionalidade.

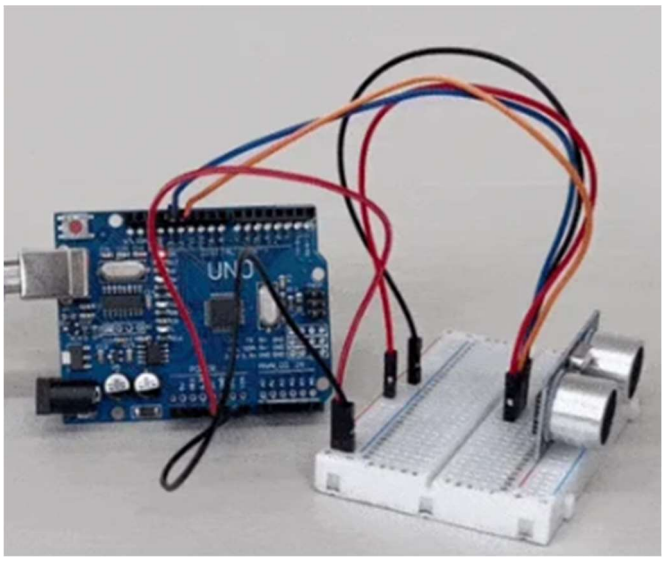
12.1 – Conhecendo o sensor ultrassônico

Você já deve ter visto alguns carros que emitem bipes dentro da cabine quando estão dando ré para auxiliar o motorista a perceber obstáculos e julgar a distância entre a parte de trás do carro e um outro carro ou uma parede. Um dos grandes responsáveis pelo funcionamento desse sistema é o sensor ultrassônico.

Esse sensor é um conjunto especial de microfone e alto falante, que emite pulsos de som em frequências acima das que o ouvido humano consegue perceber. O som então reflete nos obstáculos, como se fosse um eco que ouvimos, e com a diferença de tempo entre a emissão do pulso e a recepção do eco, é possível calcular a distância em que o objeto está.

12.2 – Trena digital

No exercício anterior vimos como utilizar um CI para otimizar o acionamento de um display com o arduino. Agora, vamos ver como podemos conectar um módulo diferente para fazer uma função bastante conhecida no mundo maker, medição de distâncias com um sensor ultrassônico.

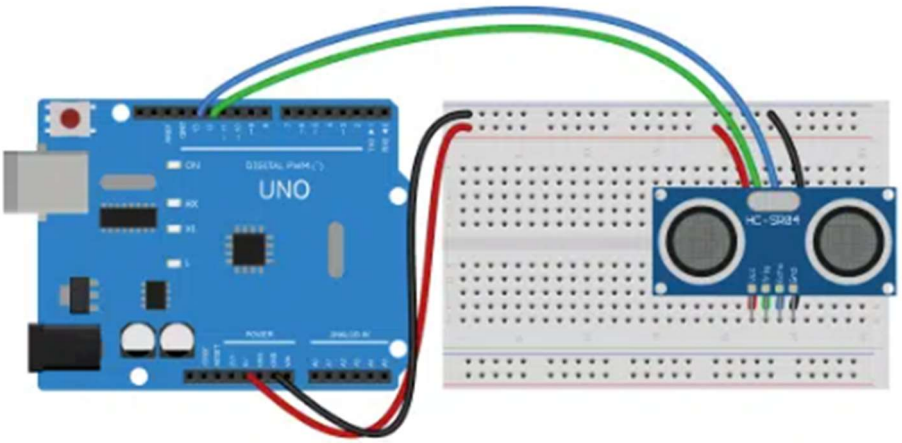


Material necessário

1x Sensor Ultrassônico
1x Protoboard 400 pontos
6x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

Para a trena digital encaixe os pinos do sensor ultrassônico na protoboard como mostrado abaixo:

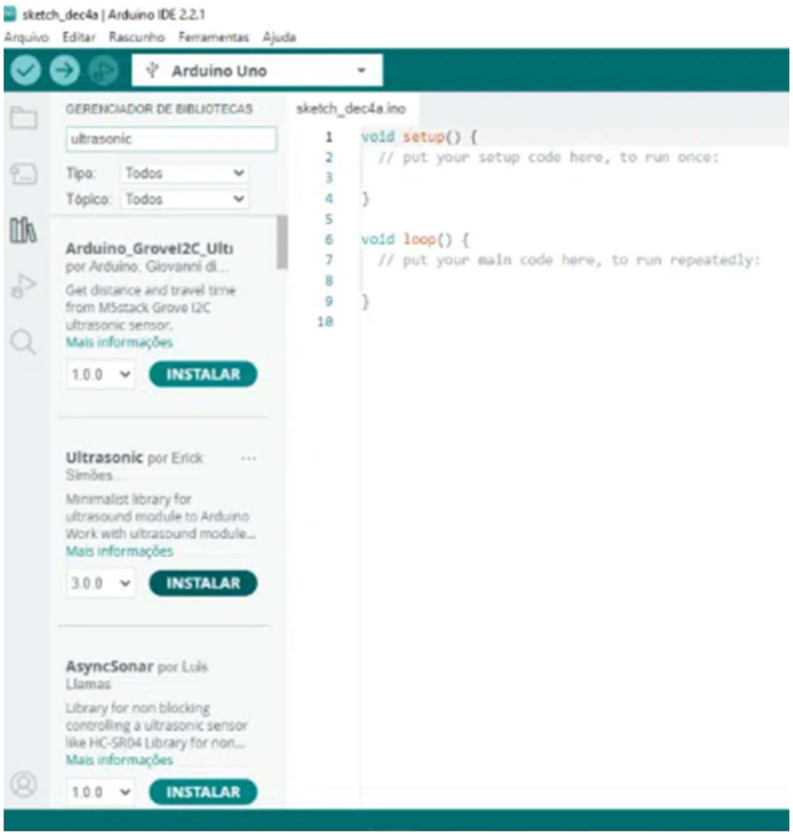


Programa

Para a saída da medição vamos usar um recurso do arduino que é a comunicação serial. A IDE possui um monitor serial, onde podemos ver as mensagens que o arduino está transmitindo.

Novamente como no caso do servo motor, usaremos uma biblioteca para nos ajudar com a comunicação com o sensor. Dessa vez a biblioteca não está instalada por padrão na IDE, então precisamos fazer a instalação dela.

Para isso, vá em **Ferramentas** -> **Gerenciador de bibliotecas** e busque por **Ultrasonic** (por Erick Simões). Clique em **instalar**



A IDE informará quando finalizar a instalação e você já poderá usar normalmente.

Essa é a forma mais comum de se instalar bibliotecas na IDE para auxiliar o uso de sensores e módulos.

Para usar a comunicação serial precisamos iniciá-la no **setup**() da seguinte maneira:

```
25 Serial.begin(9600);                    // Inicializa a interface serial com a velocidade 9600
```

A velocidade é um parâmetro que pode ser alterado dependendo de com o que irá se comunicar, alguns dispositivos têm velocidades específicas. Para o nosso tutorial usaremos a velocidade de 9600

Depois de iniciada a comunicação serial, usamos um dos comandos abaixo para fazer a saída do arduino para o monitor serial:

```
36 Serial.print(F("Distancia em cm: "));          // Escreve na saida serial o texto entre aspas
```

Esse comando vai imprimir o texto entre aspas duplas no monitor serial e não pular a linha nem deixar espaço extra, então o próximo comando de print, seja ele qual for, será feito na mesma linha de forma contínua

Já se quisermos que o próximo texto pule para a próxima linha, usamos o comando abaixo:

```
    Serial.println(distancia);                    // Escreve na saida serial o valor da variavel distancia e
38  passa para a próxima linha
```

Perceba que agora não usamos as aspas duplas, então a saída desse print será o valor da variável distância

As duas linhas acima em sequência vão gerar a seguinte saída no monitor serial, supondo que o valor da variável distância seja 10

Distancia em cm: 10

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Trena Digital

// AUTOR: MAKERHERO

//----------------Inclui as bibliotecas do codigo----------------


#include <Ultrasonic.h>                    // Inclui a biblioteca Ultrassonic facilitar o controle do
sensor de distancia



//----------------Declara as variaveis do codigo----------------
int pinoTrigger = 12;                      // Declara a variavel pinoTrigger e atribui o valor 12 a ela
int pinoEcho = 13;                         // Declara a variavel pinoTrigger e atribui o valor 12 a ela
int distancia;                             // Declara a variavel distancia sem atribuir valor a ela


Ultrasonic sensorUltrassonico(pinoTrigger, pinoEcho);        // Declara o objeto sensorUltrassonico
para utilizar as funcoes da biblioteca Ultrassonic
//----------------Funcao executada uma vez na inicializacao do sistema----------------
void setup() {
 Serial.begin(9600);                       // Inicializa a interface serial com a velocidade 9600
```

```
}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------

void loop() {

  distancia = sensorUltrassonico.read();                  // Le o valor informado pelo sensor ultrassonico e atribui o valor lido a variavel distancia

  Serial.print(F("Distancia em cm: "));                   // Escreve na saida serial o texto entre aspas

  Serial.println(distancia);                              // Escreve na saida serial o valor da variavel distancia e passa para a próxima linha

  delay(1000);                                            // Aguarda 1000 milissegundos (1 segundo)

}
```

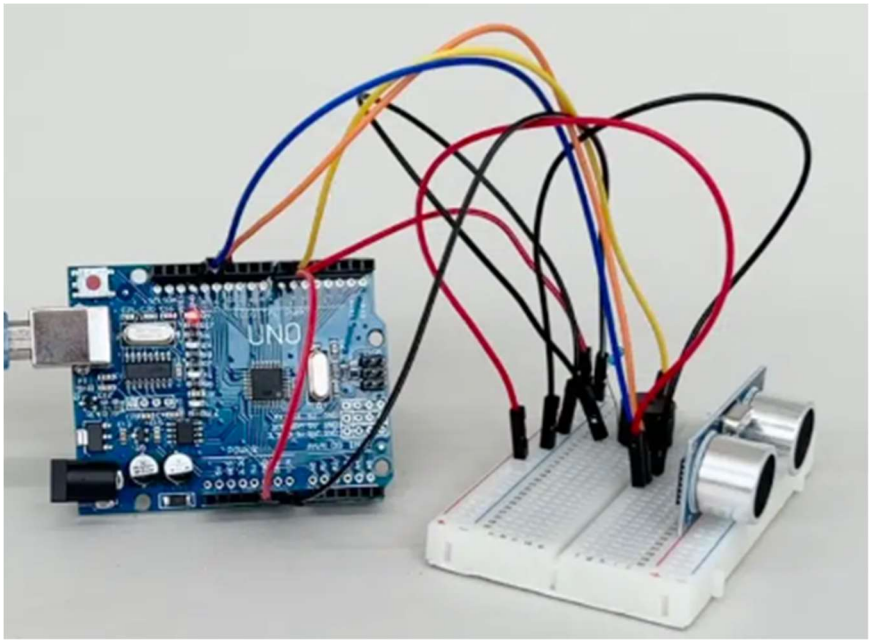
Reparou que não acentuamos a palavra distância, e nem nenhuma palavra nos códigos até agora? Às vezes o sistema de caracteres de um computador e de outro não são compatíveis, então evitamos utilizar caracteres especiais e acentos nos códigos para não haver problemas e termos caracteres estranhos nas saídas, ou até mesmo comportamentos inesperados do nosso programa.

Com o monitor serial conseguimos mostrar informações mais complexas e também realizar testes se as funções estão agindo da maneira esperada.

12.3 – Alarme de Movimento

Neste projeto, faremos um alarme que apita quando detecta movimento. Para o apito utilizaremos o já conhecido buzzer, e para detectar o movimento iremos utilizar o sensor de

distância ultrassônico. Se houver variação na distância lida pelo sensor, é detectado como movimento e soa o buzzer.

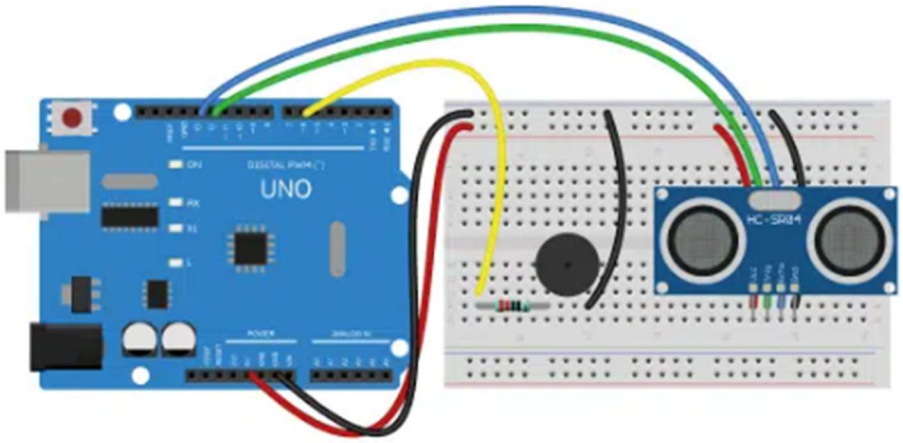


Material necessário

1x Sensor Ultrassônico
1x Buzzer
1x Protoboard 400 pontos
7x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

Monte o projeto como abaixo encaixando os pinos do sensor ultrassônico na protoboard.



Programa

Para a programação do alarme de movimento não teremos nenhuma novidade nas funções.

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Alarme de Movimento

// AUTOR: MAKERHERO

//----------------Inclui as bibliotecas do codigo----------------

#include <Ultrasonic.h>                    // Inclui a biblioteca Ultrassonic facilitar o controle do sensor de distancia

//----------------Declara as variaveis do codigo----------------

int pinoTrigger = 12;                      // Declara a variavel pinoTrigger e atribui o valor 12 a ela

int pinoEcho = 13;                         // Declara a variavel pinoTrigger e atribui o valor 12 a ela

int pinoBuzzer = 6;                        // Declara a variavel pinoBuzzer e atribui o valor 6 a ela

int distancia;                             // Declara a variavel distancia sem atribuir valor a ela

int distanciaAnterior;                     // Declara a variavel distanciaAnterior sem atribuir valor a ela
```

```
bool iniciando;                                    // Declara a variavel iniciando sem atribuir valor a ela

Ultrasonic sensorUltrassonico(pinoTrigger, pinoEcho);                    // Declara o objeto sensorUltrassonico para utilizar as funcoes da biblioteca Ultrassonic

//----------------Funcao executada uma vez na inicializacao do sistema----------------

void setup() {

  iniciando = true;                                // Atribui o valor verdadeiro(true) a variavel iniciando
  pinMode(pinoBuzzer, OUTPUT);                                // Configura a porta 6 (valor da variavel pinoBuzzer) como saida
}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------
void loop() {
  distancia = sensorUltrassonico.read();                                // Le o valor informado pelo sensor ultrassonico e atribui o valor lido a variavel distancia
  if (iniciando) {                                // Se iniciando for verdadeiro, executa as instrucoes entre chaves
    distanciaAnterior = distancia;                                // Atribui o valor de distancia a variavel distanciaAnterior
    iniciando = false;                                // Atribui o valor falso(false) a variavel iniciando

  }

  if (distancia <=  distanciaAnterior-2 || distancia > distanciaAnterior+2){       // Se a distancia for menor que a distancia anterior menos dois ou for maior que a distancia anterior mais dois, executa as instruções entre chaves
```

```
    for(int i=0; i< 10; i++){                    // Repete as instruções entre chaves 10 vezes
     tone(pinoBuzzer, 440);                      // Aciona o Buzzer na frequencia de 440 Hz (La)
     delay(200);                                 // Aguarda 200 milissegundos
     tone(pinoBuzzer, 523);                      // Aciona o Buzzer na frequencia de 523 Hz (Do)
     delay(200);                                 // Aguarda 200 milissegundos
    }

    distanciaAnterior = sensorUltrassonico.read();          // Le o valor informado pelo sensor
ultrassonico e atribui o valor lido a variavel distanciaAnterior
    noTone(pinoBuzzer);                          // Desliga o Buzzer
   }
   delay(200);                                   // Aguarda 200 milissegundos

  }
```

12.4 – Projeto: Cancela Eletrônica

Com o sensor de distância e o servo motor, vamos simular o funcionamento de uma cancela de estacionamento? Não será tão complexa quanto a real que precisa passar um cartão ou retirar um ticket, mas ela vai detectar quando algo se aproximar, dará um pequeno atraso, e abrirá, ficando aberta enquanto detectar um objeto dentro da distância programada. Após a saída do objeto, aguardará alguns segundos e então fechará novamente.

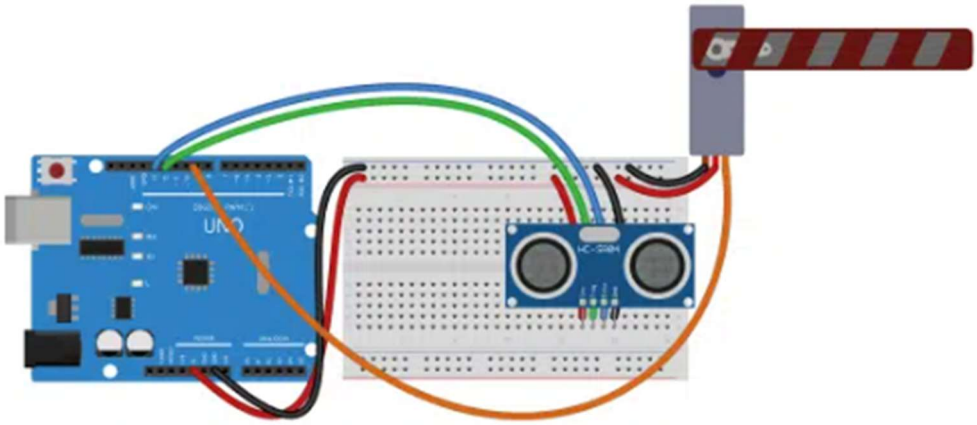
Material necessário

1x Sensor Ultrassônico
1x Servo motor
1x Cancela 3D
1x Protoboard 400 pontos
7x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

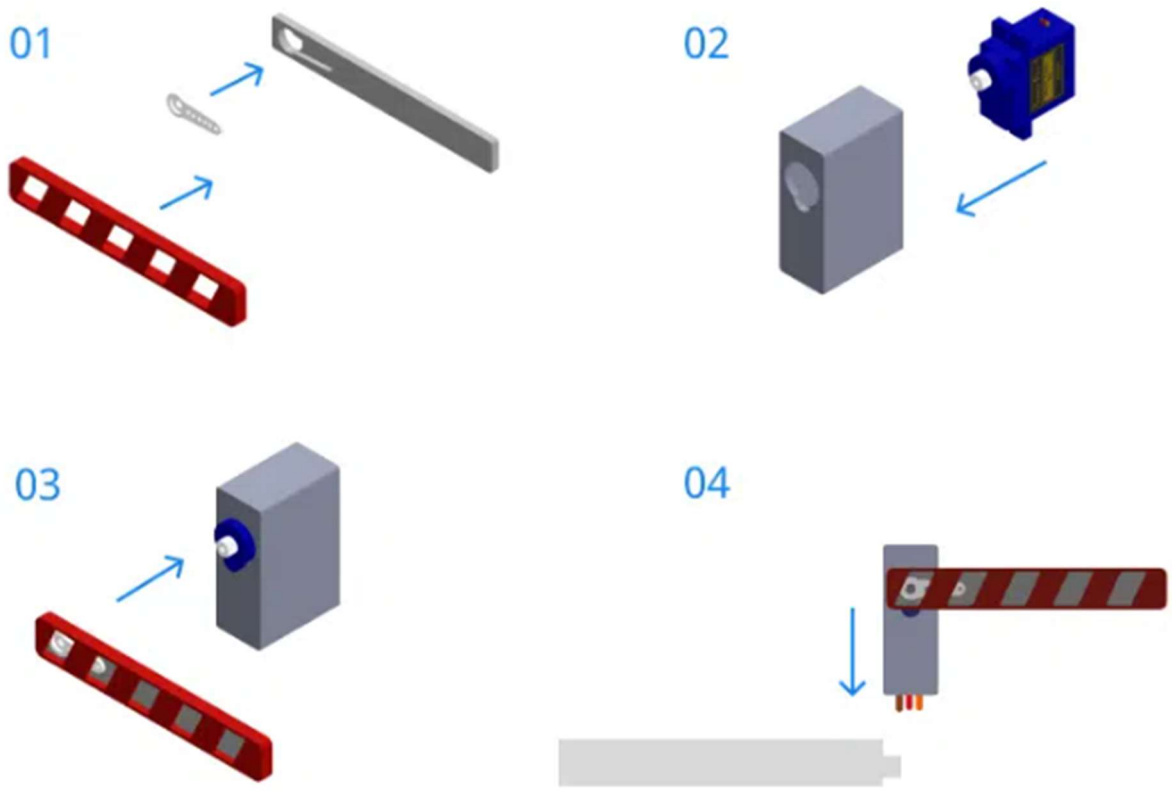
Monte o projeto como abaixo encaixando os pinos do sensor ultrassônico na protoboard.

Use os jumpers para fazer a conexão com o servo motor como nos exercícios anteriores.



Montagem Cancela

Para montar a cancela com componentes e peças em impressão 3D, siga os seguintes passos:



Programa

Para que a cancela aguarde enquanto houver alguma coisa na detecção do sensor usaremos a estrutura **do-while**. Já vimos o **while**() no exercício do interruptor liga e desliga, a estrutura **do-while** não é muito diferente. A principal questão é que ela irá executar as instruções uma vez sem checar a condição, e depois da primeira execução irá checar a condição e repetir as instruções caso a condição seja atendida, e repetirá esse laço até que a condição não seja mais atendida.

do{ // Executa as instrucoes entre chaves uma vez e depois verifica a condição do while

delay(200); // Aguarda 200 milissegundos

distancia = sensorUltrassonico.read(); // Le o valor informado pelo sensor ultrassonico e atribui o valor lido a variavel distancia

} **while** (distancia <= distanciaAcionamento); // Checa se a condicao foi atendida, repete os comandos entre chaves do "do" acima ate que a condicao não seja mais verdadeira, entao segue com o codigo

Perceba que agora as chaves ficam após o **do** e após o while temos apenas o **;** para indicar o fim da instrução.
Veja o código completo abaixo:

```
// CODIGO: Cancela Eletronica

// AUTOR: MAKERHERO
//----------------Inclui as bibliotecas do codigo----------------

#include <Ultrasonic.h>                    // Inclui a biblioteca Ultrassonic facilitar o controle do sensor de distancia

#include <Servo.h>                         // Inclui a biblioteca Servo para facilitar o controle do servo motor

//----------------Declara as variaveis do codigo----------------

int pinoTrigger = 12;                      // Declara a variavel pinoTrigger e atribui o valor 12 a ela

int pinoEcho = 13;                         // Declara a variavel pinoTrigger e atribui o valor 13 a ela

int pinoServo = 9;                         // Declara a variavel pinoServo e atribui o valor 9 a ela

int distancia;                             // Declara a variavel distancia sem atribuir valor a ela

int distanciaAcionamento = 5;              // Declara a variavel distanciaAcionamento e atribui o valor 4 a ela

Ultrasonic sensorUltrassonico(pinoTrigger, pinoEcho);        // Declara o objeto sensorUltrassonico para utilizar as funcoes da biblioteca Ultrassonic

Servo meuServo;                            // Declara o objeto meuServo para utilizar as funcoes de biblioteca Servo

//----------------Funcao executada uma vez na inicializacao do sistema----------------

void setup() {

  meuServo.attach(pinoServo);              // Inicializa o envio de comandos para o servo no pino 9 (valor da variavel pinoServo)
```

```
  meuServo.write(0);                          // Envia o comando para o servo mover para o angulo informado

}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------

void loop() {

  distancia = sensorUltrassonico.read();                  // Le o valor informado pelo sensor ultrassonico e atribui o valor lido a variavel distancia

  if (distancia <=  distanciaAcionamento){                 // Se a distancia for menor que a distancia anterior menos dois ou for maior que a distancia anterior mais dois, executa as instruções entre chaves

    delay(800);                               // Aguarda 800 milissegundos

    meuServo.write(120);                          // Envia o comando para o servo mover para o angulo informado

    do{                                 // Executa as instrucoes entre chaves uma vez e depois verifica a condição do while

     delay(200);                               // Aguarda 200 milissegundos

     distancia = sensorUltrassonico.read();              // Le o valor informado pelo sensor ultrassonico e atribui o valor lido a variavel distancia

    } while (distancia <= distanciaAcionamento);            // Checa se a condicao foi atendida, repete os comandos entre chaves do "do" acima ate que a condicao não seja mais verdadeira, entao segue com o codigo

    delay(3000);                              // Aguarda 3000 milissegundos (3 segundos)

    meuServo.write(0);                          // Envia o comando para o servo mover para o angulo informado

    delay(2000);                              // Aguarda 2000 milissegundos (2 segundos)

  }
```

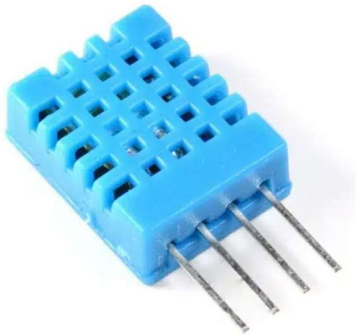
```
  delay(200);                    // Aguarda 200 milissegundos

}
```

13 – Sensor de temperatura e umidade

Módulo 13 – Sensor de Temperatura e Umidade – Confira mais um sensor para perceber o mundo ao redor do seu projeto e faça ainda mais, veja a temperatura e umidade do ambiente e refresque-se no calor.

13.1 – Sensor de temperatura e umidade

Você com certeza já ouviu na TV que a umidade relativa do ar está em x%. A umidade relativa se refere a quanto vapor de água está presente no ar frente a quanto teoricamente o ar naquela temperatura consegue segurar antes de começar a chover.



O Sensor de Umidade e Temperatura DHT11 opera com base em um sensor capacitivo para medir a umidade relativa do ar e um termistor para medir a temperatura ambiente. Ele possui uma membrana porosa que permite que a umidade do ar entre em contato com o sensor capacitivo, alterando sua capacitância de acordo com a umidade. O termistor registra a variação de temperatura e ambos os valores são convertidos em sinais digitais pelo sensor.

Esses sinais digitais são então lidos pelo microcontrolador (como um Arduino) por meio de um protocolo de comunicação serial. Assim, o microcontrolador pode interpretar os dados e fornecer

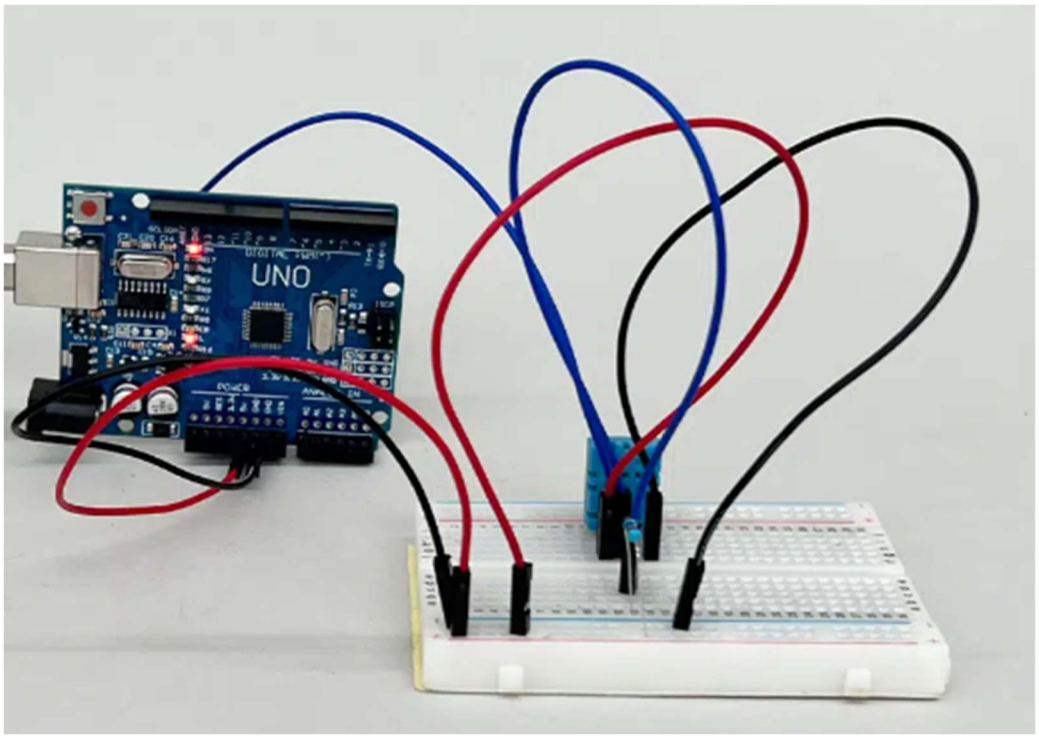
leituras de umidade e temperatura em tempo real. Este processo permite que o DHT11 forneça medições precisas e confiáveis da umidade e temperatura ambiente.

Pinagem do Sensor de Umidade e Temperatura DHT11

O sensor de umidade e temperatura DHT11 possui quatro pinos: VCC, DADOS, N.C e GND. O pino VCC é conectado à alimentação, o pino DADOS é usado para enviar os dados do sensor para o microcontrolador (geralmente um Arduino) e o pino GND é conectado à terra. O pino N.C não deve ser conectado.

13.2 – Leitura de temperatura e umidade

Outro projeto muito comum com arduino é monitorar a temperatura e umidade de um local, como um quarto, ou uma chocadeira, por exemplo. Vamos usar o sensor de temperatura e umidade DHT11 para entender como o arduino pode fazer esse monitoramento.

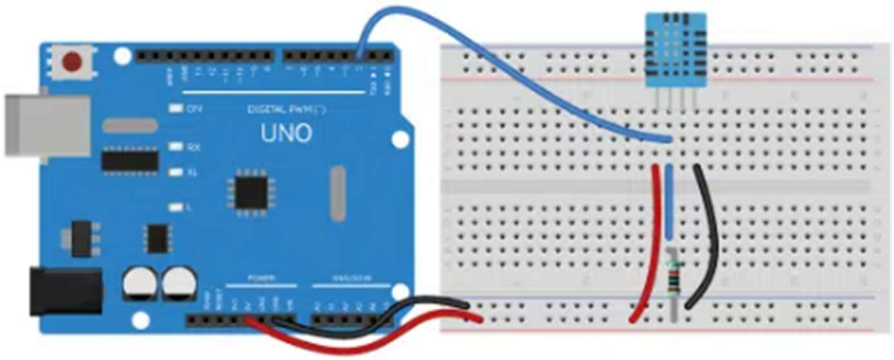


Material necessário

1x Sensor de temperatura e umidade DHT11
1x Resistor 10k Ohms
1x Protoboard 400 pontos
6x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

Para esse exercício vamos encaixar o sensor de temperatura DHT11 na protoboard conforme o esquema abaixo:



Programa

Novamente vamos usar uma biblioteca, então vá em **Ferramentas** -> **Gerenciador de Bibliotecas** e busque por **DHT sensor Library** (por Adafruit). Clique em instalar.

Você receberá uma mensagem avisando que a biblioteca possui dependências, que é quando uma biblioteca utiliza outras para o seu funcionamento. Muitas vezes há bibliotecas que fazem a comunicação base, o protocolo de comunicação e outras que fazem a comunicação com o sensor ou módulo. Clique em **instalar todas as dependências**.

Por ser um sensor mais complexo, o DHT às vezes pode apresentar erros de leitura. Para verificar isso e não tentar escrever valores incoerentes no monitor serial, usamos a função **isnan**(). Caso o sensor retorne algo que não é um número (representado por NaN na linguagem do arduino), essa função retornará um valor verdadeiro. Se for um número normal, retornará falso. Utilizamos isso para verificar se os dados recebidos são válidos

```
    if (isnan(event.temperature)) {                    // Se a temperatura nao for um numero, executa as
45                                                     instrucoes entre chaves
```

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Leitura de temperatura e umidade

// AUTOR: MAKERHERO

//----------------Inclui as bibliotecas do codigo----------------


#include <Adafruit_Sensor.h>                    // Inclui a biblioteca Adafruit_Sensor para facilitar o
controle do sensor de temperatura e umidade


#include <DHT.h>                                // Inclui a biblioteca DHT para facilitar o controle do sensor
de temperatura e umidade


#include <DHT_U.h>                              // Inclui a biblioteca DHT_U para facilitar o controle do
sensor de temperatura e umidade


//----------------Define as constantes que a biblioteca necessita

#define DHTTYPE    DHT11                        // Define DHTTYPE como DHT11


//----------------Declara as variaveis do codigo----------------


int pinoDHT = 2;                                // Declara a variável pinoDHT e atribui o valor 2 a ela

DHT_Unified dht(pinoDHT, DHTTYPE);              // Declara a variável dht para utilizar as funcoes
de biblioteca Servo

//----------------Funcao executada uma vez na inicializacao do sistema----------------
```

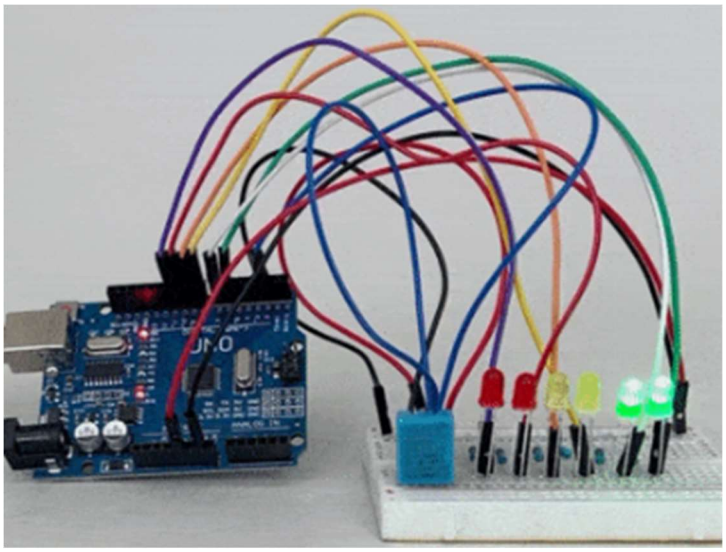
```
void setup() {
  Serial.begin(9600);                          // Inicializa a comunicação serial

  dht.begin();                                 // Inicializa o sensor de temperatura e umidade

}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------

void loop() {

  delay(1500);                                 // Aguarda 1500 milissegundos (1,5 segundos)

  sensors_event_t event;                       // Cria uma variável temporaria para a leitura das
mensagens enviadas pelo sensor

  dht.temperature().getEvent(&event);          // Faz a leitura da temperatura

  if (isnan(event.temperature)) {              // Se a temperatura nao for um numero, executa as
instrucoes entre chaves

    Serial.println(F("Erro lendo a temperatura"));     // Escreve o texto entre aspas no monitor serial
e passa para a próxima linha

  } else {                                     // Caso contrário (temperatura e um numero), executa as
instrucoes entre chaves

    Serial.print(F("Temperatura: "));          // Escreve o texto entre aspas no monitor serial
```

```
  Serial.print(event.temperature);                    // Escreve o valor de event.temperature no monitor serial

  Serial.println(F("°C"));                            // Escreve o texto entre aspas no monitor serial e passa para a próxima linha

 }

 dht.humidity().getEvent(&event);                     // Faz a leitura da umidade

 if (isnan(event.relative_humidity)) {                // Se a umidade nao for um numero, executa as instrucoes entre chaves

  Serial.println(F("Erro lendo a umidade"));          // Escreve o texto entre aspas no monitor serial e passa para a próxima linha

 } else {

  Serial.print(F("Umidade: "));                       // Escreve o texto entre aspas no monitor serial

  Serial.print(event.relative_humidity);              // Escreve o valor da umidade no monitor serial
  Serial.println(F("%"));                             // Escreve o texto entre aspas no monitor serial e passa para a próxima linha
 }

}
```

13.3 – Termômetro luminoso

Agora que sabemos utilizar o sensor de temperatura, que tal fazer um termômetro com uma escala luminosa?

Vamos usar 6 LEDs, 2 de cada cor, para fazermos um termômetro que apaga ou acende os LEDs conforme a temperatura ambiente.

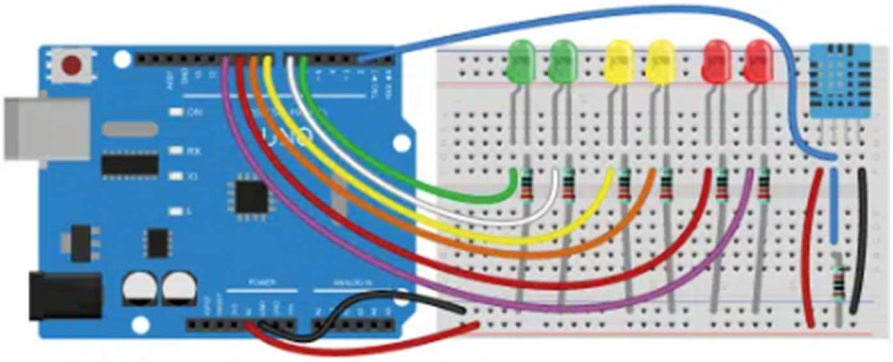


Material necessário

1x Sensor de temperatura e umidade DHT11
2x LED Amarelo
2x LED Verde
2x LED Vermelho
6x Resistor 220 Ohms
1x Resistor 10k Ohms
1x Protoboard 400 pontos
12x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

Para esse projeto vamos mover o DHT para a borda da protoboard para dar espaço para os LEDs. As demais ligações são como já vimos nos exercícios anteriores.



Programa

Para o programa vamos precisar verificar se a temperatura está dentro de algumas faixas de valores. Para isso vamos usar os condicionais **if**() e **else**, mas dessa vez iremos “aninha-los”, ou seja, vamos fazer uma sequência de **if**s e **else**s que serão analisados em sequência.

Abaixo você encontra o programa completo que pode ser copiado e colado direto na IDE Arduino:

```
// CODIGO: Termometro Luminoso

// AUTOR: MAKERHERO

//----------------Inclui as bibliotecas do codigo----------------


#include <Adafruit_Sensor.h>                    // Inclui a biblioteca Adafruit_Sensor para facilitar o controle do sensor de temperatura e umidade

#include <DHT.h>                                // Inclui a biblioteca DHT para facilitar o controle do sensor de temperatura e umidade

#include <DHT_U.h>                              // Inclui a biblioteca DHT_U para facilitar o controle do sensor de temperatura e umidade


//----------------Define as constantes que a biblioteca necessita


#define DHTTYPE    DHT11                        // Define DHTTYPE como DHT11
```

```
//----------------Declara as variaveis do codigo----------------

int pinoDHT = 2;                                    // Declara a variável pinoDHT e atribui o valor 2 a
ela

int pinoLedVerde1 = 6;                              // Declara a variável pinoLedVerde1 e atribui o
valor 6 a ela

int pinoLedVerde2 = 7;                              // Declara a variável pinoLedVerde2 e atribui o
valor 7 a ela

int pinoLedAmarelo1 = 8;                            // Declara a variável pinoLedAmarelo1 e
atribui o valor 8 a ela

int pinoLedAmarelo2 = 9;                            // Declara a variável pinoLedAmarelo2 e
atribui o valor 9 a ela

int pinoLedVermelho1 = 10;                          // Declara a variável pinoLedVermelho1 e
atribui o valor 10 a ela

int pinoLedVermelho2 = 11;                          // Declara a variável pinoLedVermelho2 e
atribui o valor 11 a ela

DHT_Unified dht(pinoDHT, DHTTYPE);                  // Declara a variável dht para utilizar
as funcoes de biblioteca Servo

//----------------Funcao executada uma vez na inicializacao do sistema----------------

void setup() {


 dht.begin();                                       // Inicializa o sensor de temperatura e umidade

 pinMode(pinoLedVerde1, OUTPUT);                    // Configura a porta 6 (valor da variavel
pinoLedVerde1) como saida

 pinMode(pinoLedVerde2, OUTPUT);                    // Configura a porta 7 (valor da variavel
pinoLedVerde2) como saida

 pinMode(pinoLedAmarelo1, OUTPUT);                  // Configura a porta 8 (valor da
variavel pinoLedAmarelo1) como saida

 pinMode(pinoLedAmarelo2, OUTPUT);                  // Configura a porta 9 (valor da
variavel pinoLedAmarelo2) como saida
```

```
  pinMode(pinoLedVermelho1, OUTPUT);                    // Configura a porta 10 (valor da
variavel pinoLedVermelho1) como saida
  pinMode(pinoLedVermelho2, OUTPUT);                    // Configura a porta 11 (valor da
variavel pinoLedVermelho2) como saida
}

//----------------Funcao executada repetidamente enquanto o sistema estiver ligado----------------

void loop() {
  delay(1500);                                  // Aguarda 1500 milissegundos (1,5 segundos)
  sensors_event_t event;                        // Cria uma variável temporaria para a leitura
das mensagens enviadas pelo sensor
  dht.temperature().getEvent(&event);           // Faz a leitura da temperatura
  if (isnan(event.temperature)) {               // Se a temperatura nao for um numero,
executa as instrucoes entre chaves

    Serial.println(F("Erro lendo a temperatura"));          // Escreve o texto entre aspas no
monitor serial e passa para a próxima linha

  } else {                                      // Caso contrário (temperatura e um numero), executa
as instrucoes entre chaves

    if (event.temperature <= 10){               // Se a temperatura for menor ou igual que
10, executa as instrucoes entre chaves
      digitalWrite(pinoLedVerde1, LOW);         // Apaga o primeiro LED Verde
      digitalWrite(pinoLedVerde2, LOW);         // Apaga o segundo LED Verde
      digitalWrite(pinoLedAmarelo1, LOW);       // Apaga o primeiro LED Amarelo
      digitalWrite(pinoLedAmarelo2, LOW);       // Apaga o segundo LED Amarelo
      digitalWrite(pinoLedVermelho1, LOW);      // Apaga o primeiro LED Vermelho
      digitalWrite(pinoLedVermelho2, LOW);      // Apaga o segundo LED Vermelho
```

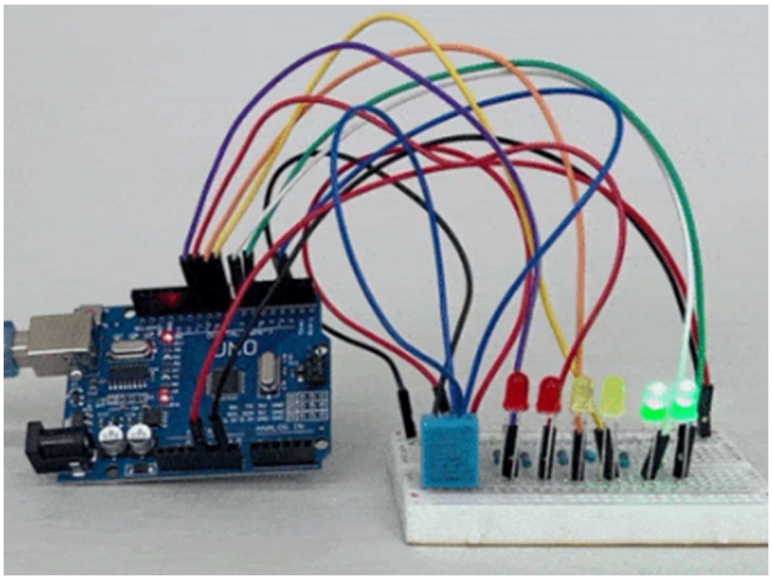
```
  } else if(11 <= event.temperature && event.temperature <= 20){          // Caso contrário, se a temperatura estiver entre 11 e 20 graus, executa as instruções entre chaves
    digitalWrite(pinoLedVerde1, HIGH);                    // Acende o primeiro LED Verde
    digitalWrite(pinoLedVerde2, LOW);                     // Apaga o segundo LED Verde
    digitalWrite(pinoLedAmarelo1, LOW);                   // Apaga o primeiro LED Amarelo

    digitalWrite(pinoLedAmarelo2, LOW);                   // Apaga o segundo LED Amarelo
    digitalWrite(pinoLedVermelho1, LOW);                  // Apaga o primeiro LED Vermelho
    digitalWrite(pinoLedVermelho2, LOW);                  // Apaga o segundo LED Vermelho

  } else if(21 <= event.temperature && event.temperature <= 25){          // Caso contrário, se a temperatura estiver entre 21 e 25 graus, executa as instruções entre chaves

    digitalWrite(pinoLedVerde1, HIGH);                    // Acende o primeiro LED Verde
    digitalWrite(pinoLedVerde2, HIGH);                    // Acende o segundo LED Verde
    digitalWrite(pinoLedAmarelo1, LOW);                   // Apaga o primeiro LED Amarelo
    digitalWrite(pinoLedAmarelo2, LOW);                   // Apaga o segundo LED Amarelo
    digitalWrite(pinoLedVermelho1, LOW);                  // Apaga o primeiro LED Vermelho
    digitalWrite(pinoLedVermelho2, LOW);                  // Apaga o segundo LED Vermelho

  } else if(26 <= event.temperature && event.temperature <= 30){          // Caso contrário, se a temperatura estiver entre 26 e 30 graus, executa as instruções entre chaves

    digitalWrite(pinoLedVerde1, HIGH);                    // Acende o primeiro LED Verde
    digitalWrite(pinoLedVerde2, HIGH);                    // Acende o segundo LED Verde

    digitalWrite(pinoLedAmarelo1, HIGH);                  // Acende o primeiro LED Amarelo
    digitalWrite(pinoLedAmarelo2, LOW);                   // Apaga o segundo LED Amarelo
    digitalWrite(pinoLedVermelho1, LOW);                  // Apaga o primeiro LED Vermelho
```

```
    digitalWrite(pinoLedVermelho2, LOW);                    // Apaga o segundo LED Vermelho

  } else if(31 <= event.temperature && event.temperature <= 35){          // Caso contrário, se a temperatura estiver entre 31 e 35 graus, executa as instruções entre chaves

    digitalWrite(pinoLedVerde1, HIGH);                    // Acende o primeiro LED Verde
    digitalWrite(pinoLedVerde2, HIGH);                    // Acende o segundo LED Verde
    digitalWrite(pinoLedAmarelo1, HIGH);                  // Acende o primeiro LED Amarelo
    digitalWrite(pinoLedAmarelo2, HIGH);                  // Acende o segundo LED Amarelo
    digitalWrite(pinoLedVermelho1, LOW);                  // Apaga o primeiro LED Vermelho
    digitalWrite(pinoLedVermelho2, LOW);                  // Apaga o segundo LED Vermelho
  } else if(36 <= event.temperature && event.temperature <= 40){          // Caso contrário, se a temperatura estiver entre 36 e 40 graus, executa as instruções entre chaves
    digitalWrite(pinoLedVerde1, HIGH);                    // Acende o primeiro LED Verde

    digitalWrite(pinoLedVerde2, HIGH);                    // Acende o segundo LED Verde
    digitalWrite(pinoLedAmarelo1, HIGH);                  // Acende o primeiro LED Amarelo
    digitalWrite(pinoLedAmarelo2, HIGH);                  // Acende o segundo LED Amarelo
    digitalWrite(pinoLedVermelho1, HIGH);                 // Acende o primeiro LED Vermelho
    digitalWrite(pinoLedVermelho2, LOW);                  // Apaga o segundo LED Vermelho
  } else if(41 <= event.temperature ){                    // Caso contrário, se a temperatura estiver maior que 41, executa as instruções entre chaves
    digitalWrite(pinoLedVerde1, HIGH);                    // Acende o primeiro LED Verde
    digitalWrite(pinoLedVerde2, HIGH);                    // Acende o segundo LED Verde
    digitalWrite(pinoLedAmarelo1, HIGH);                  // Acende o primeiro LED Amarelo
    digitalWrite(pinoLedAmarelo2, HIGH);                  // Acende o segundo LED Amarelo
    digitalWrite(pinoLedVermelho1, HIGH);                 // Acende o primeiro LED Vermelho
    digitalWrite(pinoLedVermelho2, HIGH);                 // Acende o segundo LED Vermelho
  }
```

```
    }
}
```

13.4 – Higrômetro luminoso

Agora vamos usar a mesma lógica do exercício anterior, mas agora para a medição de umidade ao invés da de temperatura.

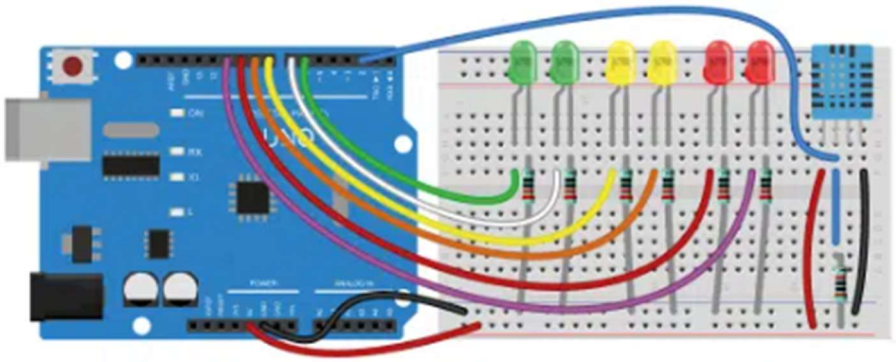


Material necessário

1x Sensor de temperatura e umidade DHT11
2x LED Amarelo
2x LED Verde
2x LED Vermelho
6x Resistor 220 Ohms
1x Resistor 10k Ohms
1x Protoboard 400 pontos
12x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

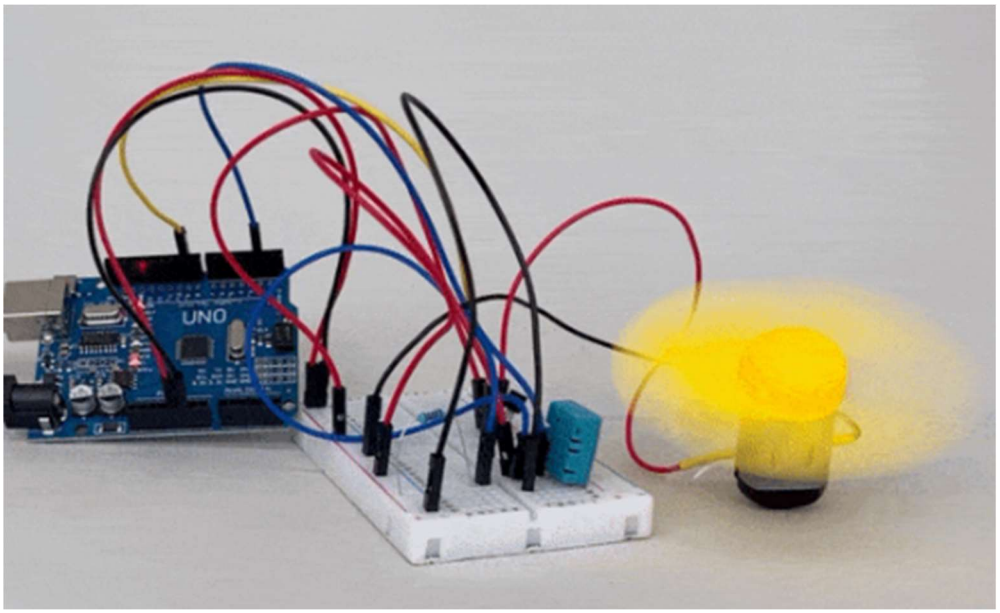
Montagem do circuito

Vamos utilizar a mesma montagem do projeto anterior.



13.5 – Ventilador automático

Agora que conseguimos medir a temperatura, que tal agir com base nela e ligar um ventilador quando estiver muito quente? Neste último exercício vamos usar o motor DC com a Hélice para nos refrescarmos se a temperatura passar de um determinado valor.



Material necessário

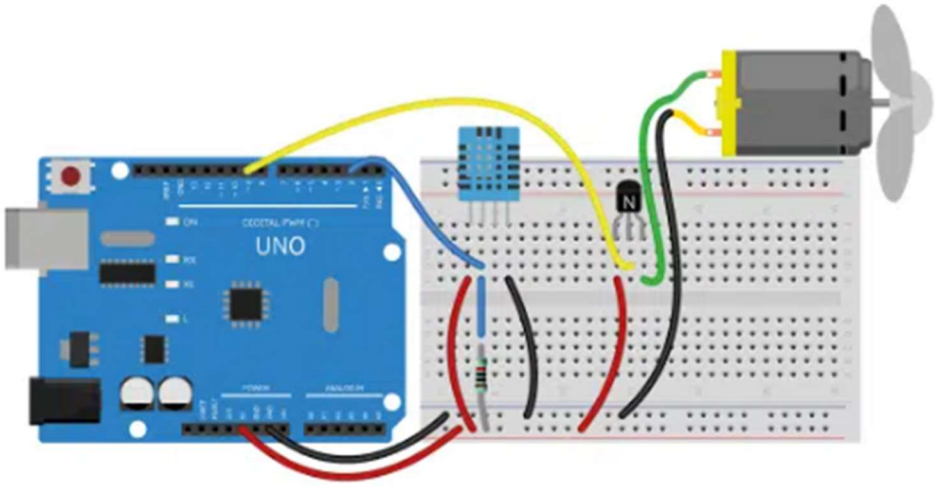
1x Sensor de temperatura e umidade DHT11
1x Motor DC
1x Transistor BC338

1x Resistor 10k Ohms
1x Protoboard 400 pontos
6x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

Nesse projeto vamos retirar os LEDs e seus respectivos resistores e incluir o motor DC

Se preferir, você pode deixar o sensor de temperatura na mesma posição do projeto anterior.



15 – Projetos Extras

Módulo 15- Projetos Extras- projetos que combinam conhecimentos apresentados no curso em projetos mais complexos e que abordam temáticas de ciências, física, matemática e outras matérias do universo Steam

Está sabendo tudo de arduino? Então agora vamos explorar uma combinação de conhecimentos do curso e aplicar em projetos com temáticas diferentes e que ao combinar componentes e funções te ajudam ainda mais a se desafiar e consolidar os seus conhecimentos de Arduino, programação e eletrônica.

15.1 – Controle Chocadeira

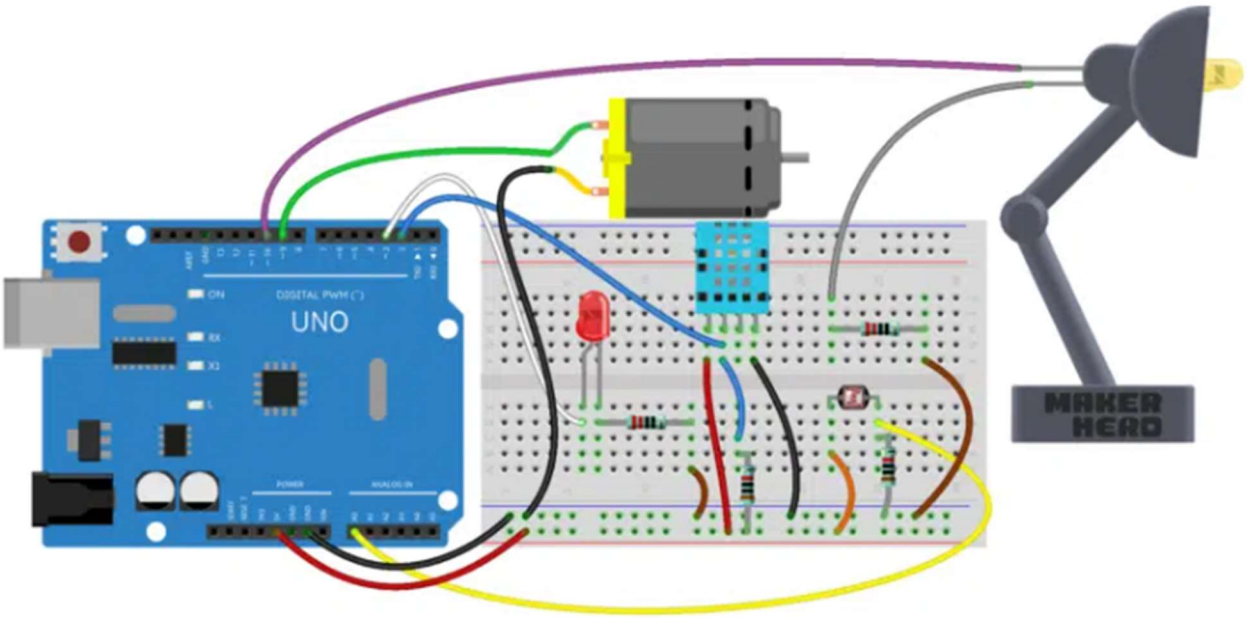
E que tal controlar uma chocadeira que monitora a temperatura e luz do ambiente e para manter a condição ideal para os ovinhos nascerem?

Material necessário

1x Sensor de temperatura e umidade DHT11
1x Motor DC
1x Hélice
1x Sensor de Luminosidade LDR
1x LED Vermelho
1x LED Amarelo
1x Luminária 3D
2x Resistor 10k Ohms
2x Resistor 220 Ohms
1x Protoboard 400 pontos
20x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

A montagem desse projeto irá seguir o esquema abaixo:



15.2 – Jogo Genius

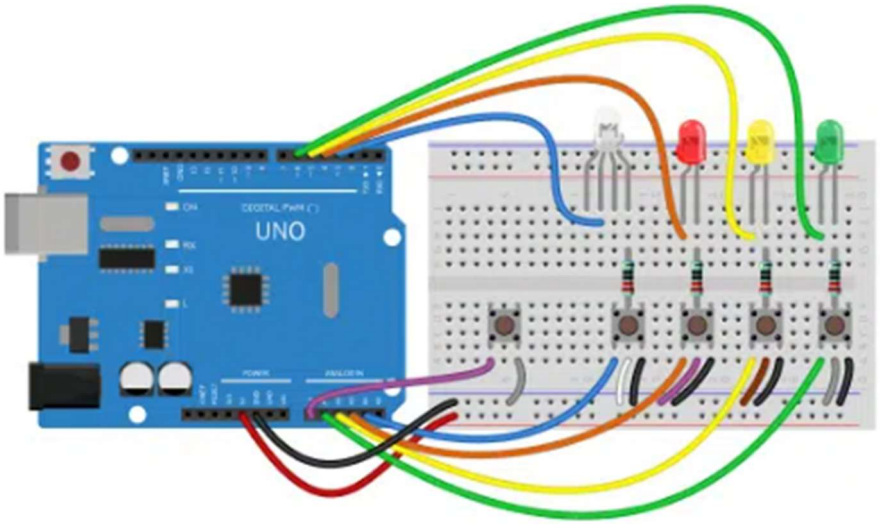
Já ouviu falar do jogo Genius? Ele foi lançado originalmente em 1978 e pode ser replicado com os componentes do seu kit maker. Vamos lá?

Material necessário

1x LED RGB
1x LED Vermelho
1x LED Amarelo
1x LED Verde
4x Resistor 220 Ohms
5x Chave push-button
1x Protoboard 400 pontos
20x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

A montagem desse projeto irá seguir o esquema abaixo:



15.3 – Jogo Tiro ao Alvo

Agora que você já está craque, vamos fazer um jogo de tiro ao alvo com o arduino?

Vamos fixar o diodo laser no servo para dificultar um pouco as coisas, e vamos tentar acertar o sensor de luminosidade para ir baixando o contador de vidas do inimigo.

Mas cuidado, se você demorar demais, ele irá recuperar a vida.

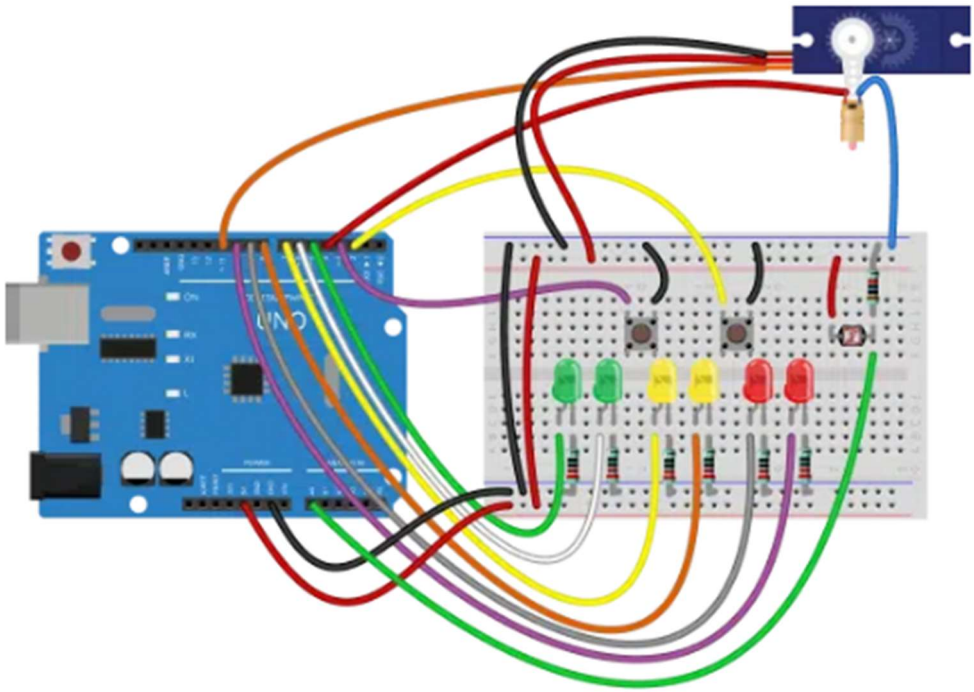
Vamos lá?

Material necessário

1x Diodo Laser
1x Servo motor
1x Resistor 10k Ohms
6x Resistor 220 Ohms
1x Chave push-button
1x Protoboard 400 pontos
18x Jumper Macho-macho
1x Cabo USB
1x Placa Uno

Montagem do circuito

A montagem desse projeto irá seguir o esquema abaixo:



Dica! Você pode utilizar fita para fixar o diodo laser na haste do servo motor. Para alongar o fio do diodo laser você pode utilizar dois jumper macho-fêmea, conectando o fio do diodo no conector fêmea.

Programa

Nesse projeto vamos usar algumas funções um pouco mais avançadas que não vimos nos exercícios e projetos anteriores, como a função millis() e interrupções.

Como são tópicos mais avançados, não iremos entrar em detalhes. A ideia desses projetos é mostrar o quanto é possível fazer com o Arduino e alguns componentes e ajudar a liberar a sua criatividade.