introdução à eletrônica para artistas

helder da rocha

Termos de uso

http://www.eletronicaparaartistas.com.br

R672i Rocha, Helder Lima Santos da, 1968-
Introdução à Eletrônica para Artistas.
__pp. 21cm x 29.7cm. PDF.
Documento criado em 18 de maio de 2017.

1. Eletrônica aplicada (621.381) – Eletrônica (621.38) – Física aplicada (621)
2. Projeto de circuitos eletrônicos – Eletrônica (537.5) – Física (530).
3. Aplicações artísticas da eletrônica – Artes em geral (707).
   I. Título.
   II. Apostila de curso livre.

CDD 621.381

CONTEÚDO

1. Introdução

Esta apostila é utilizada como referência e fonte de aprofundamento teórico em vários tópicos que são explorados de forma prática na oficina prática *Introdução a Eletrônica para Artistas*. Mas o a ordem e o conteúdo deste material poderá ser diferente dependendo da carga-horária e forma de realização da oficina. Nem todos os assuntos e experimentos abordados nesta apostila serão explorados durante a oficina.

1.1. Oficina: Introdução à Eletrônica para Artistas

Durante os próximos encontros iremos explorar a arte e ciência de controlar a eletricidade usando eletricidade – **a eletrônica** – um dos conhecimentos mais importantes da humanidade e que teve influência incontestável na cultura do século 20, sem a qual não existiria o rádio, a TV, as viagens espaciais, os computadores, a Internet. A eletrônica está em toda parte e faz parte da natureza da nossa civilização. Ao conhecer seus princípios você terá os recursos para usar a eletrônica como matéria-prima em suas obras.

A oficina é um curso introdutório projetado para artistas. Durante os encontros nós iremos ligar e desligar coisas, mover motores, criar e detectar som, luz e calor. Vamos também fazer medições, algumas contas e também queimar alguns circuitos. O foco não é a eficiência, mas a investigação de usos criativos dos princípios da eletrônica. Para isto precisaremos conhecer algumas leis, princípios, regras, saber fazer medições, estimativas, cálculos simples (mesmo que o objetivo seja posteriormente quebrar as regras.) O curso é uma oportunidade de experimentação e também de troca com outros artistas. Ao longo do curso e no último dia você terá a oportunidade de usar a eletrônica em seus projetos, e em colaboração com os outros participantes.

1.1.1. Estrutura da oficina

O curso acontecerá durante cinco ou dez encontros e se divide em cinco partes. Os tópicos abaixo descrevem o que esperamos abordar em cada parte, mas o programa pode variar um pouco se necessário. Os próximos cinco capítulos desta apostila refletem essa organização.

Parte 1 – Introdução – Fundamentos de Eletrônica, Resistores, LEDs e Capacitores

Nesta primeira parte faremos uma introdução do curso, apresentar os artistas participantes, conhecer um pouco da história e alguns exemplos de aplicações criativas da eletrônica. Apresentaremos também o kit e o laboratório. Em seguida iniciaremos com alguns experimentos: medições de tensão, corrente e resistência, identificação de resistores, como usar o protoboard, como identificar capacitores, e como calcular circuitos com LEDs usando a Lei de Ohm. Construiremos uma bateria com batatas e acenderemos LEDs, ligaremos um motor e uma mini cigarra.

Parte 2 – Transistores

Nesta parte exploraremos circuitos mais complexos usando o transistor – o componente semicondutor de silício que foi o protagonista da revolução eletrônica nos anos 50. Usaremos o protoboard para construir circuitos com transistores e sensores de luz e som, acionando LEDs e relés com presença e ausência de luz, magnetismo e efeitos sonoros. Faremos também um oscilador que irá piscar LEDs alternadamente e soar um apito em um alto-falante. Poderemos explorar também alguns experimentos com eletromagnetismo.

Parte 3 – Circutos integrados

Esta parte será dedicada à aprendizagem do circuito integrado 555, que custa menos que 1 real e permite fazer circuitos temporizadores, acionar motores, piscar LEDs, apitar, disparar cronômetros, acender luzes em sequência e displays de LED. A temporização é a base de qualquer circuito digital, do mais simples ao mais complexo. Conhecer o 555 é essencial para saber fazer circuitos simples e baratos, e entender melhor o funcionamento de circuitos com resistores e capacitores. Usaremos capacitores para calcular a frequência e o tempo das temporizações.

Parte 4 – Introdução ao Arduino

Nesta penúltima parte conheceremos o Arduino, um dos mais populares microcontroladores. Você não precisa mais calcular circuitos de temporização com capacitores: pode escrever um programa simples no computador dizendo quando o LED vai apagar e acender, e fazer muito, muito mais. O Arduino requer a programação em computador, mas isto é muito mais simples que calcular circuitos. Faremos uma série de experimentos que introduzem o essencial do Arduino usando o Arduino Nano, incluído no kit. Se também quiser programar, traga seu computador.

Parte 5 – Projeto

Esta última parte do curso é livre para que você possa desenvolver e apresentar um projeto usando eletrônica. Descubra uma forma de aplicar a eletrônica na sua arte, junte-se a outros colegas e apresente algo no final. Pode ser algo bem simples, por exemplo, fazer um LED piscar na sua bolsa do curso, um alarme acionado por luz ou um carrinho que se move com um grito ou quando a luz acende (esses são projetos que você conseguirá fazer ao final do curso). Dependendo de como a oficina for organizada, esta parte poderá ocorrer ao longo dos vários dias do curso.

1.1.2. Sobre o laboratório e o material

Esta é uma oficina prática. Todos terão uma bolsa com um kit contendo diversos e variados componentes eletrônicos: resistores, capacitores, transistores, circuitos integrados, sensores de luz e calor, leds, chaves e interruptores, fios, etc. O kit também inclui uma barra de prototipagem (para montagem dos circuitos), um multímetro, uma fonte de 9V, um alicate de bico-fino e uma placa-clone Arduino Nano. A maioria dos componentes serão usados e demonstrados durante a oficina. Outros podem ser usados em seus projetos durante ou depois da oficina.

Uma lista com todos os materiais contidos no kit está em uma seção própria no final da apostila. O laboratório em sala de aula também dispõe de materiais e ferramentas compartilhadas, e inclui:

- ferro de soldar com suporte
- solda, pasta de solda, sugador de solda e malha de cobre
- suporte para placa de circuito impresso e terceira mão com lupa
- alicates de corte, de ponta-fina e para desencapar fios
- estiletes e tesouras
- fio esmaltado, fios flexíveis e sólidos
- fita isolante, espaguete termo-retrátil, epóxi, super-bonder, cola quente e silicone
- lupas e calculadoras
- multímetros, protoboards, placas e componentes extras

1.1.3. Sobre as baterias e fonte incluídas no kit

Incluímos 3 tipos de fonte de energia: 1 bateria CR2032, 2 baterias AAA de 1,5V e uma fonte de 9V (que será usada para alimentar a maior parte dos circuitos). A fonte de 9V tem a capacidade de fornecer corrente para circuitos de até 1 ampere. Ela pode ser usado no lugar de uma bateria de 9V, e também para alimentar circuitos com Arduino Uno ou Nano.

Em vez de usar a fonte de 9V você talvez ache mais fácil trabalhar com uma bateria de 9V (que pode não estar incluída no kit). Uma vantagem da bateria é não precisar de tomada. A vantagem da fonte é que ela não descarrega e fornece corrente e tensão constante.

Outra vantagem da bateria, para iniciantes em eletrônica, é que ela perde carga rapidamente quando há um curto-circuito (assim você pode ganhar algum tempo para desligar o circuito antes de queimar alguma coisa.) O ideal é usar uma bateria alcalina. Uma bateria recarregável (Ni-Mh) também pode ser usada. Mas não use baterias de lítio (Li-ion ou Li-po). Elas esquentam muito e podem explodir em caso de curto-circuito. Baterias baratas de zinco-carbono também podem ser usadas.

1.1.4. Infraestrutura para o módulo de Arduino

Na quarta parte da oficina demonstraremos o uso de Arduino, e se você trouxer seu computador podemos tentar instalar o ambiente (IDE) do Arduino nele. Mesmo que você não leve um notebook, você ainda poderá criar circuitos com o Arduino (teremos um computador disponível para programa-lo). Para evitar dificuldades no acesso WiFi (que pode não estar disponível ou ser limitada em alguns

locais de realização da oficina), você pode baixar os drivers e ambiente de desenvolvimento antes da aula (veja detalhes no capítulo sobre Arduino).

1.1.5. Experimentos extras

Alguns experimentos presentes na apostila poderão ser apenas *demonstrados* durante a oficina (não haverá tempo para que todos possam construí-los juntos durante a aula), mas você pode montá-los fora do horário de aula. Eles também podem ser montados por alunos avançados que terminarem os outros experimentos antes.

Experimentos marcados como "extra" não serão realizados durante esta oficina, mas alunos avançados podem tentar fazê-los fora do horário de aula, discuti-los durante o curso e usá-los em seus projetos.

Todos os experimentos, inclusive os extras, usam apenas materiais disponíveis no kit.

1.2. Sobre esta apostila

1.2.1. Seções de referência

Além dos módulos que refletem a estrutura da oficina *Introdução à Eletrônica para Artistas*, esta apostila também contém várias seções de referência.

Técnicas

Esta seção contém alguns tutoriais sobre técnicas úteis para construir projetos com eletrônica. Como soldar, como costurar circuitos para vestir, como criar circuitos de papel, onde encontrar materiais como cola condutiva, fitas condutivas e como fazer massa condutiva.

Links e referências

Esta seção contém uma lista dos livros, artigos, tutoriais e outras referências bibliográficas usadas para escrever esta apostila. Também contém links para vários outros recursos, como simuladores de circuitos, software e calculadoras online.

Material usado nos experimentos

Uma lista de todos os materiais, ferramentas e componentes distribuídos no kit da oficina *Introdução à Eletrônica para Artistas*.

Referência rápida dos componentes

Uma lista ilustrada de todos os principais componentes do kit, com foto do componente, símbolo usado nos esquemas, exemplos de uso, especificações de limites de tensão, corrente e potência, links para a especificação (*datasheet*) e informações adicionais sobre forma de uso.

Mini-referência de Arduino

Uma pequena referência rápida contendo os comandos e estruturas da linguagem do Arduino que exploramos neste curso, e informações técnicas sobre especificações do Arduino Nano.

1.2.2. Circuitos

A maior parte dos circuitos usados nesta apostila foram criados especificamente para a oficina. Alguns foram adaptados de circuitos publicados em livros, revistas e sites, revisados e testados. Os circuitos não têm restrição de uso e podem ser usados em outros projetos e adaptados livremente.

1.2.3. Código-fonte

Os programas escritos para Arduino estão disponíveis na internet em um repositório GitHub. Você pode baixá-los em https://github.com/helderdarocha/EletronicaParaArtistas. Eles são software livre e também podem ser usados e alterados livremente.

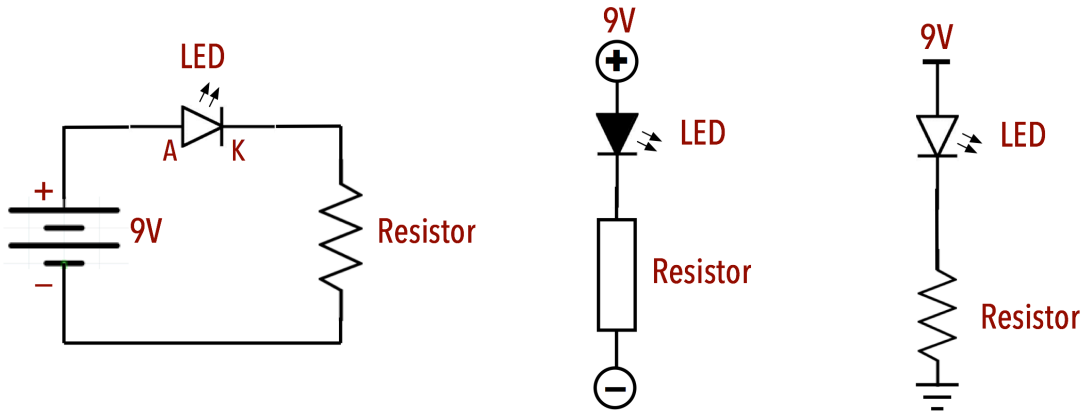
2. Fundamentos de Eletrônica

2.1. Circuitos

Para usar eletricidade para realizar algo é preciso construir um **circuito**. Um circuito interliga os polos de uma **fonte** de potencial elétrico (**tensão**) permitindo o fluxo de **corrente**. Circuitos elétricos podem ser construídos de várias formas: interligando fios metálicos, usando uma placa de prototipagem, usando uma placa de circuito impresso, usando caminhos desenhados com tinta condutiva, alinhavados com linha de costura condutiva, até espremidos com massa condutiva.

Um circuito é representado graficamente através de um **esquema**. O esquema é um diagrama que ilustra de forma objetiva os caminhos por onde passa a corrente e os componentes que estão interligados. É como usar um mapa do metrô, que foca nas conexões e omite os detalhes desnecessários. Um mesmo esquema pode representar um circuito montado em uma placa ou costurado em uma roupa. Os componentes eletrônicos são representados por símbolos mais ou menos padronizados. Também é comum indicar do lado de cada símbolo valores ou outras informações necessárias.

Existem várias formas de desenhar esquemas. Os três diagramas abaixo representam esquemas idênticos para ligar uma lâmpada LED a uma fonte de 9 volts.



Observação: o símbolo ⏚, é o símbolo de **terra**, ou *ground*, às vezes abreviado **GND**. Em circuitos de alta tensão ele realmente indica um cabo ligado à terra, que funciona como referência zero de tensão. Mas esse símbolo também é usado em circuitos de baixa tensão para representar o **terminal negativo** (que deve ser conectado ao negativo da bateria, não à terra).

É muito importante aprender a ler esquemas eletrônicos. Para montar um circuito é preciso transcrever o esquema para o meio usado na montagem (uma base de protótipos, placa de circuito impresso, fios interligados, linha condutiva costurada em um tecido, etc.)

2.2. Fontes de energia

Para funcionar, um circuito necessita de uma fonte de energia elétrica. Todos os circuitos que iremos construir nesta oficina são circuitos de **energia contínua** que podem ser fornecidos por uma bateria. O símbolo abaixo, usado em esquemas de eletrônica, representa uma bateria.

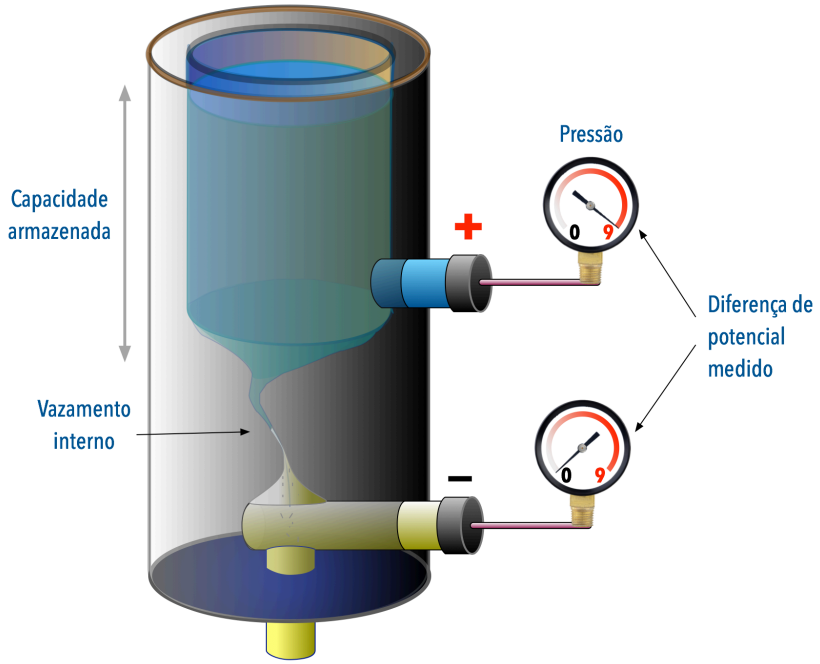
Uma **bateria** é uma fonte **química** de potencial elétrico. Ela possui um valor nominal de **potencial elétrico** (indicado em volts) e permite o **fluxo** de uma quantidade máxima de **corrente** por unidade de tempo (indicado em mAh – miliamperes por hora). Para limitar a corrente é preciso que o circuito atravesse componentes que forneçam **resistência** à corrente. Esses componentes diminuem o fluxo de corrente transformando-a em outro tipo de energia (ex: luz, movimento, calor). Potencial elétrico é **tensão**, fluxo de carga elétrica é **corrente**. Essas duas propriedades, mais a **resistência** ao fluxo de elétrons do material por onde flui a corrente, serão os valores mais importantes que precisamos aprender a observar, calcular e medir ao projetar circuitos eletrônicos.

2.2.1. Tensão

1 volt (indicado pelo símbolo V) é a unidade padrão de potencial elétrico, que também é chamado de **tensão**, ou **voltagem**. O nome é uma homenagem ao cientista italiano Alessandro Volta, o inventor da pilha elétrica. Baterias geram potencial elétrico (**tensão**) através de uma **reação química**.

Outra forma comum de gerar eletricidade envolve **magnetismo**. A eletricidade fornecida pela fonte de 9 volts distribuída no kit é obtida da rede elétrica que obtém sua energia de geradores que transformam energia mecânica em elétrica. A fonte **retifica** essa energia (converte energia alternada em contínua), e **reduz** a tensão de 120V para 9V, para que possa ser usada no lugar da bateria.

O funcionamento de uma bateria ou fonte de tensão contínua quanto ao seu potencial elétrico pode ser comparado ao potencial energético um **tanque cheio de água**. A pressão da água pode ser considerada análoga à tensão. Enquanto a saída da água estiver fechada, não haverá fluxo, mas a pressão (potencial energético) existe. Um encanamento com a torneira fechada, que não deixa fluir água, é análogo a um **circuito aberto**, por onde não flui corrente elétrica.



Pressão e tensão são medidas **relativas**, e devem ser medidas **em relação a um valor de referência** (pressão do ar, potencial da terra, potencial do polo negativo, etc). Em uma bateria, usamos convencionalmente como referência o polo negativo. A tensão é a diferença de potencial entre polos. Portanto, se medimos a tensão no polo negativo, o valor é zero, mas se medirmos a tensão no polo positivo (ou seja, entre o polo positivo e o negativo), o valor é o da capacidade da bateria. Esta analogia também pode ser representada pela caixa d'água acima.

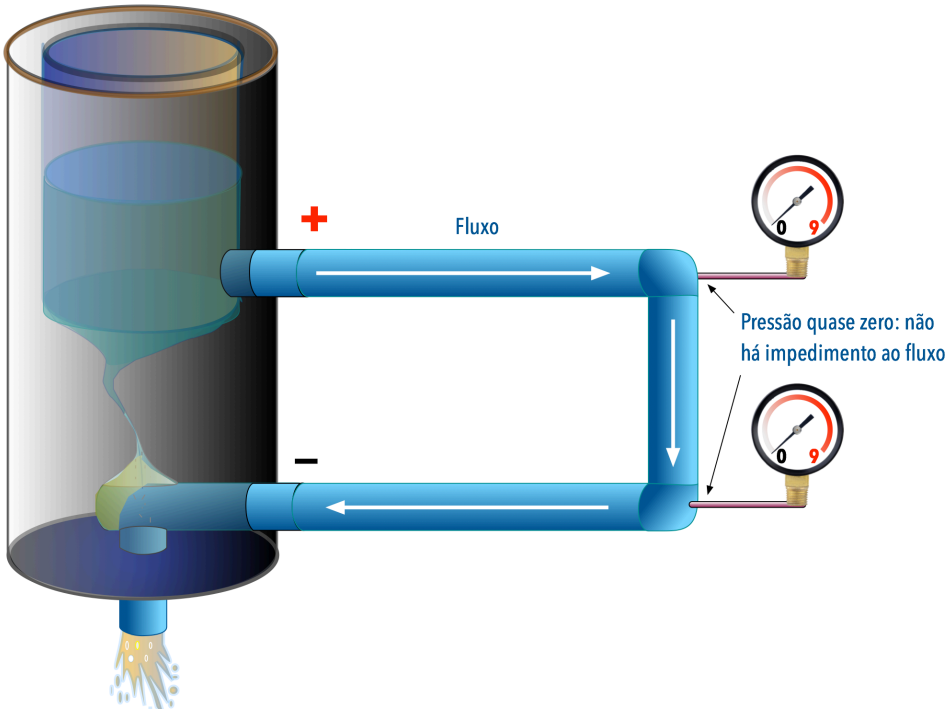
Não existe uma bateria 100% eficiente. Qualquer bateria que permanecer armazenada por muito tempo sem uso, também irá descarregar, pois pequenas quantidades de corrente **vazam** entre os polos da bateria. Na analogia acima, representamos o vazamento interno por uma passagem estreita, por onde a água vaza lentamente.

2.2.2. Corrente

Corrente elétrica é a **quantidade de elétrons** que flui por um condutor a cada segundo. É a corrente que efetivamente faz o circuito funcionar. **1 ampere** (indicada pelo símbolo A) é a unidade padrão de **corrente elétrica**. O nome é uma homenagem ao cientista francês **Jean-Marie Ampère**.

Pode-se usar a analogia do fluxo de água também para entender o fluxo de corrente elétrica. Não é uma analogia perfeita, mas ajuda a entender o mecanismo básico da corrente elétrica.

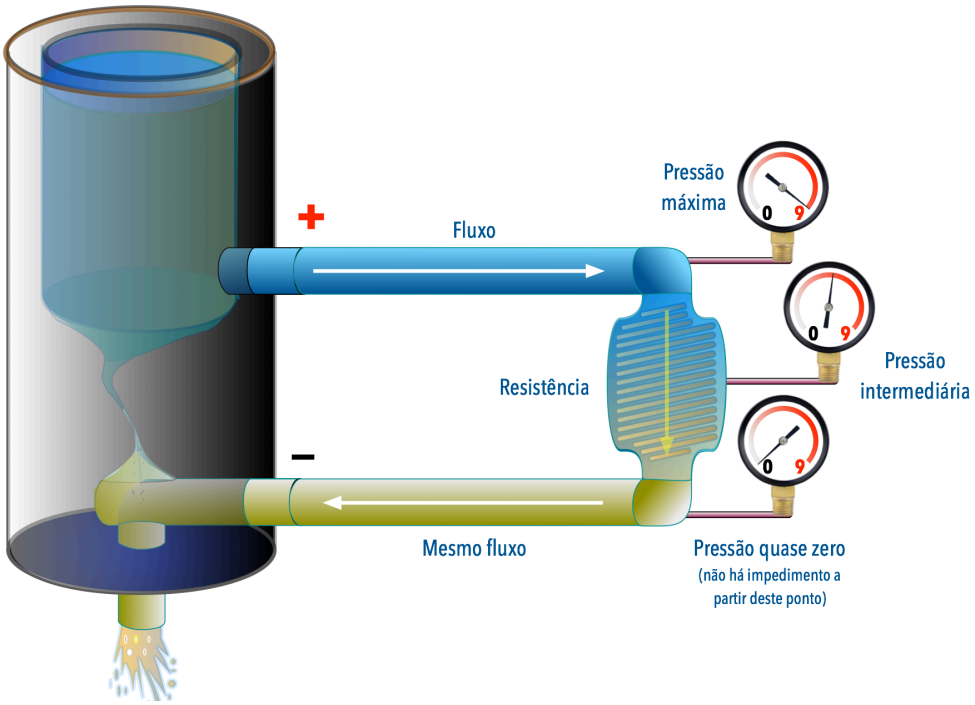
Considere a ilustração abaixo. Na hora que a passagem da caixa d´água é aberta, a água sai pelo cano "+" e passa por um caminho que retorna pelo cano "–" (fazendo analogia aos polos da bateria.) Se nada impedir a passagem da água, a pressão cairá drasticamente, e em pouco tempo a caixa d´água estará vazia. Na realidade, o cano irá oferecer alguma resistência, já que é uma passagem estreita com atrito e veremos a pressão diminuir pressão lentamente ao longo do caminho. Mas vamos supor um cano ideal, quase sem atrito. Se medirmos a pressão entre dois pontos próximos do cano, teremos um valor tão baixo que será percebido pelo medidor como **zero**:



A vazão da água pode ser comparada à corrente em um circuito. Fazendo uma analogia com o desenho acima, isto seria um **curto-circuito** - onde o positivo da bateria é ligado diretamente no negativo. O curto-circuito é um fenômeno destrutivo: um valor máximo de corrente flui entre os polos, fazendo a tensão cair drasticamente e a bateria esquentar muito. Em pouco tempo toda a carga da bateria é esgotada (se ela não explodir ou se o fio não derreter e se romper antes).

Circuitos elétricos projetados para serem úteis buscam controlar o fluxo de corrente através de **cargas resistivas**. Uma carga pode ser uma lâmpada, um motor, ou qualquer dispositivo que consuma corrente transformando-a em algum outro tipo de energia (movimento, luz, calor). Uma carga pode também ser um circuito complexo, com vários caminhos por onde a corrente se divide.

Na analogia da caixa d'água, a corrente pode ser comparada à medida da quantidade de água que flui pelo cano em um determinado intervalo de tempo, considerando que os canos estão sempre cheios de água e que não haja vazamentos no encanamento. Essa medida **será igual em qualquer ponto**, mesmo que haja algum tipo de mecanismo retardando o fluxo da água. Se o fluxo diminuir de velocidade, ele diminuirá tanto no início, onde há mais pressão, quanto no final, onde há menos. Esse mecanismo é análogo a uma carga resistiva em um circuito elétrico. Esse tipo de carga terá, no ponto em que for ligada ao encanamento, uma pressão próxima à pressão da caixa d´água, mas no final, quando encontrar novamente o encanamento, uma pressão próxima de **zero**, pois nada mais impede a passagem da água até o polo negativo, apenas a resistência do próprio cano. Se medirmos a pressão no meio da carga, ela deve estar perto da **metade** da pressão fornecida.



Um circuito elétrico funciona de forma similar. Se apenas uma carga estiver conectada a uma bateria, a tensão medida (diferença de potencial) entre os terminais da carga será **igual à tensão da bateria**. Se a carga estiver dividida em duas partes iguais (ex: duas lâmpadas ligadas em série), e pudermos medir a tensão no meio dela, encontraremos aproximadamente **metade da tensão** nesse ponto. Por fim, a tensão medida entre o terminal que está ligado ao negativo e o polo negativo da bateria, deverá ser (praticamente) **zero**. Já a corrente será igual em todo o circuito.

Portanto, embora relacionadas, as propriedades tensão e corrente podem aumentar e diminuir de forma independente, já que dependem da resistência. Em circuitos alimentados por baterias **com a mesma carga resistiva**, mais tensão garante mais corrente, mas se a resistência diminuir, mesmo a tensão permanecendo constante, a corrente irá aumentar. A tensão será (normalmente) limitada pela capacidade de fornecimento da bateria. Já a corrente será geralmente limitada pela resistência da carga. Mesmo com uma tensão baixa você pode ter correntes muito altas, que podem destruir um circuito. Iremos explorar bastante essas propriedades nos próximos experimentos.

Experimento 1 – Medição de tensão de uma bateria

Tudo ficará mais fácil de entender com alguns experimentos. Neste primeiro experimento vamos aprender a usar o multímetro e tentar medir o potencial elétrico (a tensão) de várias e diferentes fontes de energia elétrica: duas ou três baterias e uma fonte chaveada de 9V ligada na rede elétrica.

Material necessário:

- Multímetro
- Baterias diversas
- Fonte de 9V ou 12V com saída em plugue P4 macho
- Plugue/tomada P4 fêmea
- Fios/jumpers

Use o **multímetro** distribuído no kit. Um multímetro é um medidor multi-função. O modelo usado (830B) possui um display numérico e um seletor de função dividido em várias partes, além de funções para medir tensão (voltímetro), corrente (amperímetro) e resistência (ohmímetro). Inclui também funções para testar continuidade e ganho de transistores. Neste experimento usaremos apenas a função **voltímetro**, que mede tensão. O voltímetro possui uma resistência interna muito alta. Quando conectado a uma fonte de tensão, ele deixa passar uma quantidade mínima de corrente (apenas o suficiente para permitir a medição da tensão da fonte ou bateria). Na analogia da caixa d'água, é como se o voltímetro fizesse um pequeno furo na caixa, só para deixar passar uma pequena quantidade de água suficiente para medir a pressão.

Para medir tensão, o **cabo preto** deverá ser colocado na **tomada preta** (a de baixo, marcada **COM**), e o **cabo vermelho** na **tomada vermelha do meio**.

Gire o seletor para a seção de 5 posições marcada como **"V="**, que serve para medir **tensão contínua**, que é o tipo de tensão produzida por baterias. (As duas posições marcadas **"V~"** servem para medir tensão alternada, como a produzida pela rede elétrica.)

2000 mV (2 V) | 20 V (Note reading is negative) | 200 V

(jbriant.eu)

Cada posição do seletor é identificada pelo **valor máximo** que pode ser medido. A posição **2000 mV** mede qualquer valor até 2V (2 mil milivolts é a mesma coisa que 2 volts). Se o valor for maior, o multímetro indicará no display um **"1___"** à esquerda do visor. Se o voltímetro estiver na posição 200V, medirá até 200 volts (de tensão contínua), mas não terá precisão suficiente para medir valores pequenos de tensão. Escolha a posição mais adequada girando o seletor dentro da faixa **"V="**. O ideal é iniciar com um valor maior que a tensão a ser medida, e ir baixando até obter uma leitura que tenha uma precisão razoável.

Meça as tensões das baterias e fonte contidas no kit. Para medir, encoste as pontas de prova nos terminais das baterias.

Você pode tocar os terminais e segurar com a mão se necessário. Não existe risco de choque, já que as tensões e correntes usadas na fonte e baterias são muito baixas. Mas não faça isto quando for medir tensões altas com o multímetro. Neste curso não mediremos nem faremos circuitos usando alta-tensão.

O kit contém uma **mini-bateria CR2032** de **3V**, duas **pilhas AAA** de **1,5V** e uma fonte de **9V**, que deve ser ligada na tomada. Experimente medir a tensão das pilhas AAA separadamente, e também em série, posicionando duas delas enfileiradas (positivo com negativo).

Para medir a tensão da fonte, use um pino P4-fêmea (foto ao lado, à esquerda) que encaixa no pino P4-macho da fonte, e encoste as pontas de prova nos terminais do lado oposto.

Você pode também tentar obter a tensão da **célula fotovoltáica de silício** incluída no kit. Ela é de vidro, finíssima e muito frágil, portanto manuseie com cuidado. Use a escala de 2000 mV e coloque uma ponta de prova de cada lado, sobre a trilha branca. O valor máximo da célula fica em torno de **500 mV** (milivolts) quando exposto à luz direta do sol.

Experimento 2 – Ligando LEDs, cigarras e motores com 1,5 e 3V

Neste segundo experimento usaremos a mini bateria de 3V para acender vários LEDs, ligar um motor e uma cigarra.

Material necessário:

- 1 bateria CR2032 de 3V
- 1 pilha AAA de 1,5V
- 1 motor de 3V
- LEDs diversos
- Cigarra de 5V

Escolha alguns **LEDs** e tente acendê-los na bateria CR2032 de 3V, segurando um terminal de cada lado da bateria. Se eles não acenderem em uma posição, inverta os terminais.

LEDs não são lâmpadas comuns. São componentes eletroluminescentes polarizados que emitem luz em um espectro que inclui infravermelho, luz visível e ultravioleta. São muito sensíveis e não podem receber grandes tensões diretamente. A maior parte dos LEDs são alimentados por tensões entre 1,8 e 3,5V, e baixas correntes (no máximo 0,02 A). Nenhum LED do kit acende com menos de 1,8V. Alguns requerem até 3V para começar a emitir qualquer luz.

Não tente ligar LEDs diretamente em baterias de 9V ou na tensão da fonte. Eles irão queimar rapidamente. Mas você pode alimentar LEDs por alguns minutos (ou mais, dependendo do LED) usando a mini bateria de 3V. LEDs possuem uma polaridade, portanto só irão acender se o **terminal menor**, o catodo (**K**), estiver ligado ao **negativo** da bateria, e o terminal **maior**, o anodo (**A**), ao **positivo**. Outra forma de identificar o catodo é procurar o lado que tem um chanfro na base circular do LED. Se o LED for transparente e você conseguir ver o interior dele, o catodo é o terminal que está ligado ao componente maior.

Ligar um LED ao contrário com 3V não irá queimá-los, mas ele não vai acender. LEDs queimam com tensões reversas maiores que 5V.

O **motor** incluído no kit tem um valor nominal de operação de 3V, mas funciona com tensões entre 1 e 6V, portanto você vai conseguir fazê-lo girar ligando à bateria de 3V, ou até mesmo com a bateria de 1,5V. Não ligue o motor diretamente na bateria de 9V. Ele irá rodar mais rápido, mas não vai durar muito tempo. Diferente do LED, o motor funciona com qualquer polaridade. A única diferença é que ele gira na direção oposta se a polaridade for trocada.

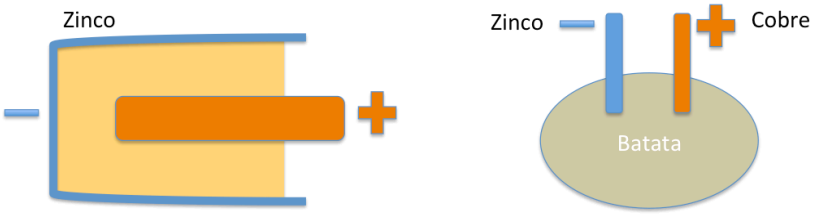
A **cigarra** também tem uma polaridade. Ligue o terminal maior no positivo da bateria e o outro no negativo e você ouvirá um apito agudo. Ela também deve ter um "+" impresso na sua embalagem indicando o terminal positivo. A cigarra também deve funcionar com 1,5V.

Experimento 3 – Bateria de cobre/zinco com eletrólito de batata

Material necessário:

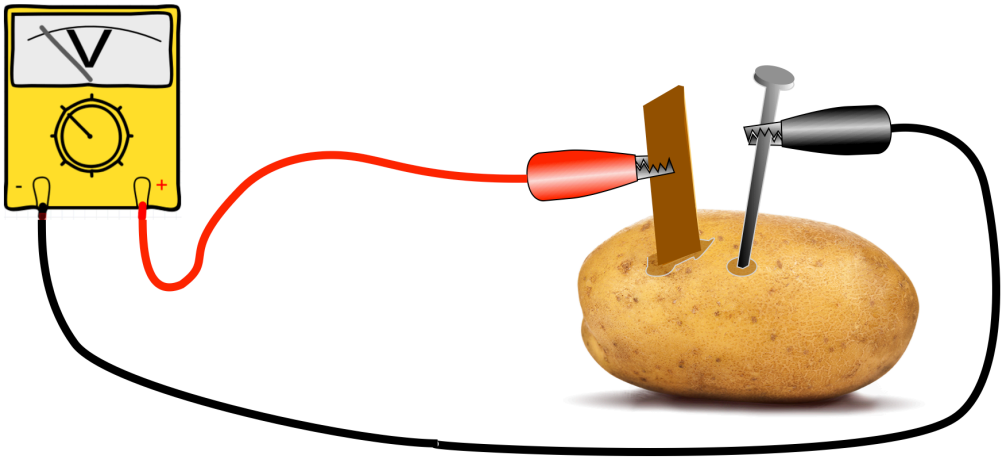
- 1 batata (ou banana ou limão)
- 1 pedaço de cobre (ex: fio de cobre sólido) com 3 a 5 cm de comprimento
- 1 pedaço de zinco (ex: prego galvanizado) com 3 a 5 cm de comprimento
- Cabos com garras jacaré
- Multímetro, na função Voltímetro
- LED

A **reação química entre os materiais zinco e cobre** permite produzir, em condições ideais, baterias com até **1,1** volts de potencial elétrico. A bateria é produzida preenchendo o espaço entre os terminais de zinco e cobre com um **ácido**. Este ácido pode ser um purê de batatas, um suco de limão, ou mesmo uma batata, limão, kiwi, ou outra fruta ácida.

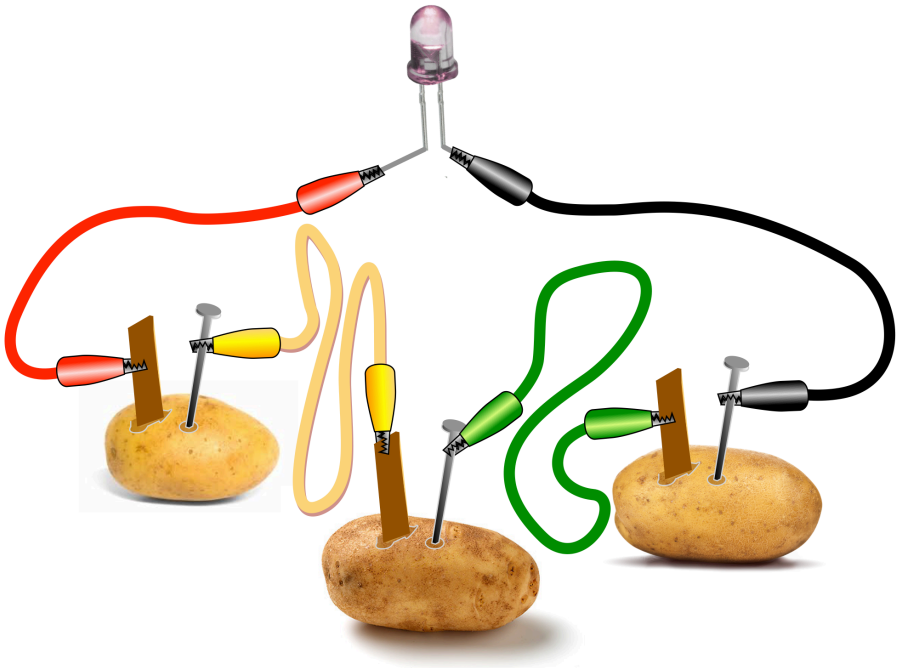


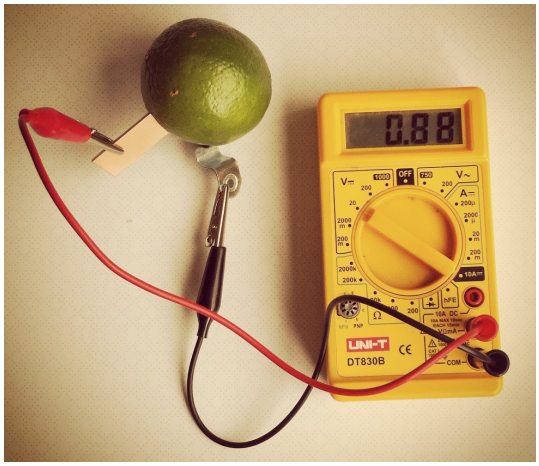
Para os terminais da bateria, use um pedaço de zinco (pode ser um prego ou parafuso de zinco galvanizado, desde que não plastificado), e um pedaço de cobre (pode ser um fio rígido grosso). Cada terminal deve ter aproximadamente 5cm. Insira na batata até onde for possível, **mas sem permitir que os terminais toquem um no outro dentro da batata**.

Depois, use o voltímetro para medir a tensão produzida (coloque na posição 2000 mV). O valor medido deve ser de pouco menos de 1 volt. Observe a polaridade. Se o valor medido for negativo, ela está invertida (o cabo vermelho indica a o positivo). A polaridade será importante para ligar o LED. O terminal positivo da batata é o **catodo** de cobre. O terminal negativo é o **anodo** de zinco.



Para acender um LED é necessário produzir **pelo menos 2 a 3V** (e uma corrente de pelo menos 5mA – 0,005 A). Como uma única batata fornece corrente insuficiente (2 mili amperes – 2mA – no máximo), será necessário obter mais baterias. Junte sua batata com a de seus colegas usando garras jacaré e ligando-as **em série** (cobre com zinco, zinco com cobre) e veja quantas batatas são necessárias para acender um LED. Mais batatas em série aumentam a tensão produzida, e também a capacidade de fornecer corrente.





Baterias de batata não são muito práticas, mas conseguem fornecer energia por um tempo razoável. Quando ela começar a perder carga, você pode limpar a oxidação dos terminais e reinseri-los. A bateria na verdade não é de batata, mas de **zinco-cobre**. O eletrólito pode ser outro, como por exemplo, suco de limão, laranja, banana, etc. A combinação zinco-cobre sempre produz em torno de **1V** por bateria. A foto ao lado ilustra uma bateria similar feita com limão.

As baterias que usamos hoje contém eletrólitos secos de material **ácido** ou **alcalino**. As mais comuns fornecem entre 1,2 a 1,6V. Baterias de valores maiores (ex: 4,5V, 9V, 12V) são obtidas conectando várias células em série.

A relação abaixo descreve algumas das baterias mais comuns usadas atualmente:

- **Zinco-Carbono** (ZnC) – baterias comuns com eletrólito **ácido**, que produzem **1,5V** cada. Baterias de 9V de ZnC contém 6 células de **1,5V**.
- **Zinco-Óxido de Manganês** (ZnMnO2) – baterias alcalinas modernas não-recarregáveis. Tem alta capacidade de fornecimento de corrente desde que descarreguem lentamente. Fornecem **1,5V**.
- **Níquel-Cádmio** (NiCd) – baterias recarregáveis com eletrólito **alcalino**, produzindo **1,25V**.
- **Níquel-Hidreto metálico** (NiMH) – baterias recarregáveis populares. Geralmente fornecem entre **1,25V** por bateria.
- **Lítio-polímero** (Lipo) e **Lítio-íon** (Li-ion) – mais eficientes – usadas em mini e microbaterias, baterias de celular, etc. Produzem **1,2V** e geralmente são distribuídas em pacotes de 3, fornecendo **3,7V**. Estas baterias tem alta capacidade de fornecimento de corrente. Um curto-circuito em uma delas pode causar incêndios e explosões.

A capacidade de fornecimento de corrente é medida em **mAh** (miliampere por hora) e varia dependendo da bateria e da velocidade da descarga. Sabendo-se o valor de mAh de uma bateria e o nível de consumo de corrente de um circuito, pode-se estimar o tempo que um circuito funcionará antes que a bateria não possa mais alimentá-lo. Isto é importante para avaliar o custo de um circuito.

A lista abaixo contém valores típicos de capacidade em mAh para baterias populares, considerando o uso da bateria em condições ideais para a sua estrutura química, e baixo consumo de corrente:

- **Bateria de 9V**. Alcalina: 565, ZnC: 400, Li-ion: 1200, NiMH: 175-300, NiCd: 120, Li-po: 500
- Pilhas botão de lítio: **CR2032**: 225. **SR41**: 25. **SR44**: 110
- **AAA**. Alcalina: 1200, ZnC: 540, NiMH: 800-1000, NiZn: 500
- **AA**. Alcalina: 2700, ZnC: 1100, Li-FeS2: 3000, NiMH: 1700-2700, NiCd: 600-1000, NiZn: 1500

Números maiores nem sempre significam baterias melhores. O comportamento de uma bateria varia bastante dependendo da sua composição química. Baterias de NiMH e lítio são muito mais eficientes em situações de consumo elevado. Já alcalinas e baterias de zinco-carbono perdem a carga rapidamente quando a demanda de corrente é alta (mas podem funcionar por muito tempo se a demanda for baixa.) Os valores nominais das baterias (informado na embalagem) também variam conforme o tipo. Alcalinas geralmente informam a tensão máxima fornecida, enquanto que baterias de lítio e recarregáveis informam uma tensão média, que é menor que a tensão medida quando a bateria está com carga máxima. Portanto, calcular o consumo usando mAh é mais preciso em circuitos que usam baterias recarregáveis ou de lítio.

2.3. Condução de eletricidade

Para transmitir eletricidade para outras partes do circuito, precisamos construir um "encanamento" feito de **condutores** elétricos. Condutores são materiais capazes de conduzir corrente elétrica com um valor desprezível de resistência. Exemplos de condutores são fios, os cabos com garras jacaré, ou os terminais metálicos de um LED.

Igualmente importantes na construção de circuitos são os materiais **isolantes**. Eles impedem (ou dificultam muito) a passagem de corrente. O ar é um isolante, assim como vários materiais sólidos como plástico e borracha.

O fluxo da corrente é convencionalmente representado do positivo para o negativo da bateria. Ligar um condutor diretamente entre os dois polos da bateria provoca um **curto-circuito**, liberando um grande fluxo de corrente em pouco tempo (o máximo que a bateria pode fornecer), o que fará com que ela descarregue rapidamente, provavelmente esquente muito ou até mesmo exploda. Para fazer um circuito útil e consumir a bateria de uma forma sustentável, precisamos que o caminho entre o positivo e negativo ofereça alguma **resistência** à passagem de corrente. Por exemplo, um LED ou um motor já oferecem muito mais resistência que um fio, e limitam a corrente, convertendo parte dela em luz ou movimento, e permitindo a descarga lenta da bateria.

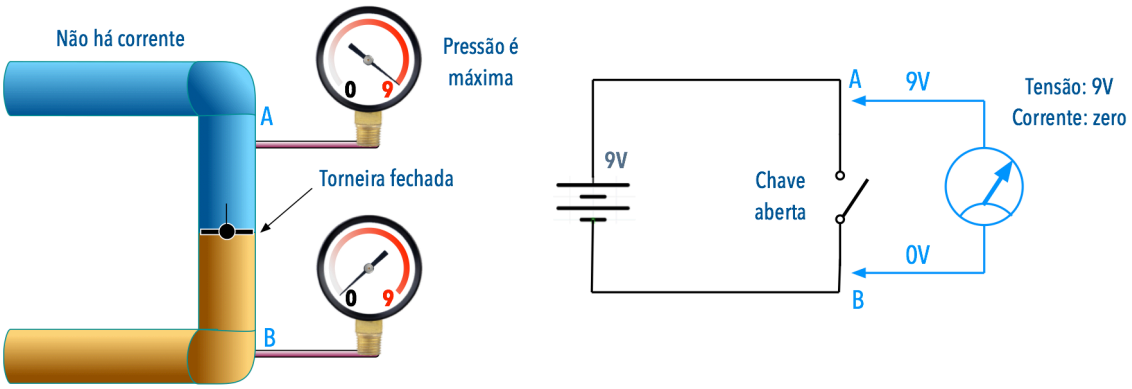
Os melhores condutores geralmente são metais como cobre, alumínio e prata. Em circuitos práticos é comum a preferência por fios de cobre e caminhos de cobre desenhados em placas de circuito impresso, soldados com prata ou estanho. Em circuitos artísticos, aproveitar a resistência natural dos materiais pode ser algo desejável. Podemos trocar os fios de cobre por linha de costura de aço inoxidável, fita adesiva de prata e cobre, tecidos condutivos, massa condutiva, papel laminado, cartolina laminada, tinta condutiva, cola condutiva, pó de ferro, líquido condutivo, e outros condutores que não são tão eficientes quanto os fios de cobre e prata, mas que também possibilitam a criação de circuitos eletrônicos.

As fotos abaixo ilustram linha de costura condutiva (aço inoxidável), fita de cobre condutiva, fita de tecido de prata condutiva e tecido condutivo.

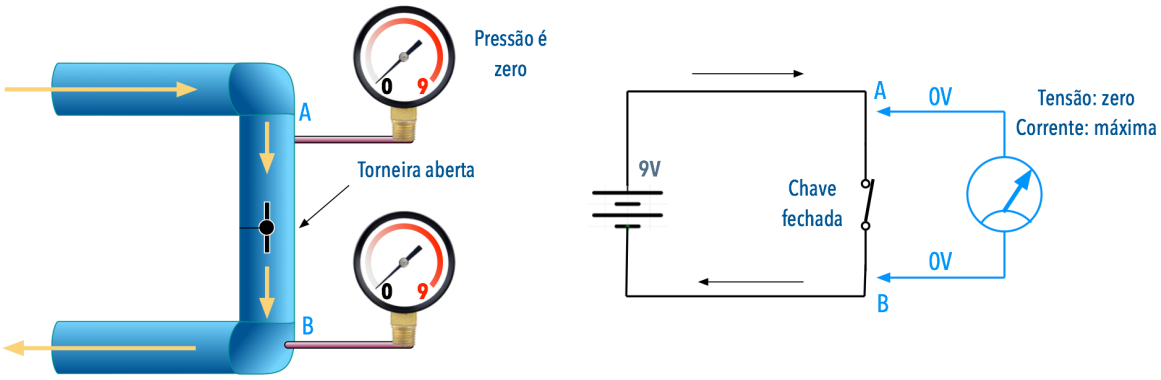



2.3.1. Circuitos abertos e fechados

Por um **circuito aberto** não passa corrente. Se ele estiver ligado a uma fonte **potencial** de energia, o valor dela pode ser medido e irá se concentrar entre os dois pontos onde o circuito está aberto. O efeito é análogo a medição da pressão concentrada em uma torneira fechada.



Uma vez **fechado** o circuito, esse ponto terá potencial quase zero, pois a corrente irá fluir sem impedimentos (convencionalmente) do positivo ao negativo da fonte de tensão. É análogo à água que flui por um encanamento sem o impedimento da torneira fechada.



(O circuito acima é ilustrativo. Na prática, todo material tem alguma resistência, o que limita a corrente e garante uma queda de tensão mínima diferente de zero.)

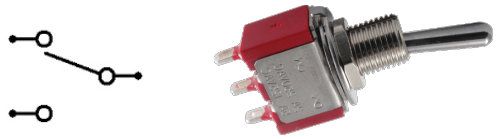
Circuitos complexos consistem de vários caminhos por onde a corrente se bifurca e possuem trechos que são abertos e fechados temporariamente, mudando e desviando o fluxo de corrente e a distribuição de potencial pelo circuito durante sua operação. O simples evento de abrir e fechar rapidamente um circuito gera **pulsos** de corrente que servem para disparar eventos em outras partes do circuito. O controle de comportamentos desse tipo é o objetivo da eletrônica.

2.3.2. Chaves

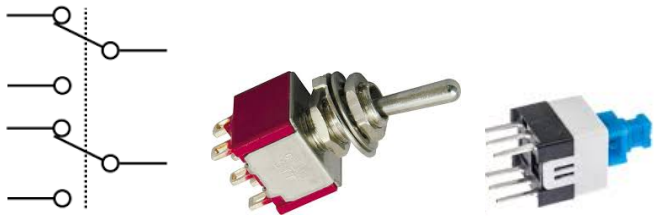
Um circuito pode ser aberto temporariamente através de um **interruptor**. Um interruptor pode ser uma **chave liga-desliga** simples, mas pode ser também uma **chave magnética** (um *reed*, ou um *relé*), uma **chave eletrônica** (um *transistor*, uma *porta lógica*) ou qualquer tipo de mecanismo que interrompa o fluxo de corrente que atravessa um condutor (um *zíper*, um *botão de roupa*, um encaixe metálico, um fio passando através de uma porta que rompe quando alguém passa).

Interruptores comerciais típicos têm **dois terminais** que são ligados ao circuito. Podem ser interruptores de pressão, **normalmente abertos**, que fecham um circuito apenas quando apertados, ou interruptores que estacionam em uma das duas posições (ligado ou desligado). Os símbolos abaixo são usados para representar esses interruptores em circuitos:

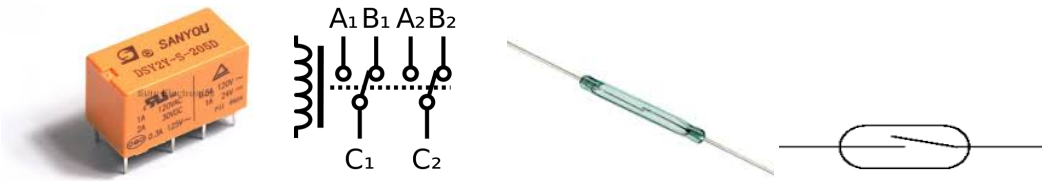
Chaves que possuem **três terminais** podem também ser usadas como interruptores se apenas dois dos seus terminais forem usados. Se os três terminais fizerem parte do circuito elas servem para **desviar o fluxo da corrente**. Essas chaves sempre têm **duas posições inicialmente ligadas entre si**. Quando mudam de posição a configuração se inverte: abrem um trecho do circuito, mas fecham outro.



Há também chaves de **dois ou mais polos**, que podem chavear vários circuitos **independentes** de uma só vez. No kit há duas chaves dessas do tipo alavanca, e uma do tipo pressão.



No kit há também foi incluído **um relé**, que é uma chave acionada **eletricamente** (o relê do kit tem dois polos e duas posições), e **um reed**, que é uma chave acionada **magneticamente** (interruptor *normalmente aberto* de um polo e uma posição).



Chaveamento eletrônico é feito através de transistores (que exploraremos mais adiante) que funciona como uma **porta lógica**, ligando ou desligando a partir de correntes e tensões aplicadas a um terminal de controle, e **circuitos integrados** (que internamente são compostos de centenas a bilhões de portas lógicas, abrindo e fechando centenas a bilhões de circuitos por segundo).

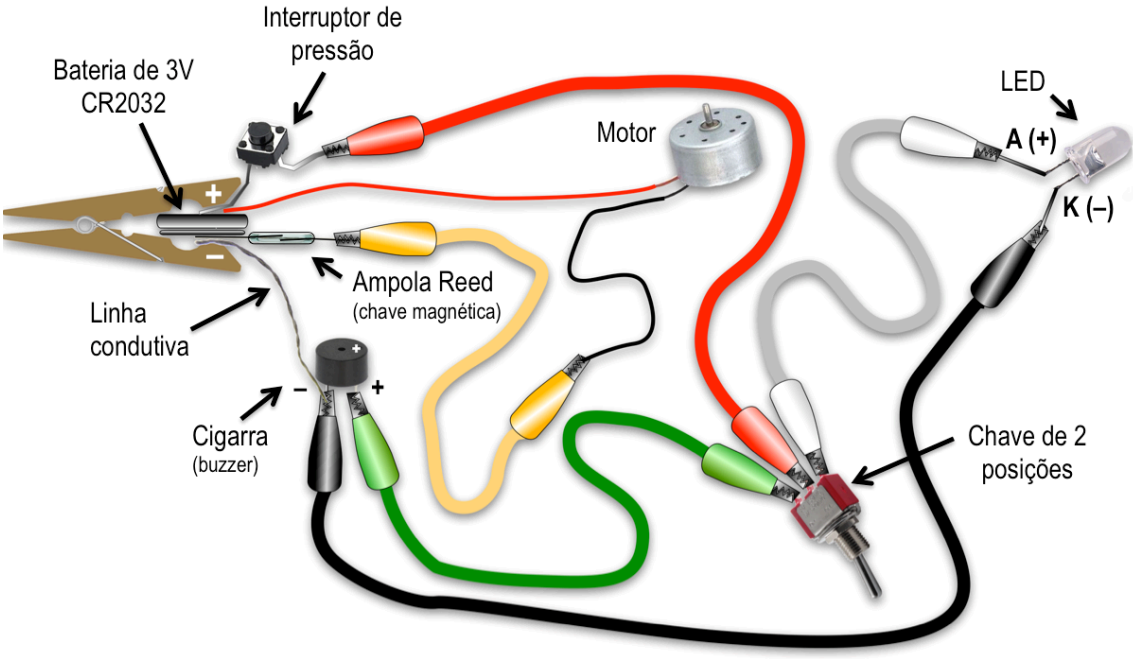
Experimento 4 – Um circuito usando chaves

Usando fios, fitas, linhas, garras jacaré, chaves e alguns componentes eletrônicos, faremos a corrente de uma bateria fluir por diversos caminhos ligando e desligando esses componentes.

Material necessário (veja a referência no final da apostila se não souber identificar o componente):

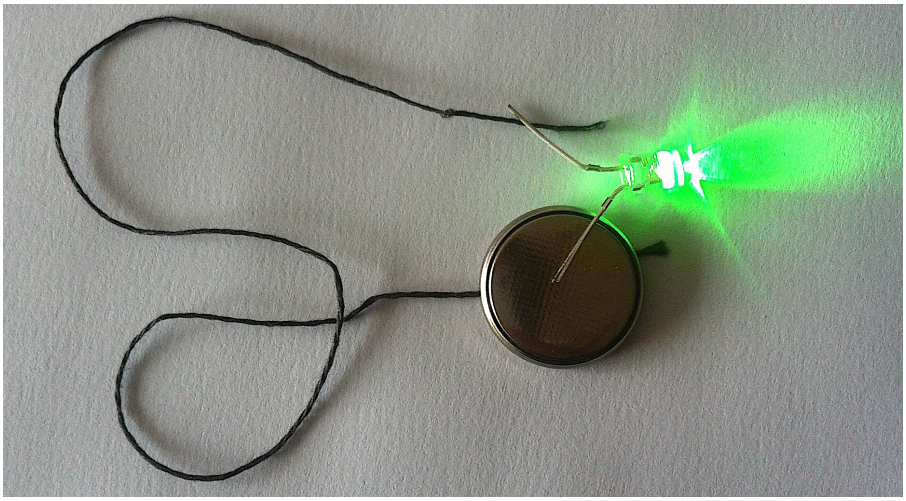
- Multímetro
- Cabos com garras jacaré
- Linha de costura condutiva
- Chave táctil de pressão
- Chave magnética reed
- Chave de duas posições
- Imã pequeno (ferrite ou neodímio)
- Um LED de qualquer cor
- Motor de 3V
- Cigarra
- Bateria de 3V
- Pegador de roupa (para servir de suporte para a bateria).

Construa o circuito abaixo. Faça as conexões usando garras jacaré, pegadores de roupa e nós. Tenha cuidado para manter os fios suficiente afastados ou isolados para que não interfiram no circuito, principalmente a linha condutiva e as garras jacaré conectadas nas chaves, que ficam muito juntas. Verifique bem as conexões para evitar maus contatos.



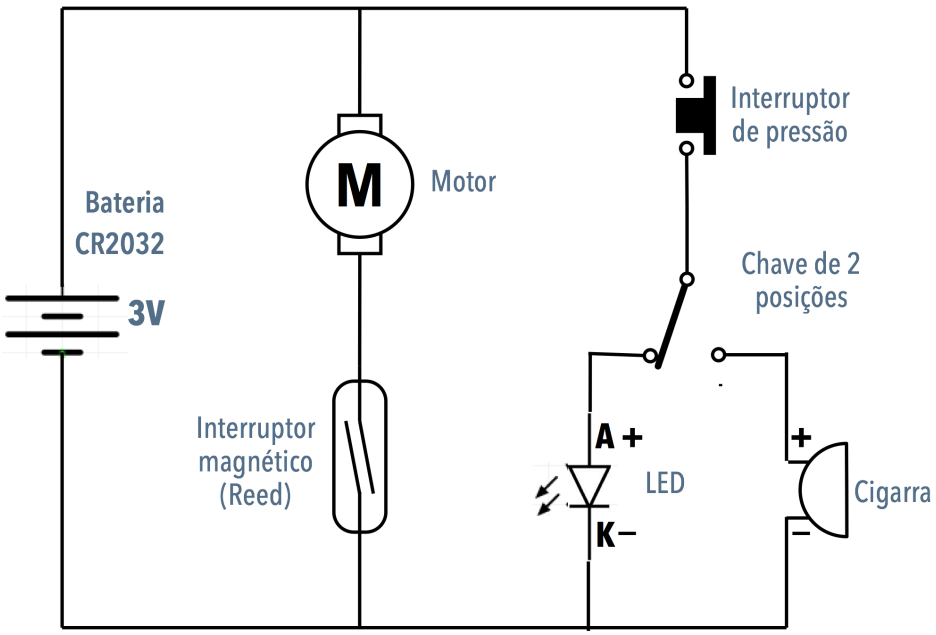
Confira a polaridade do LED e da cigarra, pois eles não funcionam se ligados ao contrário. A perna maior do LED é seu anodo (A) e deve ser ligada ao positivo. A cigarra tem uma indicação na embalagem indicando qual dos terminais é positivo. Os outros componentes não têm polaridade e podem ser conectados em qualquer posição.

A linha condutiva não é um fio elétrico comum. Ela não é isolada e é preciso ter cuidado para que ela não toque em outras partes do circuito, pois pode causar um curto-circuito e descarregar a bateria. **Você pode trocá-la por um jumper ou pedaço de fio se quiser.** Ela serve para costurar caminhos condutivos em tecido. Você pode testar a condutividade dela com um LED:



À medida em que for conectando cada componente, risque a conexão no desenho do circuito acima para que fique mais fácil lembrar o que já foi feito. Tenha cuidado ao manusear o reed pois sua embalagem de vidro é muito frágil.

Às vezes é mais fácil montar o circuito usando um esquema, que foca no essencial que são as conexões. A ilustração abaixo é um esquema deste circuito. Tente prever o que vai acontecer quando as chaves mudarem de posição:



Aperte o botão e veja o que acontece. Depois mude as posições das chaves e aperte o botão novamente. Aproxime um imã do reed. Experimente com diferentes imãs. O de neodímio consegue fechar a chave a uma distância muito maior que o imã de ferrite

Experimente fazer alterações no circuito. Coloque um LED em série com o motor. O que acontece? Troque o interruptor de pressão por outra chave de 2 posições, para selecionar entre o circuito do motor ou o circuito da cigarra e LED.

2.4. Resistência elétrica

A **condutividade** elétrica de um material é uma medida da sua capacidade de conduzir corrente elétrica. O valor inverso da condutividade é a sua **resistência**, que mede quanto o material limita a passagem de corrente. A resistência elétrica é medida em **ohms** (**Ω**) em homenagem ao cientista alemão **Georg Simon Ohm** (1789-1854). Metais como cobre e prata, usados em fios e circuitos impressos, possuem uma resistência muito baixa (da ordem de bilionésimos de ohm).

2.4.1. Resistores

Resistores são componentes de dois terminais que possuem uma resistência **nominal e definida**. Podem ser usados para controlar o fluxo de corrente e para dividir tensão com precisão. Variam em tamanho, tipo de material usado na construção, precisão, e capacidade de dissipar calor (sua potência).

O kit contém muitos resistores. A maior parte são de carbono, têm precisão de 95% (**tolerância** de 5%) e **potência** máxima de 0,25 watts. Todos esses têm corpo de cor bege e o mesmo tamanho. Um dos resistores disponibilizados no kit é de metal. Tem precisão de 99% (tolerância de 1%), potencia máxima de 1 watt e corpo de cor azulada.

A tolerância significa que, se um resistor tem valor nominal de 1000Ω, ele pode na verdade ter entre 990 e 1010 Ω, se a tolerância for de 1%, ou 950 a 1050 se a tolerância for de 5%.

Os resistores do kit estão na faixa de 100 a 1 milhão de ohms, e são expressos usando os prefixos padrão **k** (x 1000) e **M** (x 1000000). Portanto **1000Ω é o mesmo que 1k Ω** e **1000000Ω é o mesmo que 1000kΩ ou 1MΩ**. Em resistores comerciais é também comum representar valores fracionados com k ou M no lugar da vírgula, e omitir o símbolo de ohm. Por exemplo, 2M2 e 3k3 significam 2,2MΩ e 3,3kΩ. Também se usa R no lugar do símbolo Ω.

Em esquemas, resistores são representados pelos símbolos abaixo (tanto faz usar um ou outro):

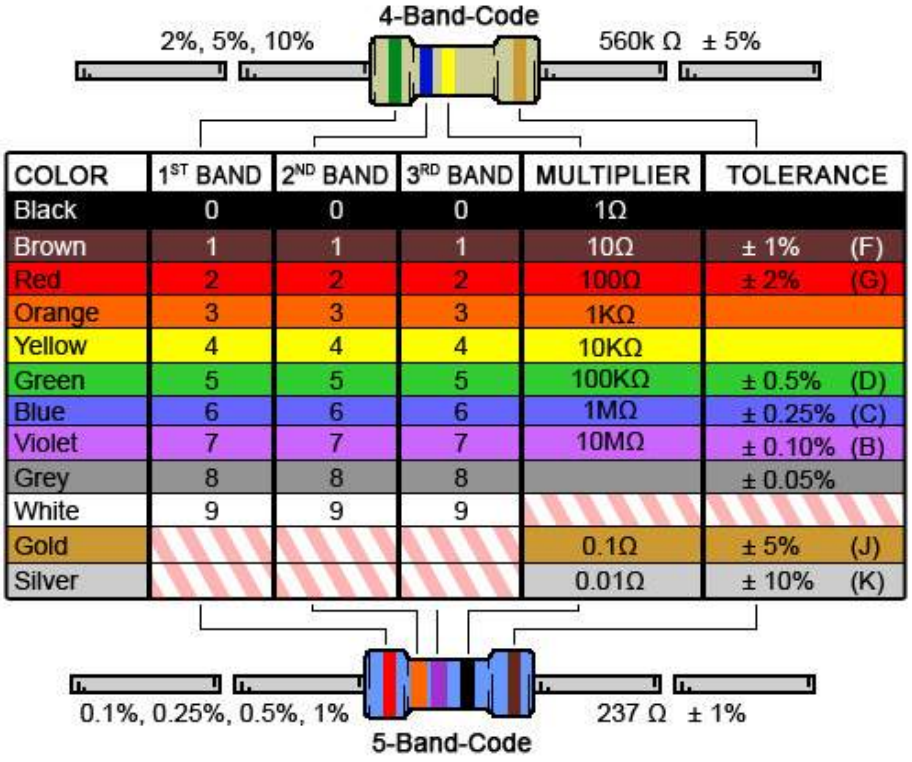
A resistência dos resistores pode ser medida no multímetro, movendo o seletor para a seção "**Ω**". O multímetro permite medir resistências desde alguns ohms até no máximo 2 mega ohms (2 milhões de ohms). As 5 posições do seletor são suficientes para medir com uma precisão razoável a maior parte dos resistores mais comuns. Este multímetro não tem precisão suficiente para medir valores de resistência muito pequenas.

2.4.2. Identificação de resistores

Resistores comerciais são identificados na sua embalagem por um **código de cores**. A maioria dos fabricantes utiliza um código de **4 faixas** pintadas no corpo do resistor. Resistores de alta precisão usam um código de **5 faixas**. Eles são semelhantes. No código de 4 faixas, as duas primeiras representam **dígitos**, e o terceiro representa um **multiplicador**. No código de 5 faixas (maior precisão), são três dígitos e um multiplicador. A ultima faixa representa a **tolerância** (margem de erro) do resistor.

Por exemplo, um resistor com quatro faixas nas cores Vermelho, Amarelo, Verde, Dourado é de 2,4MΩ. Vermelho e amarelo são os dígitos 2 e 4, respectivamente. Verde representa 5, mas aqui é um multiplicador, ou seja 10^5 = 100000. Pode-se também simplesmente adicionar cinco zeros. Logo, o valor do resistor é 2400000 ou 2,4M. Dourado é a tolerância, que de acordo com a tabela é de 5%, ou seja, o valor real do resistor é 2,4 +/- 120kΩ. A resistência deve estar entre 2,28MΩ e 2,52MΩ.

O diagrama abaixo ilustra o uso dos códigos de cores em resistores. Veja na referência de componentes no final da apostila. Lá estão identificados todos os resistores do kit com seu código de cores. Na dúvida meça a resistência usando o multímetro.



COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)

(Wikimedia)

Resistores SMD (*Surface-Mounted Device*) são resistores minúsculos **de 2mm ou menos** usados em circuitos modernos. Eles não são identificados por cores mas por três dígitos que representam os números do código. Por exemplo, um resistor com marcação **104** significa dígitos "1", "0" e multiplicador "4" (+4 zeros), ou seja, 100kΩ.

Experimento 5 – Teste de condutividade e medição de resistência

Material necessário:

- Multímetro
- Resistores diversos
- Condutores diversos: fios, jumpers, cabos com garras jacaré, um prego, linha de costura condutiva, fitas condutivas (cobre e tecido de prata), papel laminado, lápis grafite
- Chaves diversas (reed, táctil de pressão, duas posições)
- Fita condutiva
- Linha de costura condutiva
- Traço de grafite
- Resistores diversos
- Potenciômetros
- LDRs
- Termistor

Para usar a função **ohmímetro** do multímetro 830B, a ponta de prova preta deverá estar na posição COM, e a vermelha na **posição VΩmA** (mesma posição que usamos para medir tensão).

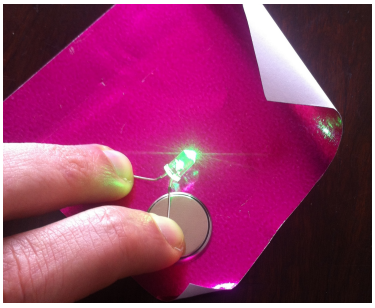
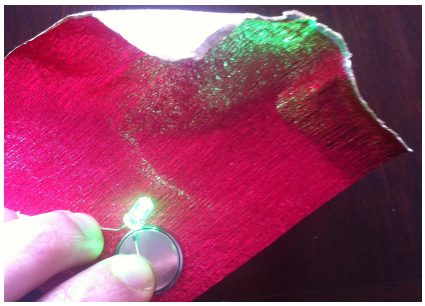
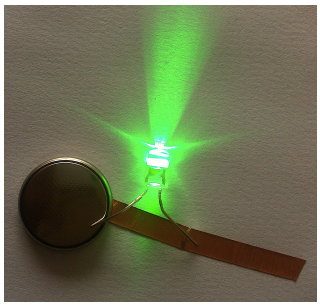
Gire o seletor do multímetro para uma das 5 posições da função **ohmímetro** (símbolo **Ω**).

Antes de iniciar, encoste uma ponta na outra. Esta é a resistência do fio, que é muito baixa, praticamente zero (muito menos do que o ohmímetro seria capaz de medir). Seria necessário 50 metros de fio de cobre com 1mm de diâmetro para chegar a 1Ω. Com as pontas encostadas o multímetro deve mostrar "000", mas nas mais baixas pode aparecer um valor pequeno, que não é a resistência do fio mas um erro devido à baixa precisão do ohmímetro.

Quando as pontas estão separadas, o valor exibido é sempre **"1___"** que indica uma resistência maior do que aquela escala é capaz de medir. O ohmímetro mede resistência até 2000k (2MΩ).

A) Teste de condutividade

Para verificar se um material conduz eletricidade ou não, você pode fazer um circuito mínimo conectando uma bateria de 3V e um LED usando o material como caminho para fechar o circuito. É um teste rápido e pode ser feito segurando a bateria, o LED e o material com os dedos. Nas fotos abaixo acendemos um LED passando corrente por cartolina laminada, papel crepom metalizado e fita adesiva de cobre:

Outra forma de medir condutividade é **usando o multímetro**. Gire o seletor para a posição 200k da função "**Ω**". Se as pontas de prova forem encostadas em duas posições de um material **condutor**, o visor deve mostrar o valor "**000**" (ou valor próximo). Se for um material **isolante**, aparecerá um número "**1**___" alinhado à esquerda, indicando a passagem de pouquíssima ou nenhuma corrente.

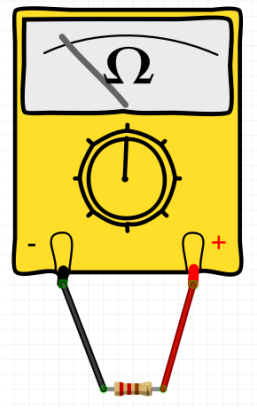
OPEN CIRCUIT DISPLAY

SHORT CIRCUIT DISPLAY *(jbriant.eu)*

1. Experimente com **um fio**. Use o multímetro na **posição 2000k** coloque uma ponta de prova de cada lado do fio. O valor deve ser zero ou quase zero, indicando que o fio não oferece nenhuma ou pouquíssima resistência à corrente.
2. Teste a condutividade de **diversos materiais**: sua roupa, sua pele, sua língua, um prego, linha de costura condutiva, fita de cobre, fita de prata, lápis grafite, uma linha desenhada em grafite numa folha de papel. Os metais geralmente conduzem melhor. A linha de costura incluída no kit é de aço inoxidável, e as fitas adesivas são condutivas.
3. Teste as chaves que você usou no experimento anterior. Prenda, usando garras jacaré, as pontas de prova do multímetro aos terminais de uma **chave táctil de pressão**. O multímetro deve indicar que não há condutividade, mas quando você apertar o botão, o visor deve indicar que o circuito está conduzindo. Teste a **chave de duas posições** e descubra qual dos lados está inicialmente fechado, e qual está inicialmente aberto usando o multímetro. Quando você mover a alavanca da chave, as posições devem se inverter. Teste a **chave magnética reed**. Prenda com garras jacaré os dois terminais a pontas de prova do multímetro, que deve indicar um circuito aberto. Aproxime (com cuidado) um **imã de neodímio** do reed e veja que a chave se fecha, fazendo o circuito conduzir.

B) Medição de resistência

Escolha 5 a 10 resistores e meça sua resistência encostando as pontas de prova nos terminais.



Se ao medir a resistência o multímetro exibir um **"1___"** à esquerda, e a faixa selecionada não for 2000k, gire o seletor para uma faixa maior até que apareça um valor. Se ainda assim o "1___" for exibido, a resistor tem mais que 2MΩ. Tente identifica-lo pelo código de cores.

Se o valor for muito pequeno, quase zero (ex: **001**, **002**) melhore a precisão da leitura girando o seletor para uma faixa menor. Lembre-se que resistências muito pequenas não são medidas com precisão.

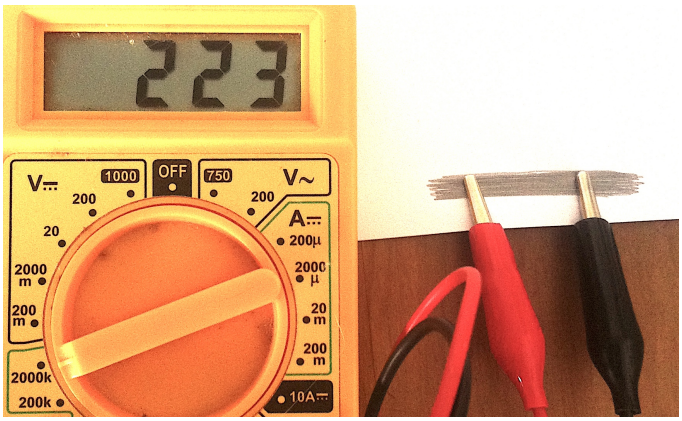
Compare os valores medidos com os valores impressos (através de código de cores) em cada resistor. O valor provavelmente não será exato. As variações se devem à imprecisão do ohmímetro (principalmente nos valores mais baixos), e à tolerância do resistor (principalmente nos valores mais altos).

Como vimos, não é possível medir valores baixos demais de resistência devido à baixa precisão do ohmímetro que temos no kit, mas o multímetro é suficiente para medir valores práticos (acima de 10 ohms) que podem afetar o funcionamento de um circuito. Use o multímetro para medir a resistência dos seguintes materiais:

- 1 metro de **linha condutiva** (disponível no kit)
- 1 metro de **fita de cobre** (disponível do kit – tente morder com o jacaré as duas extremidades, sem tirar da embalagem)
- 0,5 metro de **fita de tecido de prata** (há duas disponíveis no kit)

Se tiver um **lápis grafite**, meça a resistência entre uma ponta e outra.

Você também pode medir a resistência de um desenho. Em uma folha de papel, desenhe uma linha grossa perto da borda da página com uns 5cm de comprimento e 0,5cm de largura. Morda as extremidades do desenho com garras jacaré, e conecte as outras pontas do jacaré nas pontas de prova. Ajuste o ohmímetro até aparecer a resistência. No desenho abaixo, temos 223kΩ de grafite.



Aumente a **largura** do desenho com o lápis grafite e a resistência deve diminuir, já que a corrente encontrará um caminho mais largo por onde passar. Se você usar um lápis grafite mais mole (ex: 6B ou 9B) poderá desenhar caminhos de poucas centenas de ohms, suficiente para acender um LED.

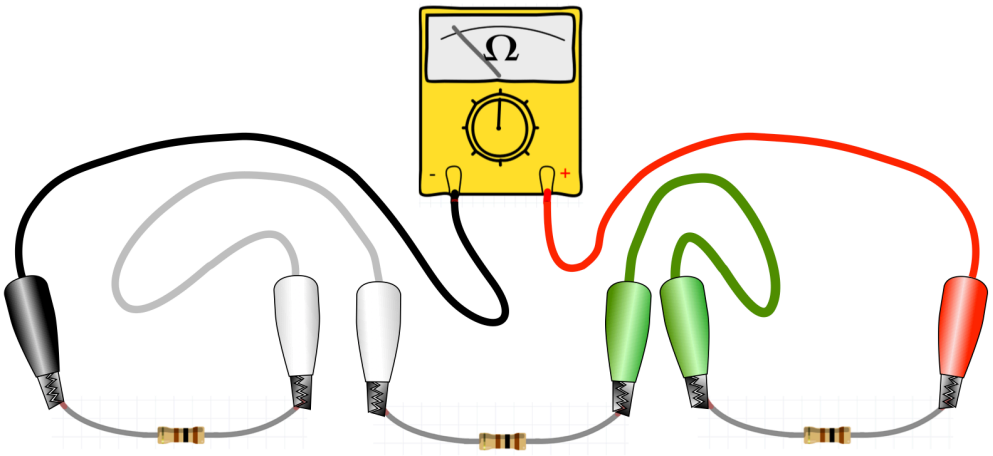
Meça a resistência de uma **folha de cartolina laminada**. Comece com uma distância pequena e aumente gradualmente deslizando a ponta de prova sobre a folha.

C) Resistores em série e paralelo

Você viu que a resistência de um caminho de grafite desenhado no papel com 1cm de largura tem aproximadamente **metade** da resistência de um caminho com apenas 0,5cm de largura. Isto é porque a corrente tem mais espaço por onde fluir. É análogo a um encanamento conectado em um sistema hidráulico de alta pressão. Se você adicionar mais canos, a resistência à pressão vai diminuir. Se os canos tiverem **diâmetro** maior, também. Por outro lado vimos vários materiais cuja resistência aumenta com o **comprimento**. Por exemplo, meio metro de linha condutiva tem metade da resistência de um metro de linha condutiva.

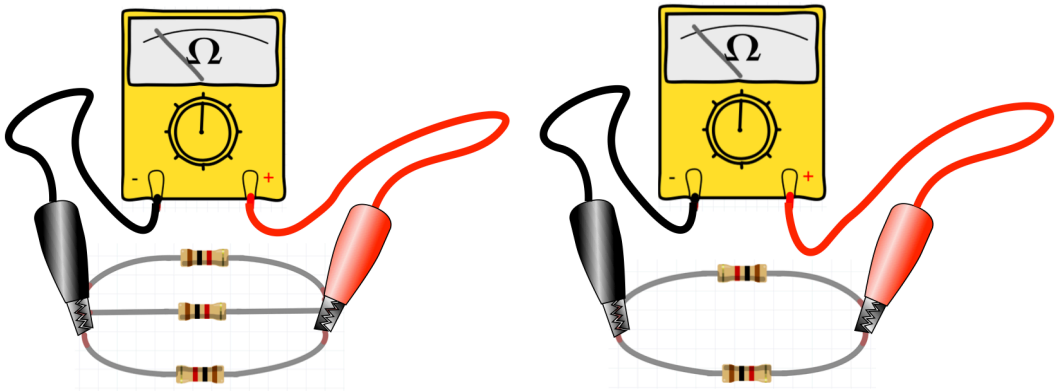
Usando resistores podemos representar valores de resistência com precisão, e obter valores diferentes de resistência conectando-os **em série** (somando suas resistências) ou **em paralelo** (dividindo a resistência proporcionalmente).

Escolha três resistores de mesmo valor (ex: 100Ω ou 1kΩ). Meça seus valores individualmente e depois e conecte-os como mostrado abaixo.



O valor deve ser aproximadamente **o triplo**. Tente outras combinações de resistores e veja que seus valores **se somam**.

Agora ponha os resistores em paralelo e veja que o valor de resistência resultante é **sempre menor que** o valor do menor resistor.



Você pode calcular esses valores. A fórmula para resistores em série é simplesmente a soma de suas resistências:

```
R = R1 + R2 + R3 + ...
```

Por exemplo, se R1 for 100Ω, R2 for 470Ω e R3 for 1kΩ, o valor de R será:

```
R = 100 + 470 + 1000 = 1,57kΩ
```

A resistência de resistores em paralelo é o seu valor médio. Se forem dois resistores iguais, a resistência será metade. Se forem três iguais, a resistência será 1/3. Se forem diferentes, use a fórmula:

```
1/R = 1/R1 + 1/R2 + 1/R3 + ...
```

Se houver apenas dois resistores, você pode usar uma equação mais simples:

```
R = (R1 x R2) / (R1 + R2)
```

Por exemplo, o valor paralelo de 100Ω, 470Ω e 1kΩ será:

```
1/R = 1/100 + 1/470 + 1/1000 = 0,01 + 0,00213 + 0,001 = 0,01313
R = 76,16 Ω
```

2.5. Teoria básica de circuitos resistivos e Lei de Ohm

Nesta seção exploraremos um princípio fundamental da eletrônica que é a relação entre corrente, tensão e resistência. Iniciaremos com a Lei de Ohm que é uma relação matemática simples (apenas multiplicação e divisão) que permite descobrir um desses valores, tendo-se os outros dois. Depois veremos como medir e estimar corrente e tensão em circuitos básicos.

2.5.1. Lei de Ohm

A **Lei de Ohm** é uma relação **linear** entre corrente, tensão e resistência. É uma equação fundamental para analisar e projetar circuitos e saber como limitar valores de tensão e corrente. A lei de Ohm é expressa através da fórmula:

```
V = R * I
```

Onde **V** é a tensão em volts, **R** a resistência em ohms, e **I** é a corrente em amperes.

Ao fazer os cálculos é importante levar em conta as **unidades**. Por exemplo, se a corrente estiver em **mA** (**miliamperes**), multiplique o valor por 1000, para que ela fique em **A** (**amperes**), e possa ser usada na fórmula. Ou se a resistência estiver em **MΩ** (**megaohms**), divida antes por **1000000** para converter o valor para **Ω** (**ohms**).

A Lei de Ohm pode ser usada para **calcular o valor de resistência** necessária para limitar a corrente de um componente, quando se conhece a tensão e corrente sobre ele, usando:

```
R = V / I
```

Por exemplo, suponha um circuito formado por um resistor alimentado por 9V. Para garantir que uma corrente 10mA esteja passando por ele, ele deve ter uma resistência de:

```
R = 9 / 0,01 = 900 ohms
```

Finalmente, para calcular a corrente conhecendo-se os valores da tensão e resistência, use:

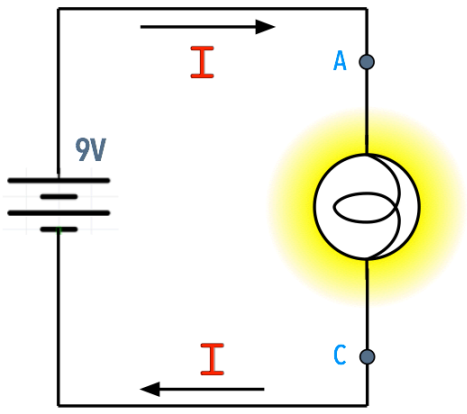
```
I = V / R
```

2.5.2. Circuitos em série e em paralelo

Um circuito é como um encanamento. Existem canos abertos, outros fechados. Uns estreitos ou entupidos que limitam a vazão retendo a pressão da água, outros largos onde a água flui livre. De maneira análoga, em um circuito as correntes se bifurcam por diversos caminhos. Há caminhos de alta resistência que provocam queda de tensão e a corrente é muito baixa, e outros onde a corrente flui livremente. **Componentes que oferecem mais resistência, retém mais tensão**, diminuindo a corrente que flui pelo trecho. Os que oferecem pouca ou nenhuma resistência, deixam passar toda a corrente que recebem, e praticamente não retém tensão.

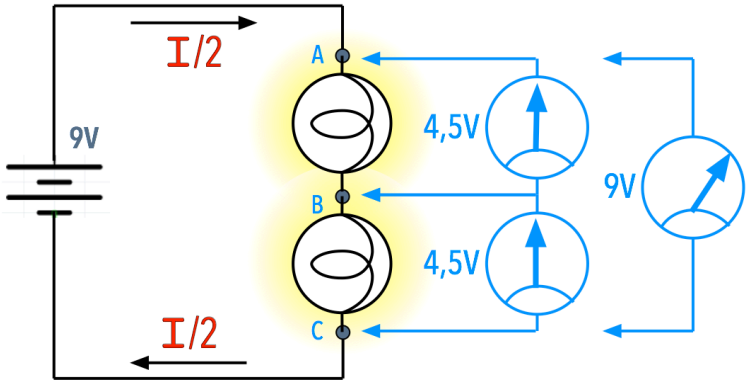
Portanto, **a tensão varia ao longo do caminho** seguido pela corrente, iniciando com o valor da bateria, e terminando em zero, mas toda a corrente que entra no circuito pelo terminal positivo da bateria, retorna ao negativo, portanto **a corrente que entra é sempre a mesma que sai**.

Considere o circuito abaixo onde uma lâmpada é alimentada por uma bateria de 9V.



A corrente **I** que entra no circuito é igual à corrente que sai. Considerando que os fios que ligam a bateria à lâmpada sejam ótimos condutores, toda a queda de tensão da bateria estará sobre a lâmpada, que oferece uma resistência à passagem de corrente. É a passagem de corrente pelo filamento que faz com que ele fique incandescente gerando a luz.

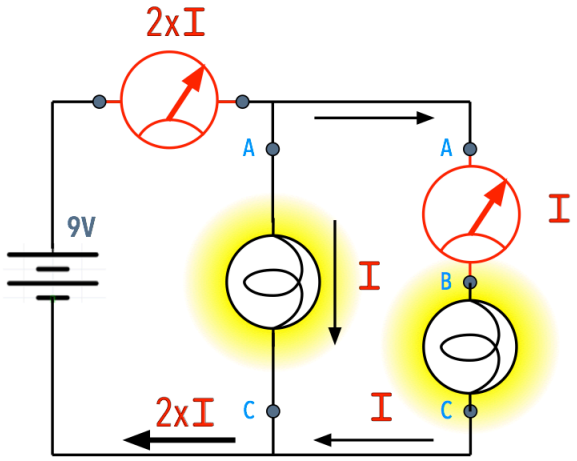
Agora vamos conectar um dos terminais da lâmpada a outra lâmpada idêntica, e os terminais restantes nos pontos A e C do circuito acima. Teremos um circuito com duas lâmpadas em série. Como as lâmpadas são **idênticas**, elas têm uma resistência interna igual, e como duas resistências em série se somam, a resistência do circuito agora é o dobro do que era antes, fazendo com que a corrente caia pela metade (e que o brilho das lâmpadas também diminua).



Outra maneira de entender porque as lâmpadas emitem menos luz, é que a diferença de potencial em cada uma delas é menor. Embora a corrente seja a mesma em todos os pontos, a tensão se divide entre as cargas. Em cada componente há uma **"queda de tensão"** que é **proporcional** à sua resistência. Um princípio importante é que a soma das quedas de tensão em um circuito equivale ao total de tensão fornecido pelo circuito, ou seja, à tensão da bateria.

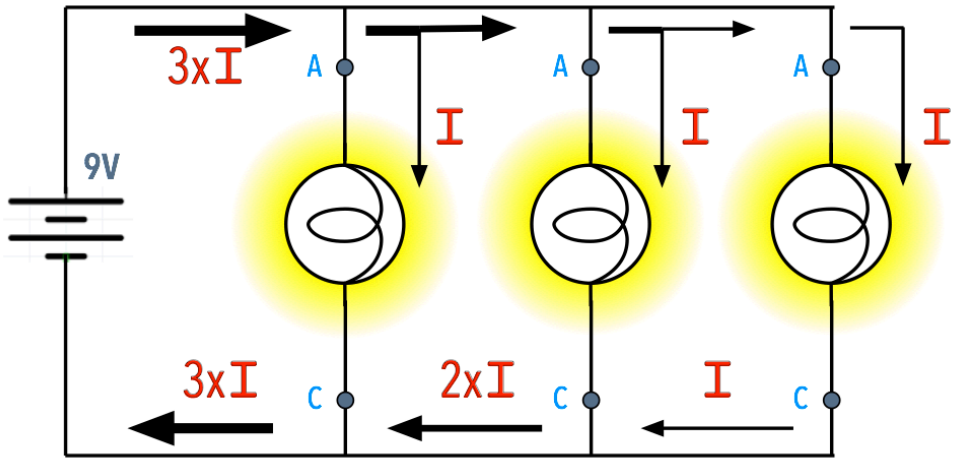
Se medirmos a queda de tensão em cada uma, veremos que há apenas 4,5V em cada lâmpada. É como se cada uma delas estivesse sendo alimentada individualmente por uma bateria com metade da carga. Menos tensão gera menos corrente, e consequentemente menos brilho.

Outra maneira de conectar as lâmpadas na bateria é ligá-las em paralelo. Neste caso, oferecemos **dois caminhos** para a corrente, e a **tensão sobre cada lâmpada é a mesma**. A tensão oferecida pela bateria corresponde a uma diferença de potencial elétrico fixo. Mas a corrente é a quantidade de elétrons circulando durante um intervalo de tempo. Se a bateria for capaz de fornecer mais elétrons (assim descarregando mais rapidamente), cada lâmpada poderá ter a mesma corrente que no circuito com uma única lâmpada. Mas para isto, o circuito irá demandar da bateria **o dobro** da corrente.



Nas instalações elétricas residenciais, lâmpadas são instaladas em paralelo. Desta forma, novas lâmpadas não interferem no brilho das outras, já que a tensão sobre elas é a mesma, mas aumentam a demanda de corrente (e o consumo residencial).

Se for acrescentada uma terceira lâmpada em paralelo, ela demandará a mesma corrente, e a bateria precisará ter capacidade de fornecer três vezes a corrente para o circuito.

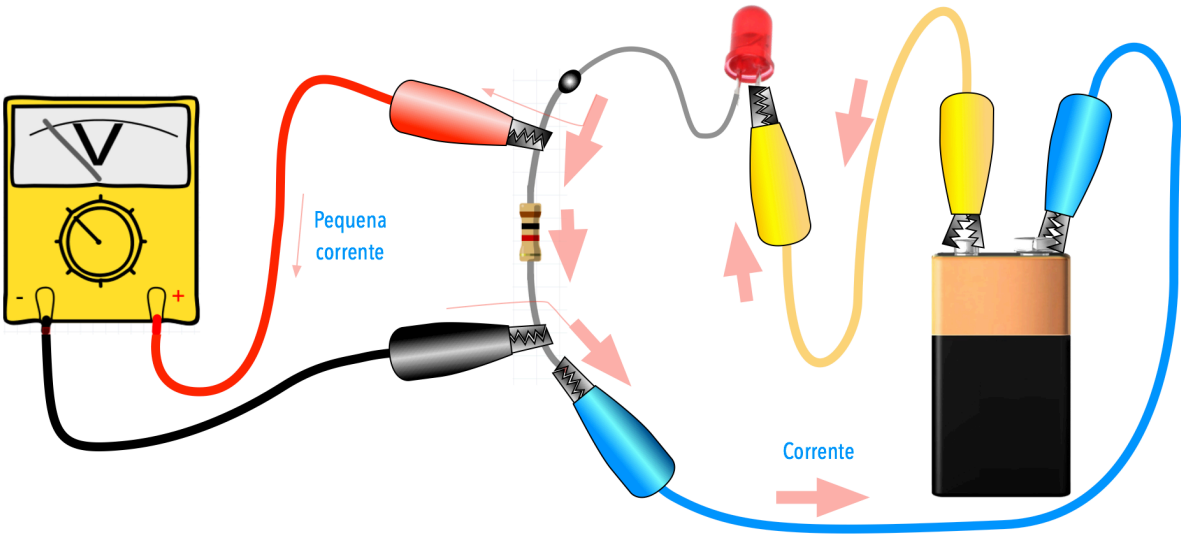


Nem sempre uma bateria é capaz de fornecer a corrente necessária, e isto irá fazer com que a sua tensão caia (é o que acontece com a batata quando tentamos alimentar um LED que demanda mais corrente que uma única batata consegue fornecer).

2.5.3. Medição de tensão

Podemos medir a tensão **entre dois pontos** de um circuito usando o multímetro na função *Voltímetro*, posicionando-o **em paralelo** com o componente (ou seja, encostando as pontas de prova nos terminais do componente, **enquanto ele está ligado no circuito**). A conexão do multímetro em paralelo garante que a mesma tensão que estiver na carga, também estará no multímetro (como a resistência interna do multímetro é muito alta, apenas uma minúscula corrente irá circular dentro dele, insuficiente para interferir na medição).

A ilustração abaixo mostra a medição da queda de tensão sobre um resistor. Durante a medição, uma minúscula corrente flui pelo multímetro, mas isto na prática não afeta o resultado.



Se um circuito contém apenas uma bateria e uma carga, toda a tensão estará sendo aplicada à carga. Mas se ele contiver duas cargas interligadas **em série** (de forma que haja apenas **um caminho para a corrente**), a queda de tensão será dividida entre os dois, **proporcionalmente** às suas resistências.

No experimento a seguir mediremos as quedas de tensão sobre **duas cargas ligadas em série**. As cargas são **resistores**.

Experimento 6 – Introdução ao protoboard e divisor de tensão

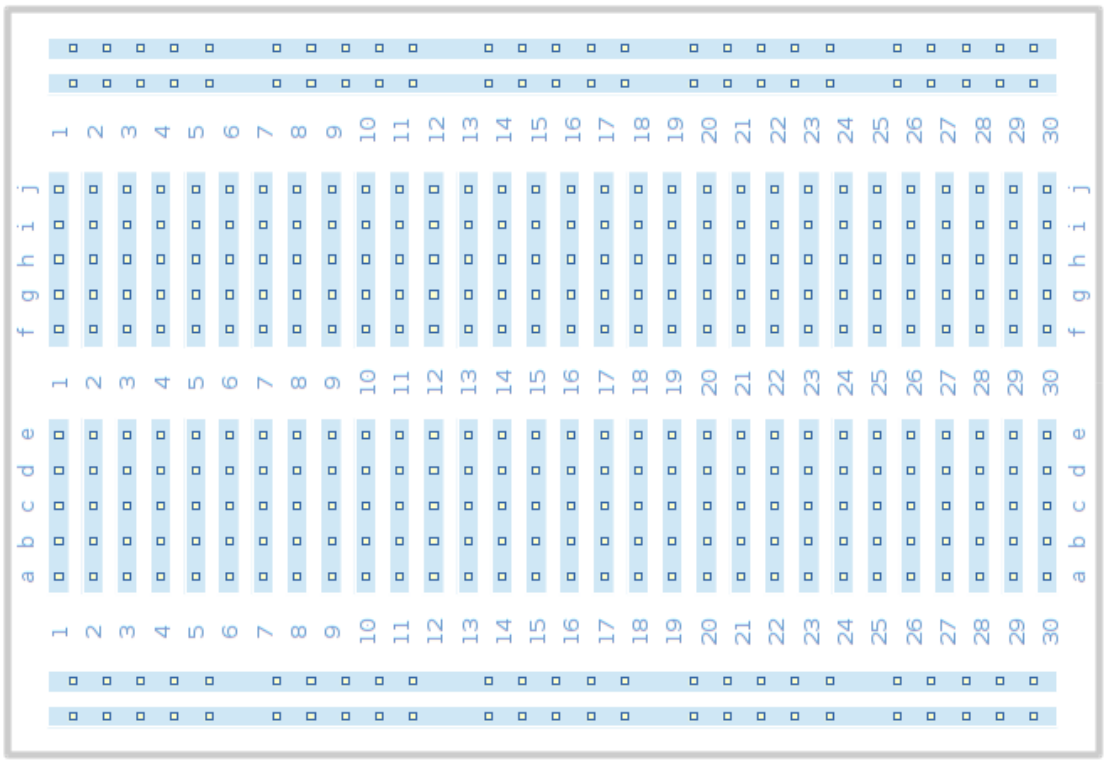
Material necessário:

- Protoboard
- Multímetro
- 2 resistores de 1k e resistores de 100 ohms, 10k, 100k e 1M (1 de cada)
- Fios e jumpers

A) Introdução ao protoboard

A partir deste experimento usaremos o **protoboard** como uma alternativa para montar circuitos. O protoboard (também chamado de breadboard), é uma **base de furos interligados** usados para construir **protótipos**. Ele permite que componentes sejam inseridos e removidos de um circuito com facilidade. Podemos continuar a fazer circuitos simples com garras jacaré, mas à medida em que tivermos que realizar mais conexões isto ficará inviável. O protoboard é ideal para experimentar e testar diferentes configurações. Quando você terminar de testar o seu circuito, e ele estiver de acordo com o que você deseja, você poderá montá-lo em algum lugar definitivo.

O protoboard incluído no kit possui **830 furos** onde são inseridos terminais dos componentes e fios. Para construir circuitos com ele é preciso conhecer como estão interligados esses furos internamente. Na ilustração abaixo, que mostra o raio-X de meio-protoboard, os retângulos representam condutores que interligam os furos. Isto significa que se você inserir o terminal de um componente no furo **a3**, e o terminal de outro no furo **e3**, eles estarão conectados.

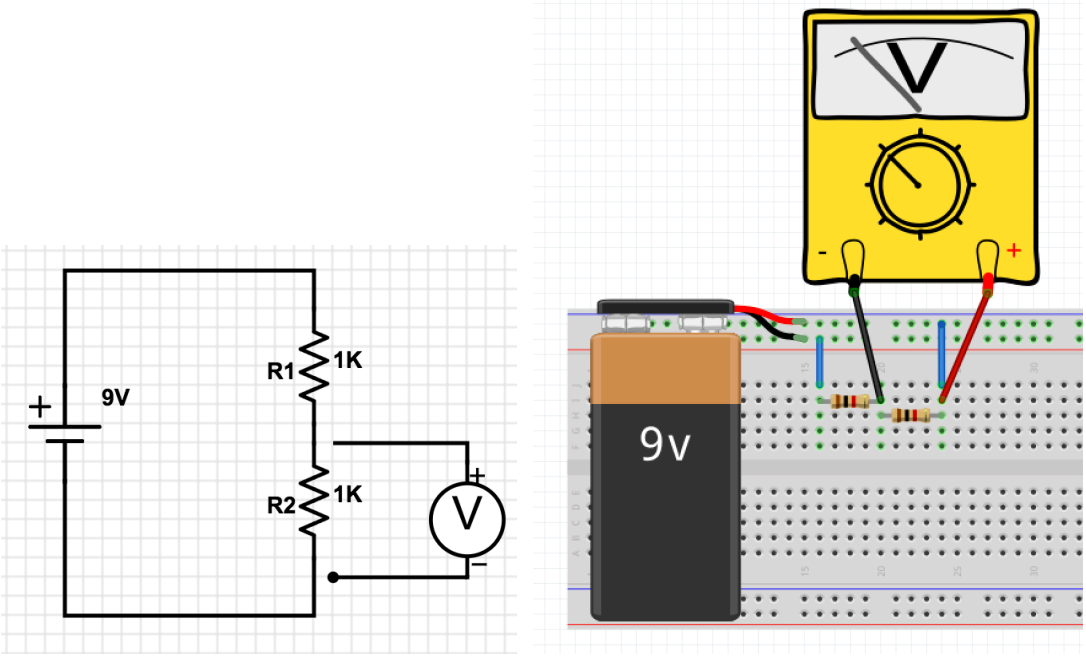


Nas laterais, as colunas marcadas **+** e **–** consistem de duas conexões de 15 furos cada. Em protoboards de 60 linhas, é comum que as linhas laterais se estendam de uma ponta a outra, mas em alguns protoboards, há uma interrupção no meio (às vezes até mais de uma). Na dúvida, retire o adesivo no fundo (ou meça a continuidade com o multímetro) para saber como é a configuração do seu protoboard.

Veja mais detalhes sobre o protoboard incluído no kit, na referência no final desta apostila.

B) Medição de tensão

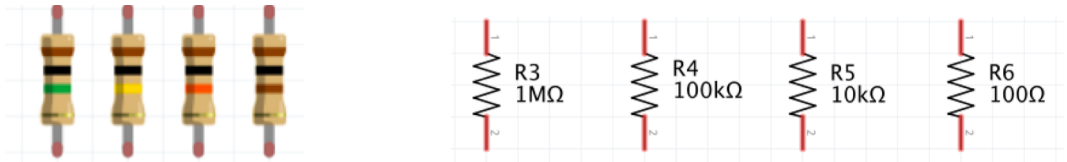
Monte o circuito representado pelo esquema abaixo. A ilustração à direita mostra uma maneira de montar o circuito usando o protoboard:



Para fazer as medições, escolha uma posição do voltímetro que seja superior à tensão da bateria ou fonte (no nosso caso, escolha a posição 20V).

Meça primeiro a tensão entre os terminais da **bateria**. Depois meça as tensões nos terminais de **cada resistor**. Anote os resultados. Como os dois resistores têm o mesmo valor nominal, a tensão sobre eles deve ser bem próxima.

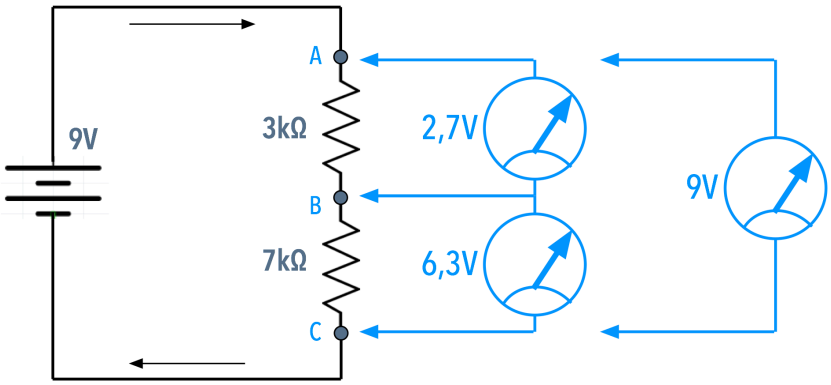
Agora substitua um dos resistores por outro da lista abaixo, e veja como mudam as tensões em cada resistor e sobre a bateria.



Se os resistores forem diferentes, a tensão sobre eles será diferente. Compare os resultados.

2.5.4. Divisor de tensão

O experimento anterior foi uma demonstração prática do **divisor de tensão**, um circuito e conceito muito importante na eletrônica. É preciso saber calcular ou medir a queda de tensão sobre um componente para que você possa saber como controlar a corrente que passa por ele, através do cálculo de resistores.



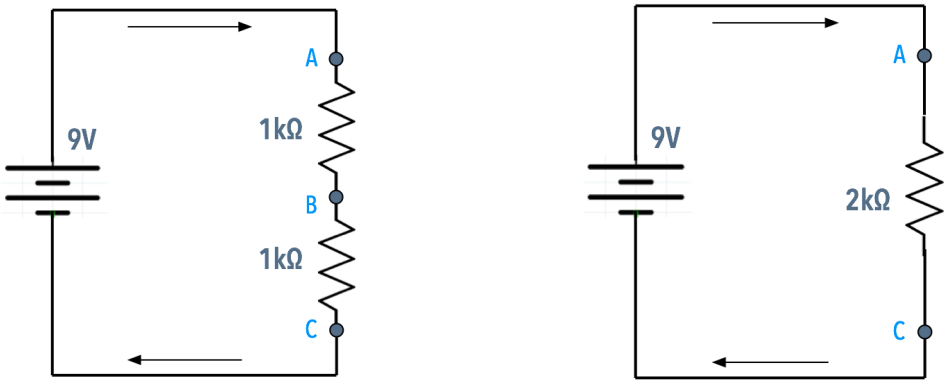
Em vez de medir as tensões, você pode usar as fórmulas abaixo para **calcular a queda de tensão** em cada resistor acima. A queda de tensão em R1:

V_{AB} = 9 x 3k / (7 + 3)k = 27/10 = 2,7 V

E a tensão em R2:

V_{BC} = 9 x 7k / (7 + 3)k = 63/10 = 6,3 V

Vimos que as resistências em série se somam, portanto os dois circuitos abaixo são equivalentes:



Agora deve ser fácil entender que a tensão da bateria se divide igualmente em cada resistor.

Podemos usar a Lei de Ohm para calcular a corrente, que é a mesma nos dois circuitos. Para isto usamos como valor de tensão V **o valor da queda de tensão sobre o componente**. Por exemplo, para calcular o valor da corrente podemos usar um dos resistores:

```
I = VAB / R = 4,5 / 1000 = 0,0045 A = 4,5mA
```

Ou a soma deles. Tanto faz, já que a corrente é uma só:

```
I = VAC / R = 9 / (1000 + 1000) = 9 / 2000 = 4,5mA
```

Como exercício, calcule a corrente que flui no circuito mostrado no início desta seção (que tem resistores de 3k e 7k em série).

2.5.5. Medição de corrente

Medir corrente não é tão simples quanto medir tensão. A corrente é medida incluindo o multímetro **em série** com o trecho do circuito por onde flui a corrente a ser medida. É preciso abrir o circuito e fazer a corrente fluir **por dentro do multímetro**. É preciso ter cuidado pois a capacidade de corrente mesmo de uma bateria irá superar o limite máximo suportado pelo multímetro se o circuito não houver nenhuma carga ou se ela não oferecer resistência suficiente.

O multímetro distribuído no kit possui dois **amperímetros** (medidores de corrente). Um deles mede correntes até 200mA. O segundo mede correntes até 10 A. Para selecioná-los, não basta girar o seletor. É preciso plugar uma das pontas de prova em local diferente.

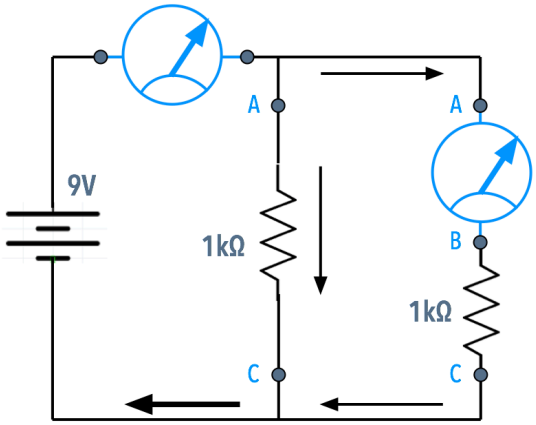
Sempre comece configurando o multímetro na posição de máxima corrente (10A), com a ponta de prova vermelha plugada no primeiro soquete (indicado 10 A). Apenas se o valor medido for inferior a 0,1 A, insira o cabo no segundo soquete (mA) com o seletor na posição 200mA, e gradualmente gire para valores menores até obter um valor que possa ser medido. Mesmo com essa precaução, motores e transformadores podem produzir pulsos curtíssimos de corrente muito intensos quando ligados e desligados, que podem queimar o fusível do multímetro.

Neste curso introdutório não faremos medição de corrente, embora ela seja uma medida importante para estimar o consumo de bateria de um circuito. Mas, conhecendo as tensões e resistências, podemos calcular a corrente que flui em um circuito.

2.5.6. Divisor de corrente

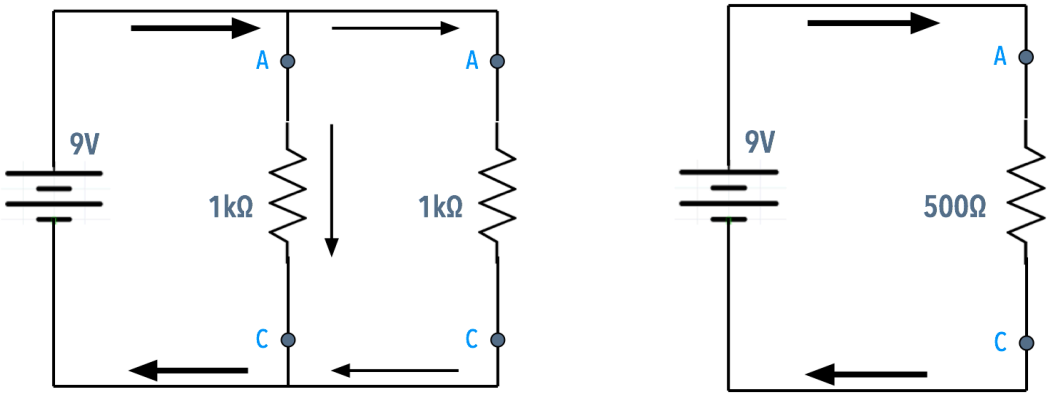
A corrente que entra em um circuito é sempre a mesma que sai, mas quando o caminho em um circuito se bifurca, a corrente **se divide**. A ilustração abaixo mostra um circuito com dois resistores

em paralelo. A corrente que é medida pelo primeiro amperímetro será o dobro da que é medida pelo segundo, já que a corrente que passa pela bateria contém a soma das correntes que fluem por cada um dos resistores. **A tensão em cada resistor é a mesma**. Veja que os pontos A e C de cada resistor são o mesmo ponto.



A quantidade de corrente que passará em cada trecho depende de sua resistência. Se os resistores forem diferentes, correntes diferentes passarão por cada trecho, mas a corrente que passa na bateria, que é a **soma das correntes que passam em cada resistor**, será a mesma.

Outra forma de analisar o divisor de corrente é considerar o efeito causado por resistores em paralelo. Nos dois circuitos abaixo, a corrente que passa pela bateria de 9V é a mesma.



2.5.7. Potência máxima de um componente

Resistores, diodos e outros componentes têm uma indicação máxima de potência. Esse valor, quando atribuído a **um componente**, refere-se à sua capacidade de **dissipar calor**. É muito importante observar esse valor para não sobrecarregar os limites de dissipação de potência de um componente. A potencia é calculada multiplicando a queda de tensão e a corrente sobre um componente:

```
P = V x I
```

A potencia P é medida em watts (W) em homenagem ao cientista escocês **James Watt** (1736-1819). **1 watt** é 1 volt vezes 1 ampere. Também se usa miliwatts para valores menores e quilowatts e megawatts para grandes valores de potencia.

Por exemplo, se uma bateria de 9V é ligada a um resistor de 10 ohms, sua corrente, pela Lei de Ohm é:

```
I = V / R = 9 / 10 = 0,9 A = 900 mA
```

E a potencia dissipada será:

```
P = 0,9 x 9 = 8,1W
```

O resistor usado deve ser capaz de suportar essa potência, caso contrário irá esquentar e queimar. Os resistores que temos no kit são quase todos de ¼ de watt (0,25 W) e não seriam suficientes. Para este cenário, teríamos que usar um resistor de 10W.

2.6. Semicondutores: diodos e LEDs

Semicondutores são materiais que oferecem maior resistência à passagem de eletricidade que os condutores, mas têm propriedades químicas interessantes que os fazem funcionar de forma especial em determinados níveis de corrente e tensão.

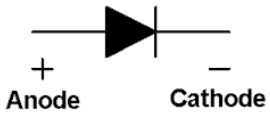
Dentre os diversos materiais semicondutores existentes na natureza, o mais importante deles para a eletrônica moderna é o **silício**. Os principais componentes que fazem funcionar os computadores, celulares e aparelhos eletrônicos modernos em geral são feitos, na sua maior parte, de silício.

2.6.1. Diodos

Um **diodo** é um componente polarizado que **só permite a passagem de corrente em um sentido**. Até os anos 50, diodos eram frágeis válvulas de vidro contendo filamentos à vácuo (como as lâmpadas incandescentes). Após as invenções que deram origem à **eletrônica de estado sólido**, eles passaram a ser feitos com materiais semicondutores e ficaram muito menores e muito mais robustos. Diodos de silício são compostos por uma junção de dois materiais semicondutores com propriedades elétricas opostas misturados com silício.

Diodos são muito usados para **proteger circuitos** contra correntes em sentido contrário (acontece com motores, relés e transformadores) e para **retificar corrente alternada** (transformar corrente alternada em corrente contínua). A maior parte dos diodos emite apenas calor, mas alguns emitem luz.

Diodos têm uma polaridade e funcionam de forma diferente se ela for invertida. O símbolo abaixo é usado para representar um diodo:

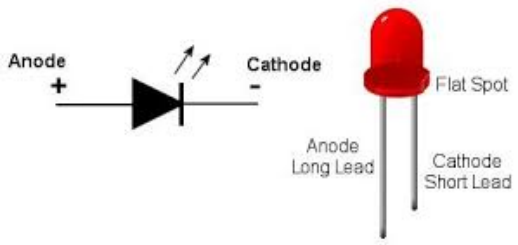


Os polos são identificados com os mesmos termos usados para identificar terminais de uma bateria (anodo - **A** e catodo - **K**), mas com polaridade oposta.

2.6.2. LEDs

LED significa *Light-Emmiting Diode*, ou **diodo emissor de luz**. Um LED é um diodo construído para emitir radiação em uma faixa de frequências estreita, e mais alta, que corresponde ao espectro de luz (inclusive luz invisível como infravermelho e ultravioleta).

O símbolo abaixo é usado para representar um LED. Em geral a embalagem possui um chanfro do lado do catodo, que deve ser ligado ao negativo:

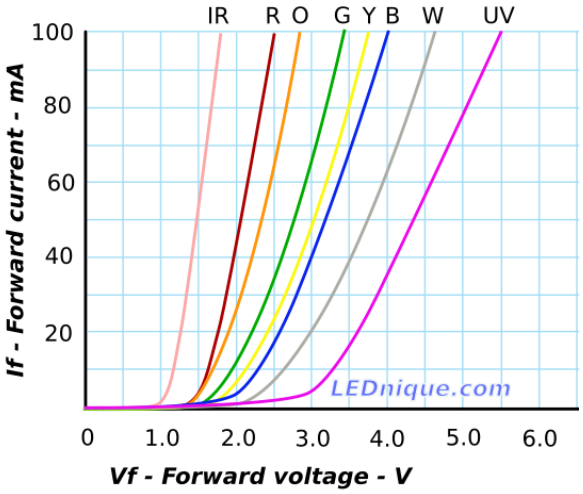


Existem LEDs de várias cores e tamanhos. As cores do espectro (vermelho, laranja, amarelo, verde, azul, violeta) são puras. Branco e cores compostas (rosa, verde-água) são produzidas combinando leds diferentes. As cores dos LEDs são identificadas pelo seu **comprimento de onda** no espectro de cores. No kit há vários LEDs vermelhos, e outros com a embalagem transparente que produzem uma cor apenas quando são ligados.

Importante: no kit há também um *fototransistor*, que é um sensor de luz e **não é um LED, mas a sua embalagem é idêntica**. Ele tem pernas mais longas. Se você tentar acender um LED do kit, de pernas longas, transparente, e ele não acender, pode ser um fototransistor (mas pode também ser um led queimado).

LEDs não tem uma relação linear entre corrente e tensão, como os resistores. Os LEDs só acendem quando alcançam uma tensão específica (sua **tensão direta**) que permanece praticamente constante enquanto a corrente sobe rapidamente. Portanto, LEDs geralmente não podem ser ligados diretamente em uma bateria, pois queimarão em pouco tempo. É necessário conectar um **resistor para limitar a corrente** em série com o LED. Para calcular o resistor, é preciso saber qual **a tensão direta** do LED usado. Ela depende principalmente de sua cor. Valores **típicos** aproximados, na corrente máxima sustentável de operação (20 mA) são:

- Infravermelho:**1,6V**
- Amarelo:**2V**
- Verde:**2V**
- Vermelho:**2V**
- Azul:**3V**
- Branco:**3V**
- Rosa:**3V**
- Ultravioleta: **3,2V**



(lednique.com)

Observe o gráfico que relaciona a corrente direta (If) e tensão direta (Vf) de vários LEDs. Eles começam a emitir alguma luz com cerca de **5mA**. Em **20-25mA** LEDs típicos atingem o brilho máximo **sustentável**. É possível fornecer mais corrente, e fazê-lo brilhar mais intensamente, mas por períodos curtos de tempo. Um **pulso de corrente de 100mA** ou mais geralmente queima o LED imediatamente. Valores menores queimam em alguns segundos a minutos. Fornecer uma **tensão inversa** ao LED nem sempre vai queimá-lo, a menos que ela exceda seu valor máximo suportável, que para os LEDs do kit é de aproximadamente 5 volts.

Para calcular o resistor ideal para um LED, é preciso **subtrair** a tensão no LED da tensão fornecida ao circuito (ou trecho do circuito), e **dividir** pela corrente nominal (ex: 20mA), de acordo com a Lei de Ohm. Por exemplo, para usar um LED vermelho (queda de 2V), na capacidade máxima (20mA) com uma bateria de 9V, o resistor deve ser de:

```
R = V/I = (9 V – 2 V) / 0,02 A = 350Ω
```

Como não existem comercialmente resistores de 350Ω, podemos usar um de 390Ω, que é maior e garante uma corrente de um pouco menos que 20mA, ou mesmo de 470Ω, que deixará passar uma corrente de:

```
I = V/R = 7 V / 470Ω = 0,0149 ~ 15mA
```

Não é o brilho máximo, mas é um valor seguro que permitirá que o LED tenha uma vida longa. Se o LED vai ficar aceso por períodos curtos, também é possível passar um pouco de 20mA e usar um resistor de 330 Ω.

O cálculo do resistor limitador do LED é simples, mas se você quiser pode usar sites que fazem cálculos para LEDs. Veja alguns links no final desta apostila.

Veja também no final da apostila maiores detalhes sobre os diferentes LEDs incluídos no kit. A identificação dos LEDs depende da embalagem. **LEDs RGB** possuem 4 a 6 terminais, e contém três LEDs na mesma embalagem. Cada um deles representando uma cor. Para ligar cada cor, é preciso observar o esquema e saber quem é cada terminal. Se você queimar um LED RGB, é provável que algumas cores ainda funcionem.

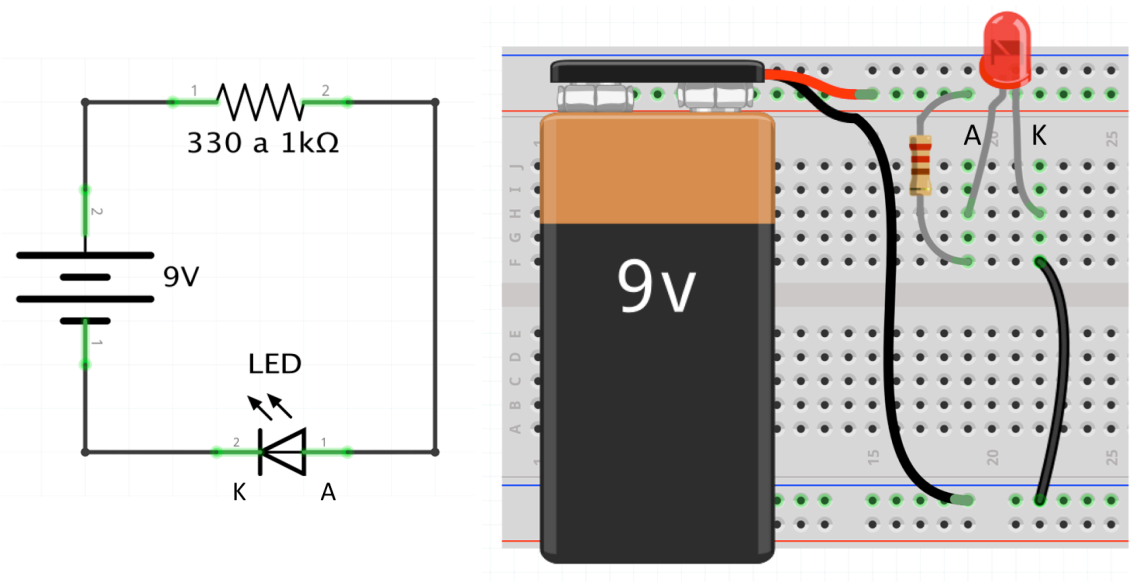
Experimento 7 – Acendendo LEDs com 9 e 12v

Neste experimento calcularemos resistores para limitar a corrente dos LEDs e permitir que operem em brilho máximo sem correr o risco de queimar.

Material necessário:

- LEDs diversos
- Resistores de 330 Ω, 470 Ω, 560 Ω, 680 Ω e 1k Ω.
- Protoboard, fios e jumpers
- Fonte de 9V ou 12V, ou bateria de 9V

Construa um circuito ligando um LED em série com um resistor e uma bateria ou fonte de acordo com o esquema e ilustração abaixo:



Calcule o resistor ideal para acender o LED de forma que ele funcione com brilho máximo sem risco de queimar. Leve em conta os dados a seguir:

1. A corrente máxima de um LED comum é **20mA.**
2. A tensão que um LED utiliza é **fixa** (não é linear, como no resistor). Pode ser de **2V** (LEDs vermelhos, amarelos) a **3V** (azuis e brancos). Veja o gráfico e tabela de queda de tensões de diferentes LEDs mostradas acima.
3. Descubra a tensão no **resistor** subtraindo a tensão do LED: **tensão da fonte – tensão do LED = tensão do resistor.**
4. Divida a **tensão do resistor** por **20mA**, para descobrir sua resistência. Use um resistor de valor igual ou maior.

Experimente usar resistores de valor maior, e veja se o brilho do LED diminui muito.

Experimente também ligar mais LEDs **em paralelo** (catodo com catodo, anodo com anodo), mas em série com o mesmo resistor (ligado no anodo ou catodo). O que acontece?

Experimente ligar mais LEDs **em série** (bateria+, resistor, LED1, LED2). Qual o máximo de LEDs que podem ser colocados em série?

2.7. Potenciômetros e sensores resistivos

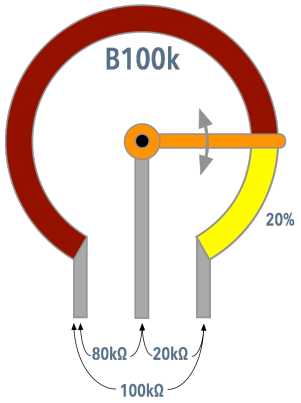
Nesta seção apresentaremos alguns componentes que possuem resistência que varia. A variação pode ser controlada manualmente (potenciômetros) ou devido à influência externa, de luz, calor, ou outros fatores (sensores).

No kit temos potenciômetros de diversos valores e dois sensores resistivos: O **LDR – Light Dependent Resistor**, que varia com a luz, e o **termistor**, que varia com a temperatura.

2.7.1. Potenciômetros

Potenciômetros são **resistores variáveis**. Têm três terminais. A resistência entre os dois terminais mais distantes é fixa, mas o terminal do meio desliza sobre a resistência interna, permitindo obter um valor variável entre cada terminal.

Um potenciômetro pode ser usado como um **divisor de tensão variável**. Se apenas o terminal do meio e um dos laterais for usado, o potenciômetro se comporta como um **resistor variável**. Pode ser utilizado para controlar volume, fazer dimmers, etc.



Símbolo de um potenciômetro:

Potenciômetros identificados com a letra **B** são lineares, ou seja, **variam linearmente** (em um potenciômetro de 100k, 20% é 20k, 40% é 40k). Existem também potenciômetros que variam de forma **exponencial**, que são identificados com a letra **A** (em um potenciômetro de 100k, um giro de 50% corresponde a 20k).

Usaremos potenciômetros no experimento a seguir para variar a corrente que passa em um LED, alterando o seu brilho.

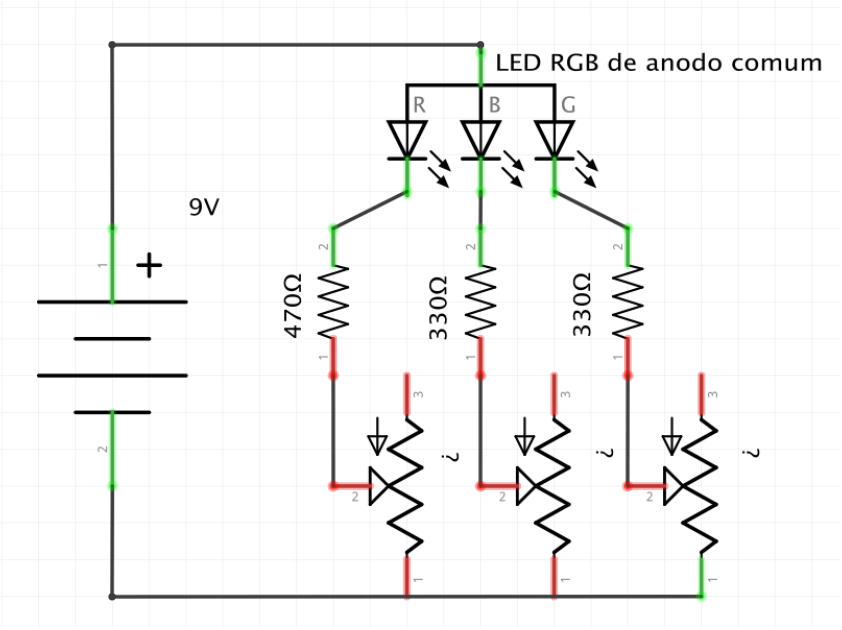
Experimento 8 – Variando as cores de um LED RGB

O objetivo é acender um LED RGB (são três LEDs em uma única embalagem). Este experimento também pode ser feito com três LEDs separados.

Material:

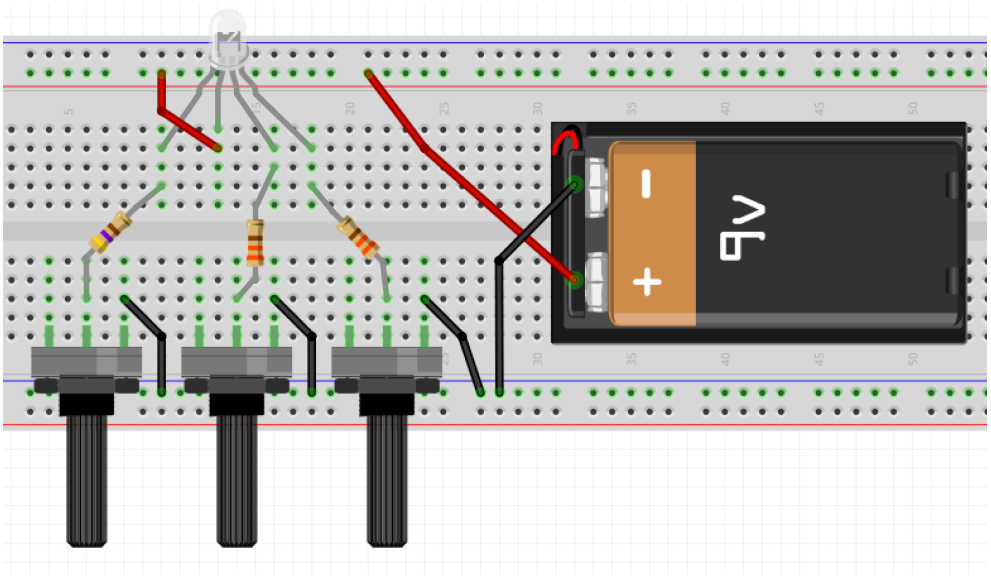
- 1 LED RGB de anodo comum (veja detalhes na referência no final da apostila)
- Fonte de 9 (ou 12 V)
- 2 resistores de 330 ohms e 1 resistor de 470 Ω (para fonte de 9V), ou 2 resistores de 470 Ω, e 1 resistor de 560 Ω (para fonte de 12V)
- Três potenciômetros de 100k Ω (no kit não há três iguais – use um de 100k para o LED vermelho, 50k para o LED verde e 20k para o LED azul)
- Protoboard
- Fios ou jumpers

Monte o esquema abaixo:



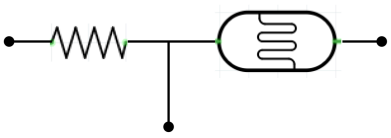
O resistor em série com o potenciômetro é importante, pois quando o potenciômetro estiver no valor mínimo de resistência (zero ohms), deve ainda haver uma resistência limitando a corrente no LED.

A ilustração a seguir contém uma possível montagem do circuito em um protoboard.



2.7.2. LDR – sensor de luz

No kit há dois LDRs. Um de **7mm** e outro de **5mm**. O de 7mm é um pouco mais sensível. Estes LDRs apresentam baixa resistência em ambientes iluminados (tipicamente 50 a 100 ohms em uma sala iluminada, ou menos de 50 ohms em luz do sol direta), e alta resistência em ambientes escuros (tipicamente 100-500k em uma sombra, a 1M ohm em uma sala escura).



O LDR pode ser usado como um resistor variável usando os dois terminais, ou como um divisor de tensão, escolhendo um resistor fixo para ligar no positivo ou negativo (ilustrado acima).

Além do LDR, existem outros componentes no kit que reagem a luz. O **foto-transistor TIL-78** tem a mesma embalagem que um LED translúcido, e reage à luz visível ou infravermelha aplicada **diretamente** à sua lente. Ele não altera a resistência, mas se comporta como uma chave liga-desliga. A célula fotovoltaica de silício gera até 0,5V de tensão quando recebe luz direta do sol.

Você pode substituir os potenciômetros usados no experimento anterior por LDRs, e perceber variação nos LEDs devido à luz ambiente.

2.7.3. Termistor – sensor de temperatura

O termistor incluído no kit reduz sua resistência com o aumento da temperatura. Ele apresenta uma resistência de 10k ohms em temperatura ambiente (25 graus Célsius). Veja na referência no final desta apostila para uma estimativa da resistência em várias outras temperaturas.

O termistor pode ser configurado da mesma forma que o LDR, como resistor variável, ou em um circuito divisor de tensão. Ele é mais eficiente para medir temperaturas elevadas. Com 100 graus ele tem aproximadamente 500 ohms de resistência. Pode ser usado para acionar circuitos que disparam uma ação quando a temperatura atinge um certo nível (ex: controlar a fervura de água).

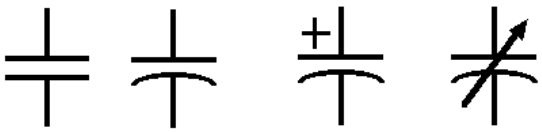
Além do termistor, existe um outro componente incluído no kit que mede temperatura com mais precisão: o circuito integrado **LM35**. Ele tem três terminais e não varia resistência com a temperatura, mas a tensão entre os terminais. Ele será usado em experimentos mais adiante.

2.8. Capacitores

Capacitores são componentes que **acumulam carga elétrica**. Em circuitos onde flui corrente contínua, um capacitor age como um **circuito aberto**, impedindo a passagem de corrente, mas durante transições (ex: quando o circuito é ligado ou desligado) ou quando flui corrente alternada, o capacitor age como **um condutor**, deixando passar a corrente.

No esquema, um capacitor é representado por duas placas separadas. Alguns capacitores têm polaridade e precisam ser usados no circuito respeitando essa polaridade (o + ligado ao positivo e o – ligado ao negativo).

Os símbolos abaixo são usados para capacitores:



A capacidade de carga de um capacitor é medida em **farads**, em homenagem ao cientista inglês **Michael Faraday**. Em geral usamos bilionésimos ou milionésimos de farad em nossos circuitos, às vezes ainda menos que isto. Portanto os capacitores que usaremos são representados em microfarads (**µF**) (1/1000000), nanofarads (**nF**) ($1/10^9$) e picofarads (**pF**) ($1/10^{12}$).

Os capacitores **eletrolíticos** são polarizados, e têm valores maiores. A identificação deles é impressa na embalagem. Já os capacitores de menor valor, **cerâmicos** e de **poliéster**, têm um código para representar o valor baseado em pF. O código é semelhante ao dos resistores, mas sem as cores. São três dígitos. Os dois primeiros representam dígitos do valor, e o terceiro o número de zeros. Por exemplo:

- **103** = 1, 0, 000 = 10000pF = 10kpF = **10nF**
- **474** = 4, 7, 0000 = 470000pF = 470kpF = **470nF**
- **225** = 2, 2, 00000 = 2200000pF = 2200kpF = 2200nF = **2,2 µF**

Um capacitor acumula carga assim que recebe um pulso de corrente elétrica. Se um capacitor está ligado diretamente a uma fonte de tensão contínua, ele recebe sua carga quase instantaneamente,

assim que a bateria for ligada no capacitor. A descarga, curto-circuitando os terminais do capacitor também é praticamente imediata.

Para limitar o tempo de carga ou descarga de um capacitor, usa-se um resistor. O valor do resistor multiplicado pelo valor da capacitância corresponde ao tempo em segundos que leva para um capacitor totalmente descarregado atingir 63% de sua carga. Cinco vezes esse tempo corresponde à carga total do capacitor.

Portanto, a **constante de tempo** para um circuito formado por um resistor R e um capacitor C é

```
t = RC
```

E o tempo de carga é:

```
5 x RC
```

Por exemplo, para um capacitor de 100 µF (0,0001 F) em série com um resistor de 10k ohms tem uma constante de tempo de:

```
t = 0,001 x 10000 = 1 segundo
```

E o capacitor levará

```
5 x 1 = 5 segundos
```

para carregar (ou descarregar) completamente.

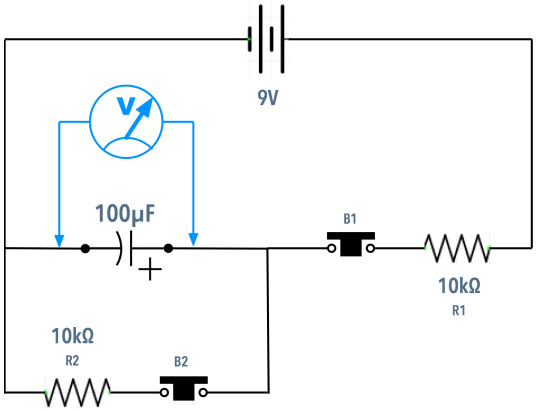
Experimento 9 – Carga e descarga de capacitores

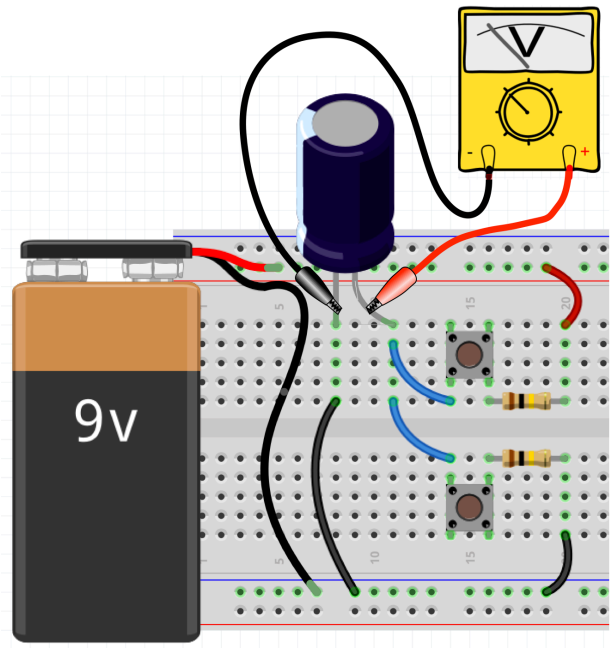
Este experimento demonstra o efeito da carga e descarga em um capacitor.

Material necessário:

- 1 capacitor eletrolítico de 100µF
- Capacitores eletrolíticos de 1000µF, 470µF, 47 e 10µF
- 2 resistores de 10k Ω
- Resistores de 1k e 100k Ω
- Voltímetro
- Bateria ou fonte de 9V
- Duas chaves tácteis (botões de pressão)
- Protoboard, jumpers e fios, garras jacaré

Monte o circuito abaixo. O desenho das chaves no protoboard (de 4 terminais) podem estar diferentes das que você tem no kit (de 2 terminais). Na dúvida use o esquema como referência para as conexões.





Antes de iniciar, meça a tensão da bateria, para saber o valor máximo de tensão que poderá ser carregado pelo capacitor. Depois prenda o multímetro nos terminais do capacitor com garras jacaré, tendo o cuidado para não deixar que encostem uma na outra.

Apertando o botão **B1** o circuito é fechado e a bateria começa a **carregar** através do resistor **R1**. Segure o botão por uns 5 segundos ou até que o multímetro indique a tensão da bateria. Agora solte o botão e perceba que a carga diminui muito lentamente (ela está vazando pelo voltímetro, que tem uma resistência muito alta).

Agora aperte o botão **B2**, que **descarrega** o capacitor através do resistor **R2**. Como os resistores são iguais, o tempo de carga e descarga é semelhante ao da carga. Experimente trocá-los por valores diferentes. Troque também o capacitor de 100µF por capacitores maiores (470 e 1000 µF) e menores (47 e 10µF), e observe o resultado.

Para carregar ou descarregar rapidamente ligue as chave correspondente diretamente ao positivo ou negativo sem usar resistor, ou use resistores de valores baixos (100 Ω).

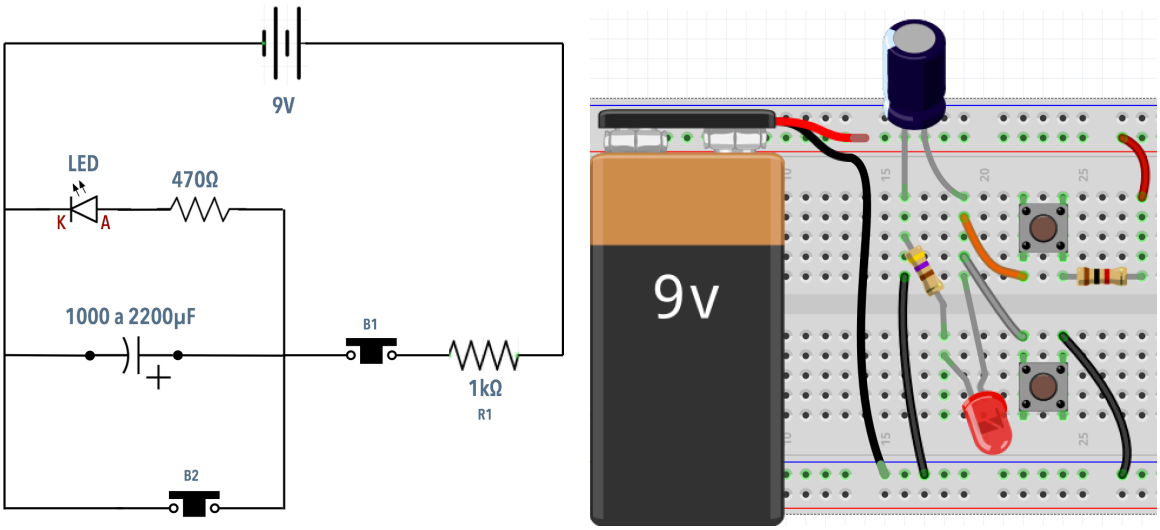
Alteração 9.1 – Usando a carga do capacitor para acender um LED

Material adicional:

- Um LED de qualquer cor
- Resistor de 470 Ω
- Capacitor de 2200 µF

Experimente ligar um LED em paralelo com o capacitor de 1000µF e veja como ele se comporta durante os estágios de carga e descarga (lembre-se que o LED não pode ser ligado diretamente; ele sempre precisa ter um resistor em série para limitar a corrente.)

Veja uma possível solução no circuito abaixo:



Nesta configuração, o botão B2 descarrega o capacitor imediatamente através do fio. Mas ele também irá descarregar um pouco mais lentamente através do LED e resistor de 470 Ω. Se você um resistor maior para limitar a corrente do LED (1k Ω) ele brilhará menos, mas ele também permanecerá aceso por mais tempo já que a resistência maior irá retardar a descarga do capacitor.

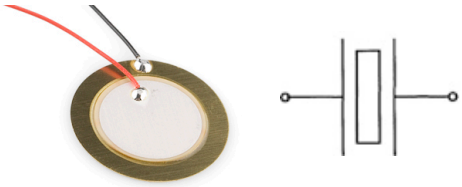
Experimente trocar o capacitor de 1000µF por um capacitor de 2200µF e troque o resistor R1 por uma **ligação direta** (para que a carga do capacitor seja imediata). Experimente colocar os dois capacitores de 1000µF e 2200µF **em paralelo**. O que acontece?

Carregue o capacitor totalmente, depois desligue a bateria do circuito. Por quanto tempo o LED ainda permanece aceso?

Capacitores são bastante usados em circuitos eletrônicos, para acumular tensão, gerar pulsos, configurar temporizadores, retificar corrente alternada, proteger circuitos de sobretensão, isolar sinais, etc. Eles serão usados em vários outros experimentos.

2.9. Célula piezoelétrica

A célula piezoelétrica pode ser usada como sensor de impacto e deformação, dentro de um circuito elétrico, ou como para geração de energia elétrica.



Experimente ligar um LED entre os terminais da célula piezoelétrica e bater rapidamente no centro dela. Isto causa uma **mini-deformação** na cerâmica que gera energia suficiente para acender o LED com um pulso. Células piezoelétricas são usadas como sensores para construir instrumentos musicais sensíveis à velocidade e intensidade do toque. Também são usadas em sapatos para acender LEDs com o impacto, detectar quando alguém bate em uma porta, e até mesmo instalados debaixo do chão em locais de grande movimento para gerar e acumular eletricidade.

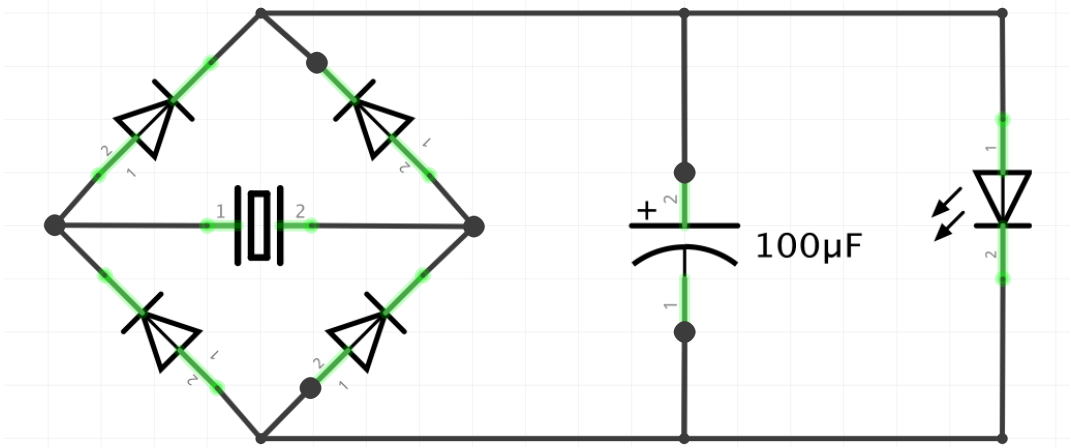
Experimento 10 – Gerador piezoelétrico

Este experimento demonstra a carga de um capacitor e o uso de uma célula piezoelétrica que gera energia através de movimento (deformação e impacto).

Material necessário:

- Célula piezoelétrica (pode ser necessário soldar terminais na placa e sensor; ou prender os fios usando mini-pegadores de roupa – não use garras jacaré pois a célula é frágil)
- Capacitor de 100 µF
- 4 diodos de propósito geral (1N4148 ou equivalente)
- Protoboard, fios e jumpers
- Voltímetro

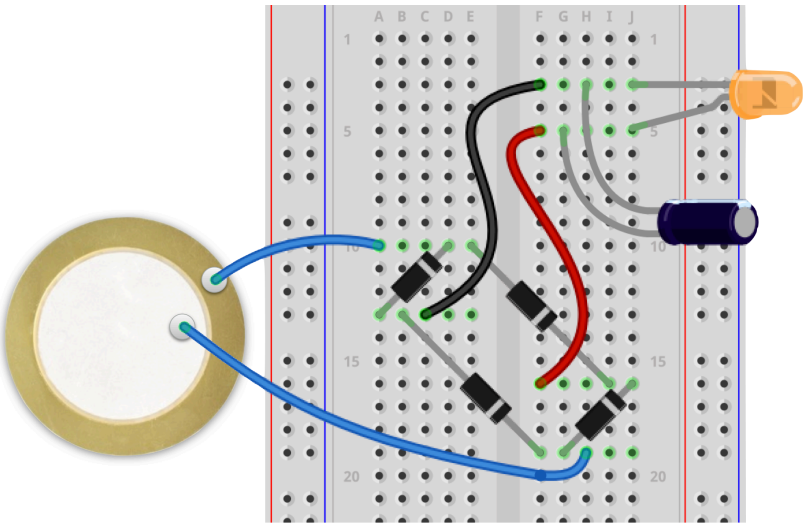
A célula piezoelétrica gera **energia alternada**: produz corrente em um sentido quando contrai e no sentido inverso quando expande. Como o LED é polarizado, ele só acende em um desses pulsos. A ponte com quatro diodos inverte os pulsos negativos permitindo aproveitar todo o ciclo (o LED acende duas vezes mais). O capacitor acumula carga, permitindo que o LED fique aceso sem pulsar, (mas no início ele vai demorar para acumular tensão suficiente para acender o LED). Esse tipo de circuito é chamado de **retificador**, e é usado também em fontes que são ligadas em tensão alternada para gerar tensão contínua (como a fonte de 9V usada nos experimentos).



O capacitor é opcional. Sem usar um capacitor, o LED brilha imediatamente, a cada impulso de compressão ou expansão gerado pela deformação do sensor (batendo no sensor com o dedo, por exemplo, ou apertando-o).

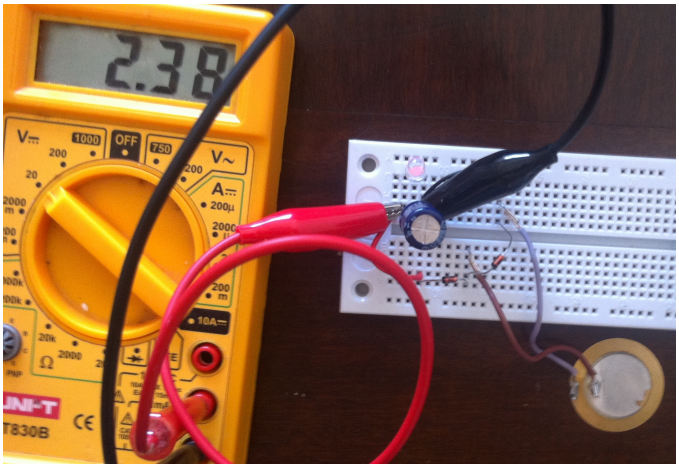
Usando um capacitor, é preciso primeiro acumular carga suficiente para que haja tensão suficiente para acender o LED. Pode ser necessário espremer muitas vezes o sensor para que o capacitor atinja a tensão mínima para acender o LED. Quanto maior o capacitor, mais tempo levará para acumular a

carga (e para descarregar depois). Cada impulso gera um pouco de carga. Assim que o LED acumular a carga mínima ele irá acender, mas também irá descarregar o capacitor mais rapidamente.



Tire o LED do circuito e coloque um voltímetro entre os terminais do capacitor, na posição de 20V. A cada impulso você verá que alguns milivolts se acumularem no capacitor. Quando houver aproximadamente 2 volts, coloque um LED vermelho que ele acenderá por alguns segundos. Você também pode deixar o LED no circuito enquanto acumula carga. Enquanto não houver tensão direta suficiente para acendê-lo, o consumo de corrente será baixo, e a carga subirá mais rapidamente.

Um LED rosa acenderá quando a tensão chegar a 2,5V aproximadamente. Na foto abaixo a luz começa a aparecer com 2,38V:



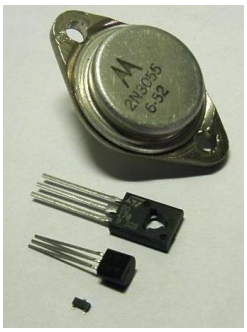
Depois de aceso, o LED rapidamente consome a corrente, diminuindo a tensão no capacitor, mas não totalmente. Mais alguns toques e ela sobe de novo. Portanto, se a fonte de pulsos for contínua (ex: se os sensores estiverem instalados em um sapato e a pessoa estiver andando, saltando ou dançando), o capacitor manterá uma luz constante no LED. Você pode usar um resistor em série com o LED, que irá atrasar um pouco a descarga, mas também irá fazer com que ele demore mais a acender.

Se você usar várias células fotoelétricas retificadas **em paralelo**, o capacitor carregará mais rápido, e o LED brilhará mais forte. Uma possível aplicação é instalar duas ou mais células piezoelétricas em sapatos para iluminar roupas e acessórios usados para correr ou dançar. O impacto no chão gera energia, que se acumula, e pode ser usado para acender LEDs distribuídos pelas roupas.

Você também pode preferir usar o circuito *sem* o capacitor, que acende o LED mais rapidamente e gera pulsos curtos (mas intensos) de energia apenas na hora do impacto.

Este link contém a demonstração de um sapato gerando energia piezoelétrica para acender LEDs: https://www.youtube.com/watch?v=lDBhMCFZfv0

3. Transistores



Transistores são os componentes eletrônicos de estado sólido responsáveis pela revolução eletrônica no século 20. Eles substituíram as válvulas (tríodos) que antes eram usadas em rádios e TVs para amplificar sinais, e permitiram a miniaturização que possibilitaram as viagens espaciais e computadores.

O transistor foi inventado em 1948 por John Bardeen, Walter Brattain e William Shockley, que receberam o prêmio Nobel de física em 1956 pelo invento.

Assim como os diodos e LEDs, transistores são feitos de junções de materiais semicondutores. Os primeiros transistores eram feitos de ligas de germânio (Ge). Os transistores modernos são feitos principalmente de Silício (Si), que é um dos minerais mais abundantes na natureza. Os modelos mais populares consistem de duas junções de três ligas de Silício, que tem suas propriedades físico-químicas alteradas através da adição de impurezas (outros semicondutores).

Hoje existem vários diferentes tipos de transistores. Os modelos mais antigos ainda são usados e são geralmente comercializados em embalagens plásticas ou metálicas de três terminais. Chegam a custar em torno de 10 centavos. Mas nos equipamentos eletrônicos modernos, como smartphones e computadores, transistores existem aos bilhões dentro dos chips ou circuitos integrados. Consomem pouquíssima corrente e são minúsculos (da ordem de milésimos de milímetro).

Utilizaremos nestes experimentos os **transistores bipolares de junção do tipo NPN**. Eles são antigos, mas são baratos, fáceis de encontrar e mais robustos que os **transistores de efeito de campo** (MOSFETs), mais modernos e que consomem muito pouca energia. Mas o kit também inclui MOSFETS, usados em alguns experimentos com Arduino, e transistores bipolares de junção PNP (que têm polaridade inversa).

Transistores bipolares têm três terminais que se chamam base (**B**), emissor (**E**) e coletor (**C**). A **base** de um transistor bipolar funciona **como uma torneira** que controla o fluxo de **corrente do coletor para o emissor**, que geralmente é bem maior (100 vezes ou mais) que a corrente na base. Esse fator de amplificação é chamado de **ganho do transistor** (também chamada de beta – β ou hFE).

Existem duas formas básicas de usar um transistor. Como **amplificador** ou como **chave**.

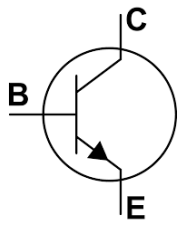
Como amplificador ele é configurado para operar em uma faixa que amplia a corrente na base de maneira mais ou menos **linear**. Se usarmos a analogia da torneira, operando como amplificador ela nunca seria totalmente fechada ou aberta, mas operaria apenas controlando a intensidade do fluxo.

Como chave, o transistor trabalha em estados **extremos**, totalmente ligado (em **saturação**) ou desligado (em **corte**). A torneira ou está completamente aberta (corrente na base suficiente para saturar o transistor) ou completamente fechada (nenhuma corrente ou corrente negativa na base). Em resumo, o transistor se comporta exatamente como uma chave entre seus terminais C e E, que é aberta ou fechada pela corrente aplicada no terminal B. Quando o transistor está desligado (sem corrente na base), a tensão entre C e E é máxima, e a corrente é zero. Quando ele está saturado a tensão entre C e E é zero e a corrente entre C e E é máxima.

Além do uso como amplificador e chave, transistores também podem ser usados para construir **osciladores** (geradores de ondas e pulsos) amplificando ciclos de carga e descarga de circuitos RC (resistor-capacitor) e **realimentando**-o na sua entrada.

3.1. Transistores bipolares de junção NPN

O funcionamento que descrevemos acima para o transistor refere-se a transistores bipolares de junção NPN, que é apenas um dos tipos de transistores usados, mas é o único que iremos explorar nesta seção., O símbolo desse tipo de transistor está ilustrado abaixo. É preciso identificar os terminais na embalagem plástica do componente. Veja na referência no final desta apostila. Ela varia de acordo com o modelo (há muitos) e o fabricante.



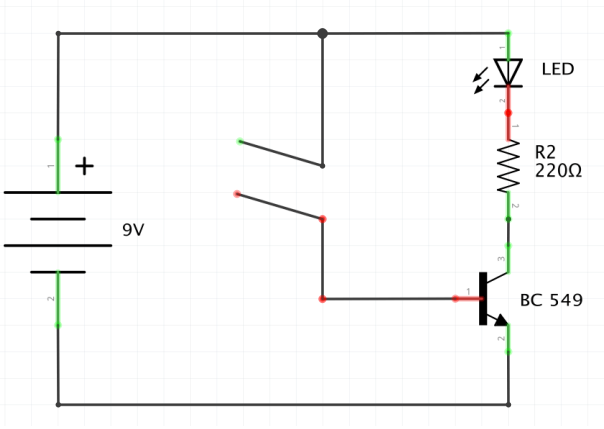
B = base (controle, baixa corrente + ou -), **C** = coletor (+), **E** = emissor (-)

Experimento 11 – Transistores: circuito básico

Para demonstrar o funcionamento básico do transistor, monte o circuito abaixo.

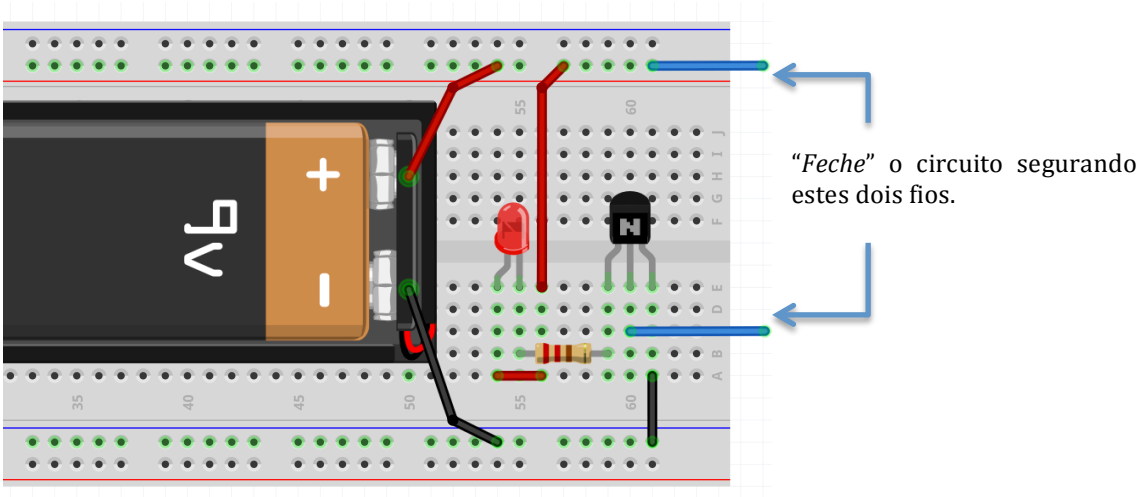
Material necessário:

- 1 transistor BC 549 (ou equivalente)
- 1 resistor de 220 Ω (ou 330 Ω, se a fonte for de 12 V)
- 1 LED
- Protoboard
- Fios e jumpers
- Fonte de 9V (ou 12V)



Você fará a ligação da base com o terminal positivo **segurando nas duas pontas dos fios**. Como a sua resistência é muito alta, a corrente que irá fluir pela base será baixíssima.

Verifique todas as conexões e ligue a fonte de 9V **por último**. Os terminais separados **não devem tocar** em hipótese alguma. Mantenha-os separados no protoboard.



Ao fechar o circuito, uma pequena corrente (da ordem de microampères) irá circular pela base, abrindo (bastante) a "torneira" do transistor e permitindo a passagem de uma grande corrente entre C e E, suficiente para acender o LED. Não seria possível acender o LED com a corrente que passa na base apenas.

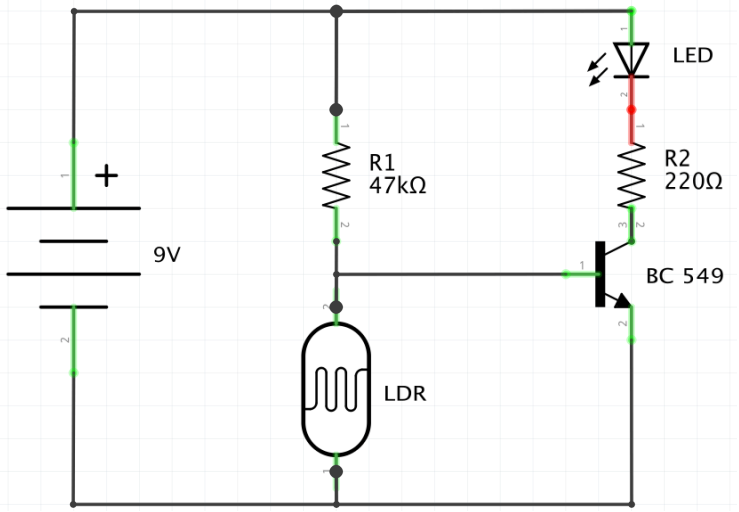
Na verdade, o circuito é tão sensível que o LED poderá acender até mesmo antes que você toque nos fios. Normalmente usamos circuitos com um divisor de tensão na base, para que se tenha maior controle sobre a corrente da base evitando **interferências**. Se houver interferência, experimente conectar um resistor de 10M ou mais ligando a base ao negativo. Neste caso, o LED acenderá quando a resistência que liga a base ao positivo for **maior** que a resistência que a liga ao negativo. O próximo experimento usará um divisor de tensão na base.

Experimento 12 – Luz de emergência com transistor

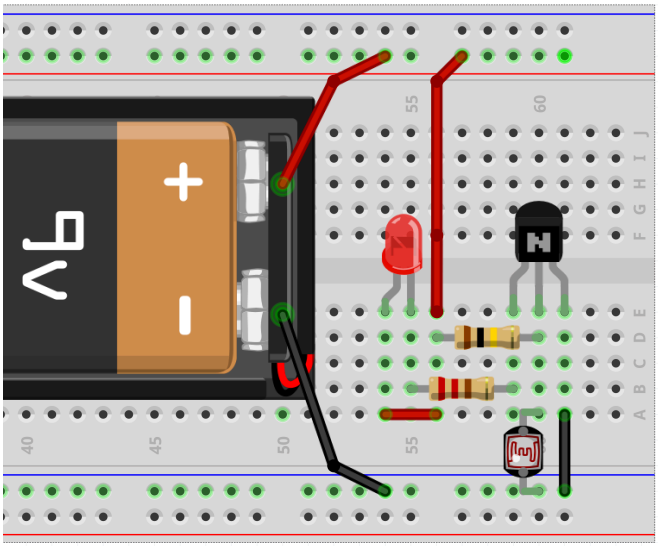
Este circuito fará um LED acender quando o ambiente estiver escuro, e apagar quando estiver claro.

Material necessário:

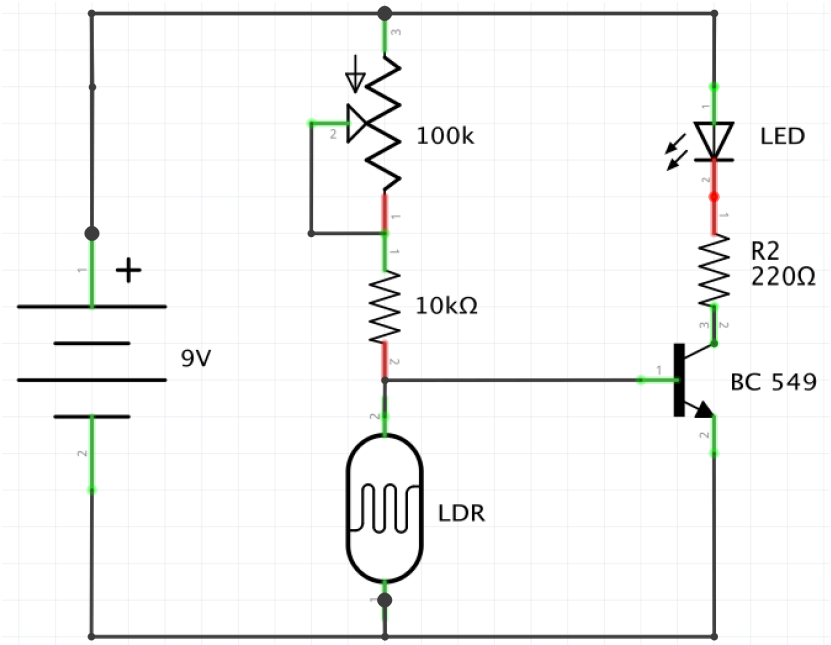
- Bateria ou fonte de 9V (ou fonte de 12V)
- LED
- Transistor NPN de propósito geral (BC549 ou equivalente)
- Resistor de 220 Ω (ou 330 Ω, se a fonte for de 12V)
- Resistor de 47k Ω, 100k Ω ou potenciômetro de 100k Ω + resistor de 10k Ω (valor vai depender da sensibilidade do LDR e luminosidade da sala)
- LDR de 7mm (pode-se usar o de 5mm, mas será preciso ajustar a sensibilidade)
- Protoboard, fios e jumpers

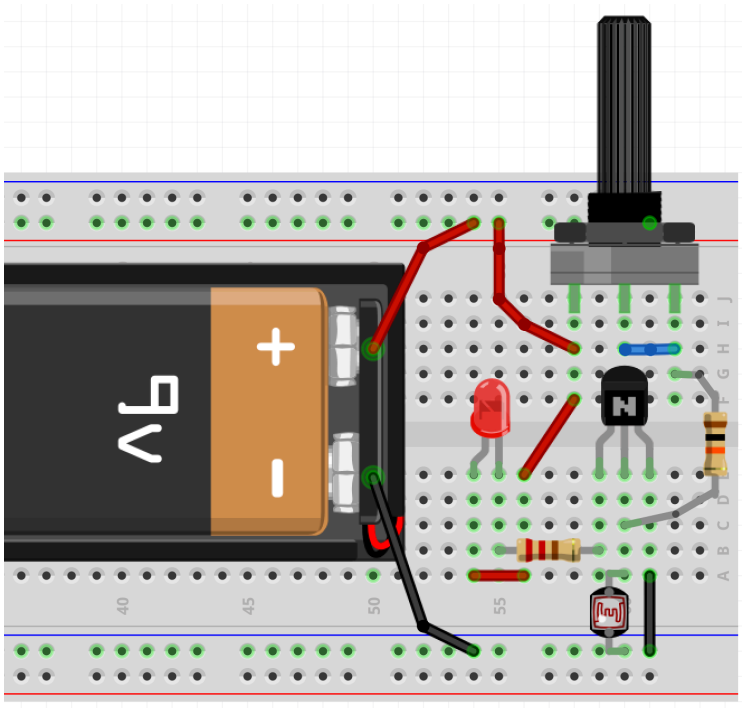


Pode ser necessário ajustar o resistor **R1**, dependendo da sensibilidade do **LDR** e da luz do ambiente. Com a mesma quantidade de luz, o LDR de 7mm terá uma resistência maior que o de 5mm, então se for usado o de 5mm, pode ser necessário uma resistência maior (ex: 100k Ω) para que o circuito funcione igual. No desenho do protoboard está sendo usado um resistor de 100k Ω:



Uma outra alternativa é substituir R1 por um resistor de 10k Ω em série com um potenciômetro de 100k, para permitir o **ajuste fino** dependendo das condições de luz do ambiente.





3.2. Fototransistor

O fototransistor é um transistor especial cuja base é ativada por luz. Se não houver luz o transistor desliga. Se houver luz **diretamente aplicada na lente**, ele liga e deixa passar corrente entre E e C. O fototransistor é acionável por luz visível e também por luz infravermelha. Fototransistores são frequentemente usados em controles remotos.

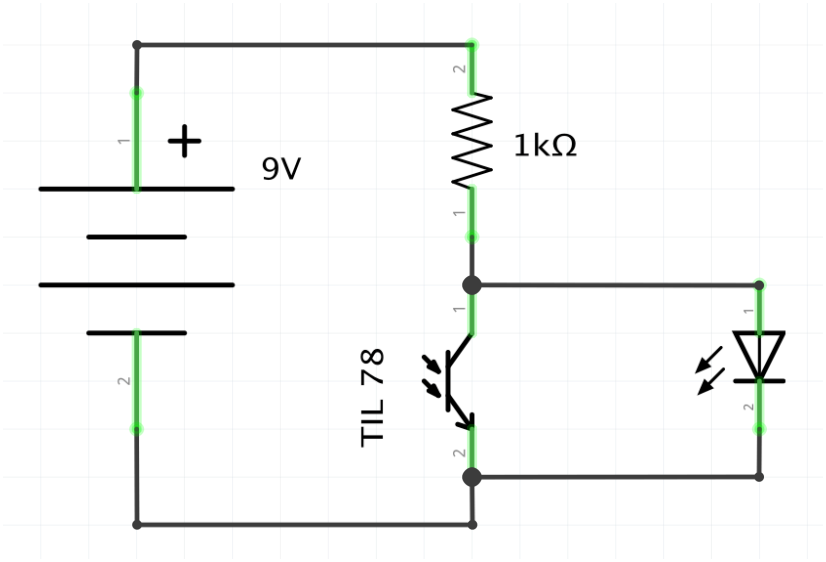
Experimento 13 – Fototransistor que desliga a carga ao ser ativado

Este circuito tem um comportamento similar ao circuito com LDR. É mais simples e mais sensível, mas o foto-transistor requer que a luz seja aplicada diretamente na sua lente.

Material necessário:

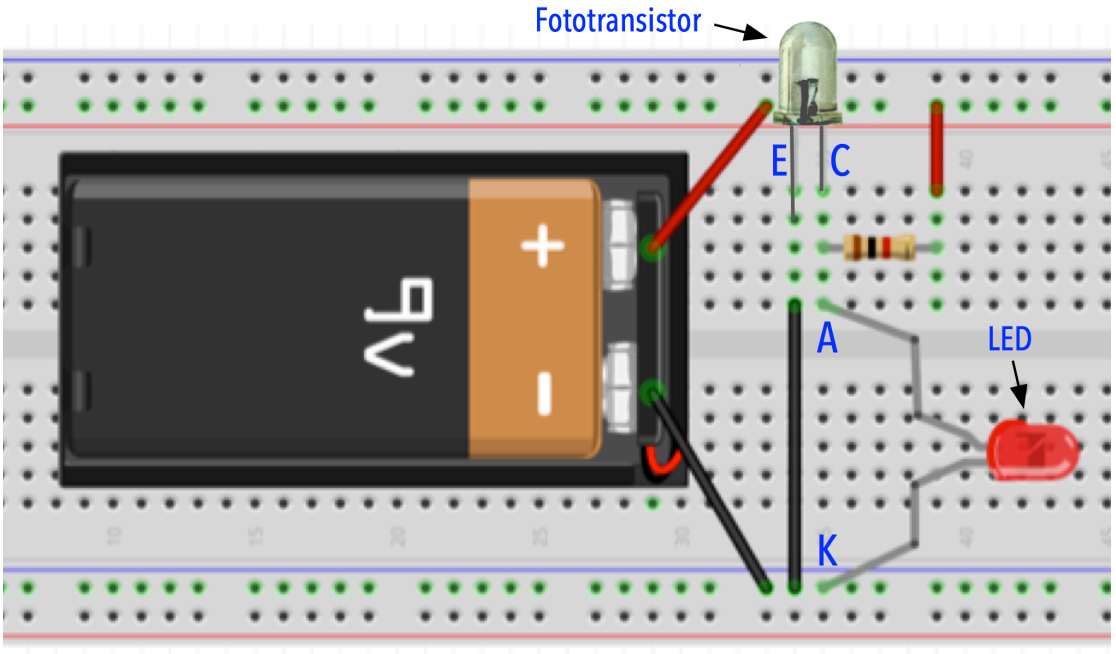
- Fonte de 9V (ou 12V)
- Fototransistor TIL 78 (tem a **mesma embalagem que um LED** – veja referência no final desta apostila para detalhes).
- Resistor de 1k Ω
- LED
- Protoboard, fios/jumpers

A embalagem do fototransistor é transparente e igual à de um LED. A perna mais longa é o emissor (**E**) e a mais curta é o coletor (**C**). No circuito abaixo o emissor (perna mais longa) deve ser conectada ao negativo e ao catodo (perna mais curta) do LED.



Quando o transistor liga, ele faz um curto-circuito nos terminais do LED, apagando-o. Se não houver luz no fototransistor, ele se comporta como um circuito aberto, e toda a corrente irá fluir pelo LED. Para apagar o LED, aplique luz diretamente sobre a lente do fototransistor (ex: use um outro LED ou uma lanterna de celular). Teste também apontando um **LED infravermelho** (não foi incluído no kit) que emite luz invisível.

O LED pode ser substituído por outro circuito, que será desligado quando o fototransistor receber luz. Neste circuito, você pode também substituir o LED por uma **cigarra** (buzzer).



Como você alteraria este circuito para que ele tenha o comportamento inverso, e acenda o LED (ou acione a cigarra) somente quando houver luz?

Não desmonte este circuito pois utilizaremos em outro experimento.

3.3. Osciladores

Um oscilador é um circuito que automaticamente alterna seu estado. É possível usar transistores para construir um oscilador realimentando a sua entrada com o sinal produzido na saída. Osciladores permitem gerar sinais de corrente alternada, produzindo ondas em vários formatos, permitindo a construção de sirenes e pisca-piscas. Também são usados em circuitos transmissores e receptores de sinais de rádio e muitas outras aplicações.

Existem várias formas de construir um oscilador. Todos envolvem realimentação de sinais, e não são muito simples de entender, mas são circuitos fundamentais na eletrônica, produzem resultados interessantes e portanto vale a pena construir alguns.

Experimento 14 – Pisca-pisca alternado com LEDs

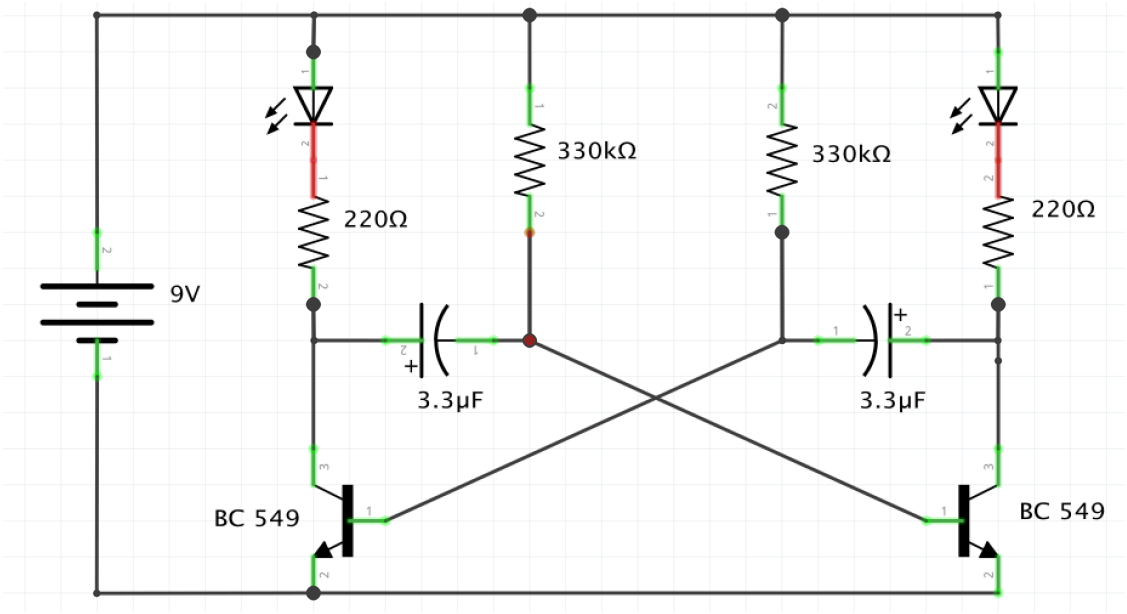
Usando capacitores e podemos acionar automaticamente a corrente na base dos transistores a partir da carga do capacitor. Combinando dois circuitos desse tipo fazemos um circuito oscilador que é chamado de **gangorra**, ou "**multivibrador astável**". É formado por dois circuitos amplificadores simétricos. A entrada (base) de um é realimentada pela saída (coletor) do outro.

Neste experimento usaremos uma combinação de capacitores e resistores para produzir uma frequência que permite que os LEDs pisquem de forma alternada.

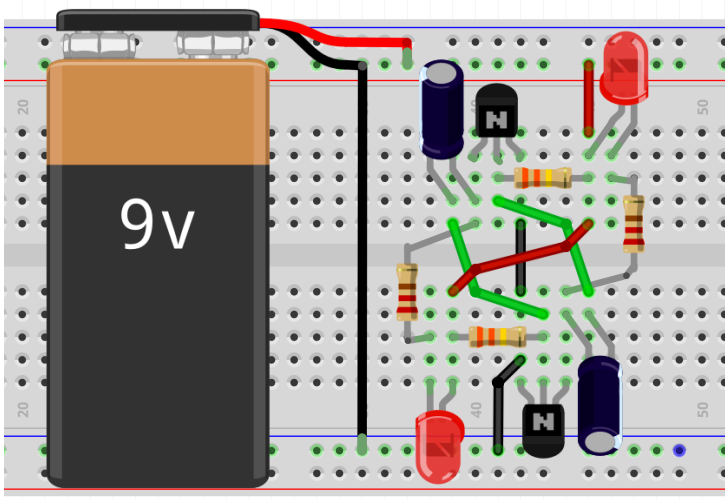
Material necessário:

- Bateria de 9V, fonte de 9V (ou de 12V)
- 2 resistores de 220 Ω (use 330 Ω, se fonte for de 12V)
- 2 LEDs
- 2 resistores de 330k Ω
- 2 capacitores eletrolíticos de 3,3 µF
- 2 transistores NPN de propósito geral (BC 549 ou equivalente)
- Protoboard, fios e jumpers

Monte o circuito abaixo prestando atenção nas conexões:



A ilustração abaixo mostra uma montagem possível que pode ser feita no protoboard:



Você pode variar os valores dos resistores da base (não use menos que 1k Ω) e dos capacitores para obter tempos e carga e descarga diferentes.

Não se limite ao protoboard! Tente montar o circuito usando o esquema. A fotografia ao lado mostra o mesmo circuito oscilador explorado neste experimento construído usando fita de cobre sobre uma folha de papel.

O tempo que cada LED fica aceso pode ser calculado usando a fórmula

```
0,7 x R x C
```

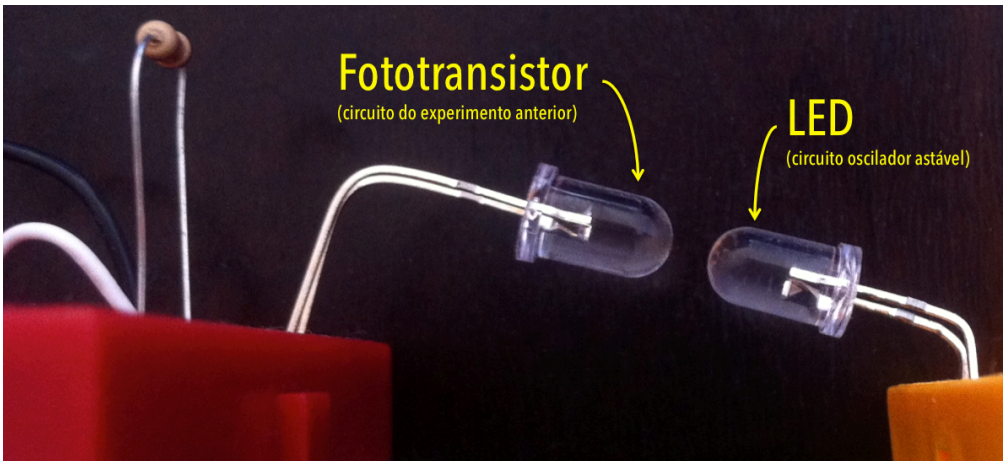
Onde R e C são o resistor e o capacitor ligados a base do transistor. No nosso exemplo os tempos são iguais, com resistor de 330k e capacitor de 3,3uF, então cada pulso dura:

```
0,7 x 330000 x 0,0000033 = 0,8s
```

Veja também as calculadoras online para esse tipo de circuito no final da apostila.

Alteração 14.1 – Acoplador ótico

Experimente posicionar a saída de um dos LEDs diretamente na lente do fototransistor do circuito do experimento anterior, como mostrado na fotografia abaixo.



Isto vai acionar o fototransistor sempre que o LED acender, fazendo com que o LED, que é controlado pelo do circuito do fototransistor, apague e acenda no mesmo ritmo que os LEDs do oscilador astável. Os dois circuitos agora estão sincronizados. O LED e o foto-transistor funcionam nesta configuração como um **acoplador ótico**.

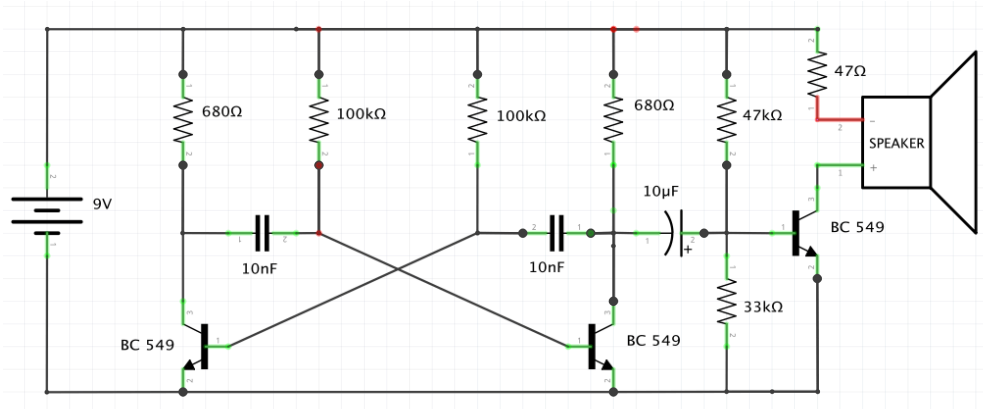
Experimento 15 – Oscilador sonoro ou sirene

Esta é uma adaptação do circuito anterior. A estrutura básica é a mesma, mas foram alterados valores dos capacitores e resistores para que a oscilação ocorra em uma frequência maior e audível. Usando resistores de 100k e capacitores de 10nF, o circuito oscilará em uma frequência próxima de 1500 ciclos por segundo. Nessa frequência, não será possível ver os LEDs piscarem, mas podemos **ouvir** a oscilação. Para isto, acrescentamos **um estágio amplificador** na saída de um dos transistores, ligado a um alto-falante.

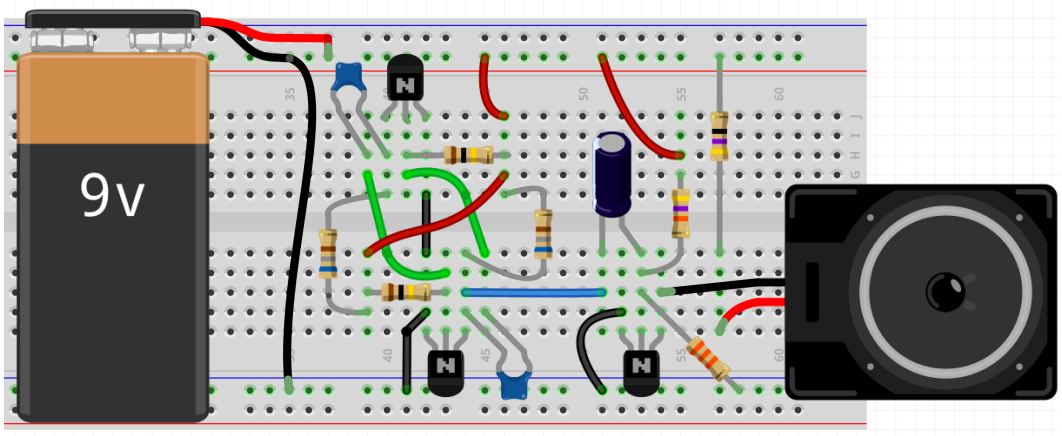
Material necessário:

- Alto-falante de 8 Ω
- 2 resistores de 100k Ω
- 2 resistores de 680 Ω
- 1 resistor de 47k Ω
- 1 resistor de 33k Ω
- 1 resistor de 47 Ω
- 2 capacitores de 10nF
- 1 capacitor eletrolítico de 10 µF
- 3 transistores BC 549 ou equivalente
- Protoboard, fios e jumpers

O circuito está ilustrado abaixo. O módulo amplificador é um circuito separado, que recebe o sinal do oscilador através do capacitor de 10µF, e o amplifica, ativando o alto-falante. O resultado, ao ligar o circuito, deve ser um apito agudo.



A ilustração abaixo mostra uma possível implementação do circuito usando o protoboard.



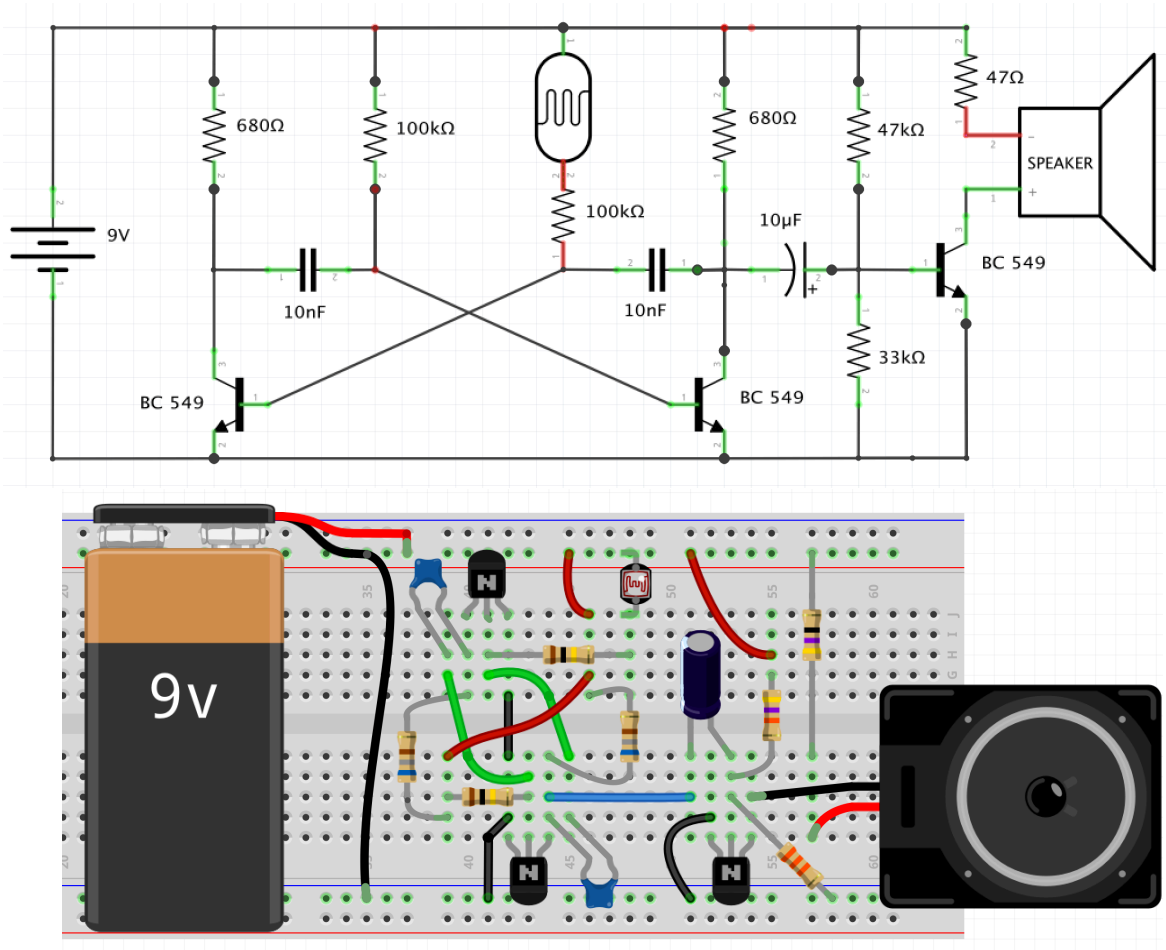
Alteração 15.1 – Um "theremin" sensível a luz

Um *theremin* é um sintetizador de som que produz tonalidades e timbres diferentes a partir de interferência de sensores externos, geralmente magnéticos. Podemos criar um theremin simples para variar a frequência com a luz incluindo um LDR no oscilador sonoro.

Material adicional:

- 1 LDR (sensor resistivo de luz)

Escolha um dos lados do circuito e coloque um LDR **em série** com um resistor de 100k. Isto vai aumentar a resistência e consequentemente o tempo de descarga de um dos capacitores, diminuindo a frequência e fazendo o som ficar mais grave. Mas quando houver luz no LDR, a sua resistência irá cair fazendo com que a frequência fique próxima à do circuito anterior, e o som mais agudo.

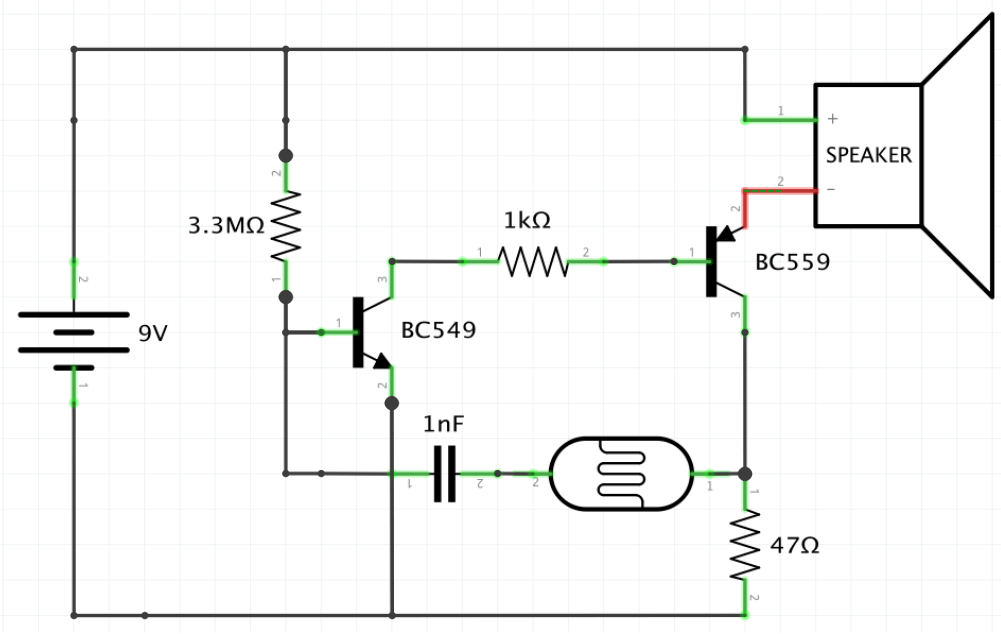


Experimento 16 (extra) – Oscilador sonoro com transistor PNP

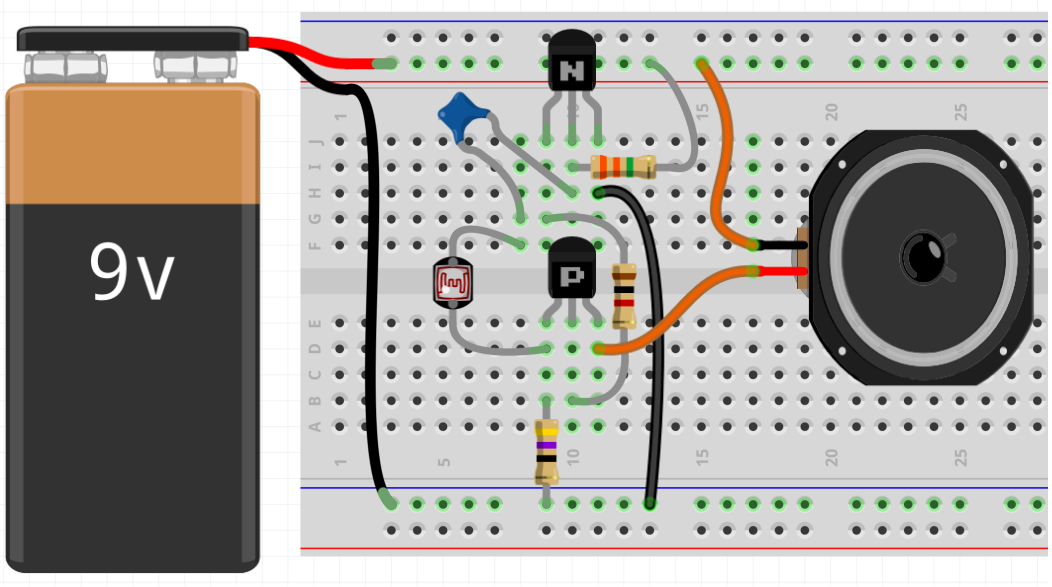
Existem diversas outras formas de construir osciladores com transistores. Nenhum é muito fácil de calcular e controlar. Dependendo da aplicação, o ideal é usar cristais de quartzo (que garante alta precisão e permite frequências elevadas) ou indutores e sistemas que geram ondas naturais (senoidais). Para finalizar este capítulo e usar o transistor PNP disponível no kit, incluímos um circuito oscilador clássico usando um transistor PNP e NPN. Ele gera som cuja frequência varia com a luz. O transistor PNP funciona com polaridade inversa ao NPN.

Material necessário:

- Transistor NPN de propósito geral (BC 549 ou equivalente)
- Transistor PNP de propósito geral (BC 559 ou equivalente)
- Capacitor de 1nF
- Resistor de 1k Ω
- Resistor de 3,3M Ω
- Resistor de 47 Ω
- LDR
- Alto-falante
- Fios, jumpers e protoboard
- Fonte de 9V



Você pode substituir o LDR por outros sensores (ex: de umidade ou temperatura). Pode também variar a frequência alterando resistores e capacitores (mas este circuito não é tão flexível quanto os outros). A ilustração abaixo mostra uma possível implementação com protoboard:



4. Circuitos integrados

Circuitos integrados são componentes eletrônicos que contém circuitos inteiros, formados por dezenas, centenas, milhares, milhões e até bilhões de transistores. Circuitos integrados também são chamados de **chips**. Geralmente eles têm uma aplicação bem definida e diversos terminais usados como entradas e saídas e para configuração. Circuitos eletrônicos que usam circuitos integrados são geralmente mais compactos e mais simples do que os que dependem de transistores e diodos individuais. Em geral, uma mesma funcionalidade é mais fácil de implementar, ou fica mais robusta ou mais segura usando um circuito integrado em vez de transistores individuais.

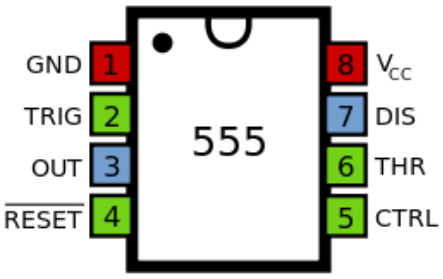
Nesta oficina apresentaremos alguns circuitos integrados baratos e populares. Não se preocupe se não entender completamente o funcionamento deles. É possível construir circuitos e até mesmo fazer alterações nos circuitos sem entender todos os detalhes. Monte os experimentos e depois, se desejar, leia as explicações do seu funcionamento.

4.1. O circuito integrado 555

O circuito integrado 555 é um componente muito versátil. Basicamente funciona como um **temporizador**, que pode ser configurado em um circuito simples contendo um capacitor e alguns resistores. O chip está contido em uma embalagem de plástico de oito pinos, alinhados em paralelo (padrão **DIP** – Dual-In-Line). Dois pinos são usados para alimentá-lo com uma tensão entre 3 e 15V (**GND**: negativo, e **Vcc**: positivo). Os outros pinos são usados como entrada e saída.

Pinos de circuitos integrados DIP são contados **a partir de uma marca que indica o pino 1**: um chanfro ou um ponto. O pino 1 está localizado à esquerda da marca, e a contagem prossegue crescente pelo lado esquerdo, e volta pelo lado direito, de forma que o primeiro e último pinos se localizam na frente do componente, em lados opostos.

O diagrama abaixo descreve os pinos do circuito integrado 555. Os pinos vermelhos (1 e 8) são para alimentação do componente. Os verdes (2, 4, 5 e 6) são entradas, e os azuis (3 e 7) são saídas.



O 555, e muitos outros circuitos integrados que funcionam com **lógica digital**, reconhece na entrada e produz na saída **valores fixos de tensão** que são identificados como **níveis lógicos**. A saída do 555 ou é zero (**nível lógico BAIXO**) ou Vcc (**nível lógico ALTO**), que normalmente é uma tensão maior que zero (tipicamente a mesma tensão usada para alimentá-lo). Os pinos de entrada (em verde) são controlados aplicando neles valores de tensão **relativas** à tensão de entrada (1/3 e 2/3 dessa tensão), para produzir os valores de tensão 0V (nível lógico baixo) ou Vcc (alto) nas saídas.

A tabela abaixo descreve em detalhes o funcionamento de cada pino.

Pino	Nome	Função
1	GND	**0V.** Ligue no polo NEGATIVO da bateria.
2	TRIG	**Disparador.** Um intervalo de temporização inicia quando a entrada neste pino cai **abaixo** de ½ do valor em CTRL (**1/3 de V_{CC}**, se CTRL não estiver sendo usado). Isto faz o valor em OUT ser ALTO. O valor ALTO em OUT será mantido enquanto este pino estiver com tensão baixa.

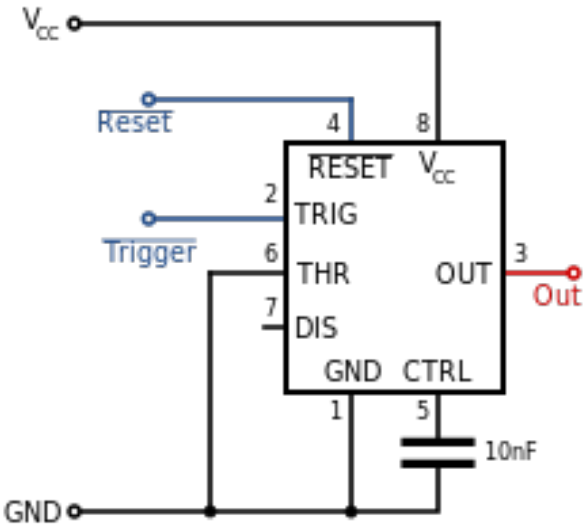
3	OUT	**Saída.** O valor de saída, que pode ser nível lógico ALTO (até 1.7 V abaixo de $+V_{CC}$) ou BAIXO (igual a GND).
4	RESET	**Reset**. Reinicia o intervalo se ligado em GND. Um novo intervalo só inicia novamente se RESET tiver uma tensão de no mínimo 0,7V.
5	CTRL	**Tensão de controle.** Permite estabelecer a **tensão de referência** usada para disparar e limitar o temporizador. Normalmente este pino não é usado (e deve ser conectado ao GND através de um capacitor de 10nF), e neste caso a tensão de referência será sempre de 2/3 de V_{CC}.
6	THR	**Limite.** Quando o nível de tensão aqui for **maior** que em CTRL (**2/3 de V_{CC}**), o valor em OUT será reduzido para zero, terminando o ciclo.
7	DIS	**Chave de descarga.** O pino é ligado temporariamente à GND entre cada intervalo de temporização. Um capacitor conectado aqui será descarregado entre intervalos. O início de novo ciclo fecha a chave que só abre novamente quando o próximo intervalo terminar (quando a tensão em OUT tiver nível lógico BAIXO).
8	V_{CC}	**Fonte de tensão** entre +3 e +15V para alimentar o componente. Ligue este pino no polo POSITIVO da bateria.

Os controles (**entradas**) disponíveis são 4: **CTRL(5), RESET(4), TRIG(2) e THR(6)**. A entrada RESET (pino 4) tem precedência sobre TRIG(pino 2), e TRIG(pino 2) tem precedência sobre THR(pino 6). Isto significa que se RESET estiver acionado (ligado no negativo), os valores de TRIG e THR são ignorados, e se RESET estiver inativo (ligado no positivo), e TRIG for acionado (ligado no negativo), o valor de THR é ignorado. THR só é considerado se RESET e TRIG estiverem ambos inativos (ligados no positivo). A entrada CTRL estabelece a referência usada por TRIG e THR. Em circuitos simples, CTRL é ligada a GND através de um capacitor de 10nF (para eliminar interferências), fazendo com que a referência seja considerada igual à tensão de entrada (Vcc).

As **saídas** são duas: **OUT(3)** e **DIS(7)**. OUT (pino 3) é usado em praticamente todas as aplicações. DIS (pino 7)é ligada ao GND (negativo) todas as vezes que um ciclo termina, e geralmente usada para descarregar um capacitor ligado neste pino.

Tudo isto será mais fácil de entender se fizermos alguns circuitos. A seguir estão três circuitos essenciais com 555. Em todos eles, o pino VCC(8) liga-se ao positivo, o pino GND(1) ao negativo, e o pino CTRL(5) ao negativo através de um capacitor de 10nF. A saída é sempre conectada ao pino OUT(3), e os outros pinos variam conforme o modo usado.

4.1.1. 555 em modo biestável



O circuito biestável é um **alternador de estado**. É uma espécie de memória que guarda um **estado** (ligado/ALTO ou desligado/BAIXO). O estado é uma **tensão**, portanto o estado **ligado** pode ser usado para acender um LED, e o estado **desligado** funciona para apagá-lo. Esta tensão irá aparecer na saída OUT (pino 3). O disparador do alternador é TRIG (pino 2), que liga o circuito. Para reinicializar o processo e voltar ao estado anterior, aciona-se o RESET (pino 4) que reinicializa o processo.

Neste circuito o pino 7 (DIS) não é usado e o pino 6 (THR) é ligado ao negativo. Somente os pinos 2 (disparador) e 4 (reset) controlam o circuito. Os dois devem *iniciar* em estado **ligado**, ou **ALTO** (conectados ao positivo).

Para **iniciar** o ciclo (começa com nível ALTO em OUT), o **pino 2** precisa ser ligado temporariamente ao **negativo**.

Para **terminar** o ciclo (causar nível BAIXO em OUT), o **pino 4** é ligado temporariamente ao **negativo**. A ligação pode ser feita com chaves, botões, sensores, transistores, etc.

Uma técnica para manter os níveis lógicos em estado ALTO é construir um divisor de tensão (entre os polos positivo e negativo da bateria, com o pino ligado no meio e um resistor ligado ao positivo que garanta uma tensão alta o suficiente no pino para mantê-lo desligado (maior que 1/3 de Vcc). Essa técnica é chamada de **resistor de pull-up** já que ela puxa para o ALTO o estado do pino. O estado do pino só mudará quando a tensão cair abaixo do valor mínimo. Isto será provocado pelo sensor, chave, capacitor ou componente que deverá conectar o pino ao negativo, quando sua resistência cair a ponto de fazer a tensão sobre ele cair abaixo do limite de 1/3 de Vcc.

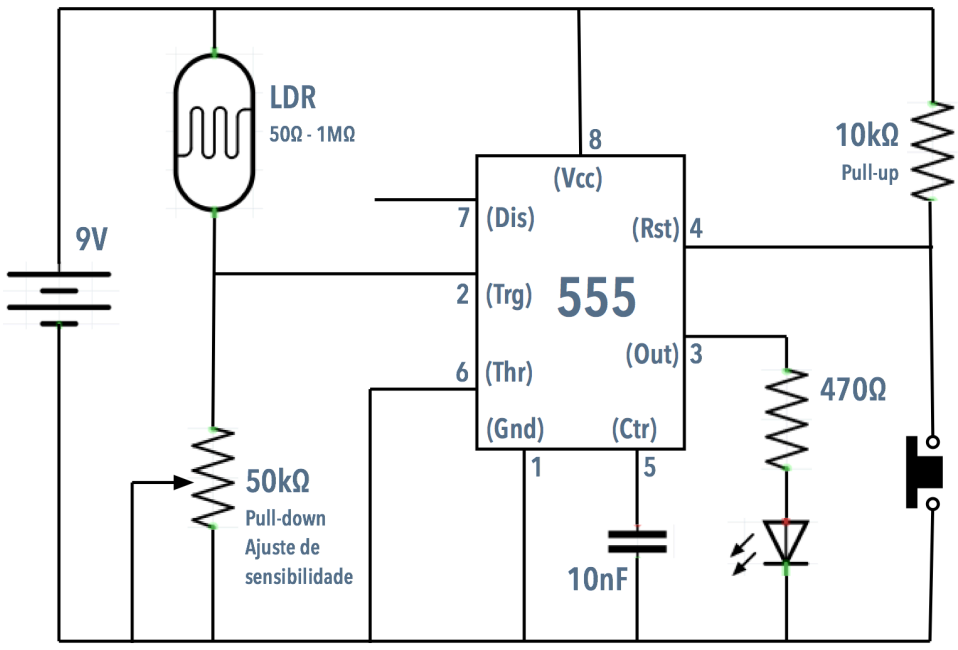
Lembre-se: os pinos de entrada 2 e 4 do 555 são disparados com nível lógico BAIXO (negativo, ou menos de 1/3 de VCC), e são inativos com nível lógico ALTO (positivo, mais e 2/3 de VCC). O pino 6 (THR) dispara com 2/3 ou mais de VCC.

O experimento a seguir ilustra esse comportamento usando a lógica inversa (**resistor de pull-down**): um sensor para disparar o pino 2 quando estiver escuro. Um botão é usado para reiniciar o ciclo.

Experimento 17 – Disparador acionado por pouca luz

Material necessário:

- Fonte de 9 ou 12V, ou bateria de 9V
- Protoboard, fios e jumpers
- LDR de 5 ou 7mm
- Potenciômetro de 50k/100k ou resistor de 22k Ω, 33k Ω, 47k Ω ou 100k Ω (de acordo com a sensibilidade desejada para o LDR)
- Resistor de 10k Ω
- Resistor de 470 Ω
- LED
- Capacitor cerâmico de 10nF
- Chave táctil tipo push-button
- Circuito integrado 555

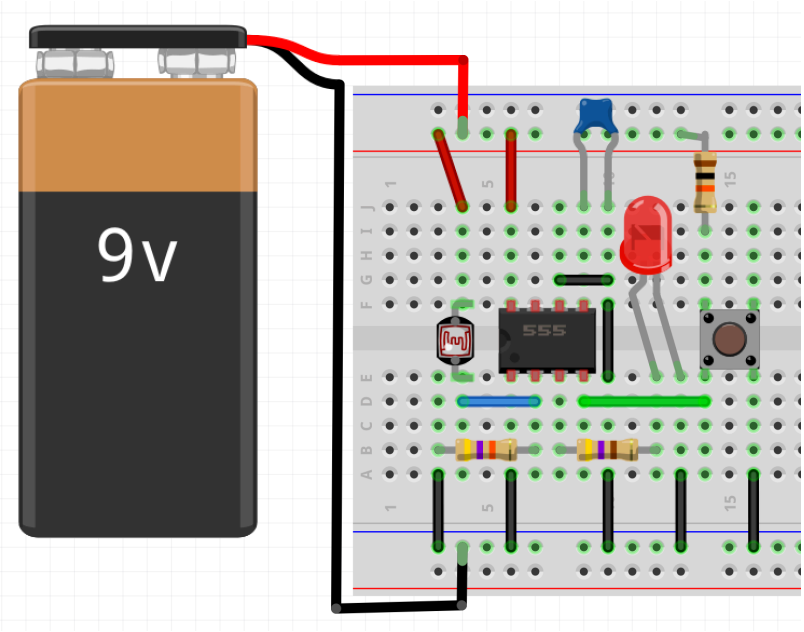


Monte o circuito acima e experimente em um lugar onde você possa variar a iluminação. Quando a luz estiver **acesa**, a resistência do LDR será baixa (menor que a resistência do potenciômetro que liga o pino 2 ao negativo), portanto o pino 2 (TRIG) terá tensão bem maior que 3V (1/3 de 9V) e não opera.

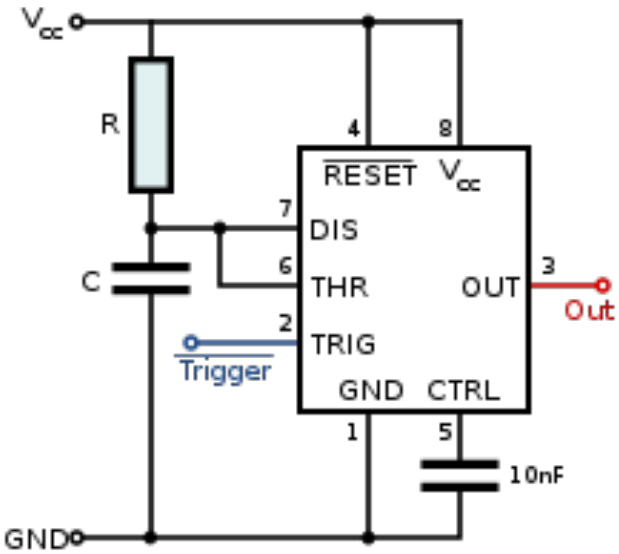
Quando a luz for **apagada**, a resistência no LDR ficará muito alta (bem maior que os 10k ohms que ligam o pino 2 ao negativo), fazendo cair a tensão no resistor para menos de 3V, que dispara o temporizador mudando o nível do pino 3 (OUT) para nível lógico **ALTO** (9V), e fazendo o LED acender. Depois que a luz for acesa, o LED *continuará aceso* e só apagará se houver um reset (apertando o botão, que ligará o pino RESET ao polo negativo, ou seja, tensão 0V).

O LED neste circuito representa uma *carga*. Pode ser substituído por outro circuito ou dispositivo. Por exemplo, pode ser substituído por um relé (chave elétrica), para acionar qualquer coisa (um alarme, um motor, etc.)

O bom funcionamento do circuito depende da luminosidade do ambiente. Os LDRs de 5mm e 7mm podem ter sensibilidades diferentes à luz. Normalmente o de 7mm é mais sensível e o LED só acenderá com uma escuridão maior (neste caso, precisará de um resistor menor). O desenho no protoboard abaixo usa um resistor de 47k no lugar do potenciômetro (substitua por um potenciômetro de 50k ou 100k para ajustar a sensibilidade se necessário, ou experimente diferentes resistores entre 10 e 100k).



4.1.2. 555 em modo monoestável



O circuito monoestável **encerra o ciclo automaticamente algum tempo depois do disparo**. Funciona como um temporizador ou cronômetro. A duração desse ciclo é determinada pelos valores do resistor **R** e do capacitor **C**.

Neste circuito, o pino 4 (Reset) é mantido inativo, sempre ligado diretamente em Vcc (8). Os pinos 6 (Threshold) e 7 (Descarga) são ligados um ao outro (e têm sempre o mesmo valor).

O pino 2 (disparador) inicialmente também é ligado ao positivo, mas **quando conectado ao negativo, ele dispara** e **inicia** um novo ciclo (fazendo o pino 3 ter valor ALTO).

O **fim** do ciclo acontece automaticamente depois de um determinado tempo provocado pela **carga do capacitor**, que está ligado aos pinos 6 e 7. O valor de **R** é calculado para controlar o tempo de carga do capacitor **C**. Quando a carga do capacitor atingir 2/3 da tensão de referência (**6V** = 2/3 de 9V), o pino 6 dispara, causando o fim do ciclo (faz o pino 3 ter nível lógico BAIXO, ou seja, 0V). Na sequência, o pino 7 descarrega o capacitor.

A fórmula (aproximada) para calcular o tempo (em segundos) em que a saída OUT permanecerá ligada é:

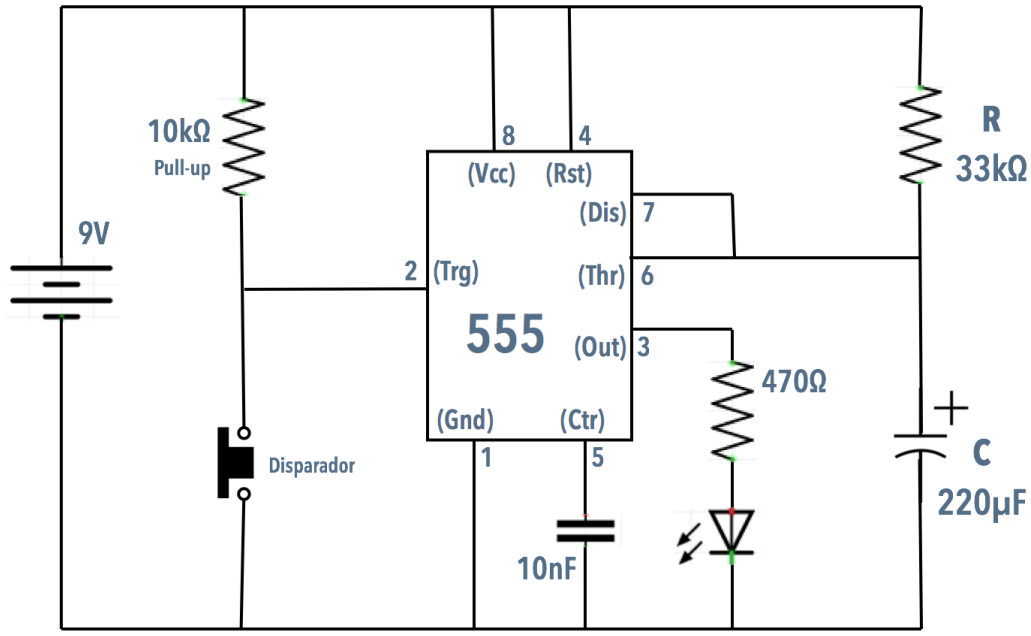
```
T = 1.1 x R x C
```

Experimento 18 – Temporizador

Material necessário:

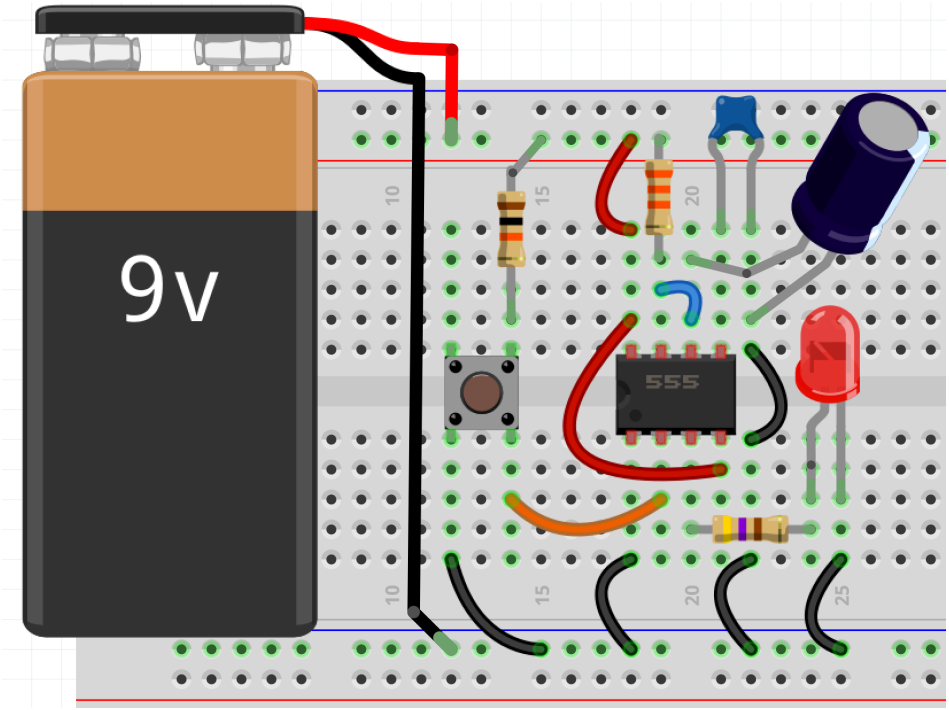
- Fonte de 9 ou 12V, ou bateria de 9V
- Protoboard, fios e jumpers
- Capacitor eletrolítico de 220µF
- Resistor de 33k Ω
- Resistor de 10k Ω
- Resistor de 470 Ω (ou 560 Ω, se fonte de 12V)
- LED
- Capacitor cerâmico de 10nF
- Chave táctil tipo push-button
- Circuito integrado 555

O circuito abaixo acende um LED por aproximadamente 8 segundos depois que o botão é apertado.



A duração foi calculada usando a fórmula acima: **33000 x 0,00022 x 1,1 = 7,26s**. Altere os valores de R e C para obter tempos diferentes.

A ilustração abaixo mostra como o circuito poderia ser montado em um protoboard:



Veja duas alterações desse circuito a seguir.

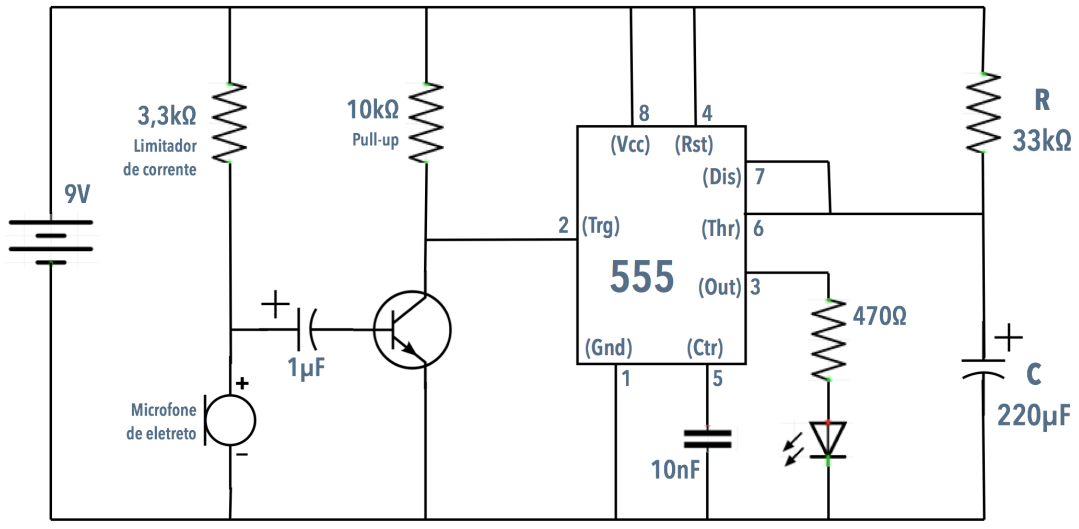
Alteração 18.1 – Usando um sensor sonoro para disparar o temporizador

Material adicional:

- Mini-microfone de eletreto
- Resistor de 3,3k Ω
- Capacitor de 1µF

O pulso que dispara o temporizador acontece quando o pino 2 (disparador) estiver em **nível lógico BAIXO** (tensão zero, ou negativa). O acionamento também pode ser causado por um sensor, como um LDR (luz), um termistor (temperatura), ou qualquer outro dispositivo ou circuito que **abaixe** a tensão no pino 2 até um valor próximo de zero.

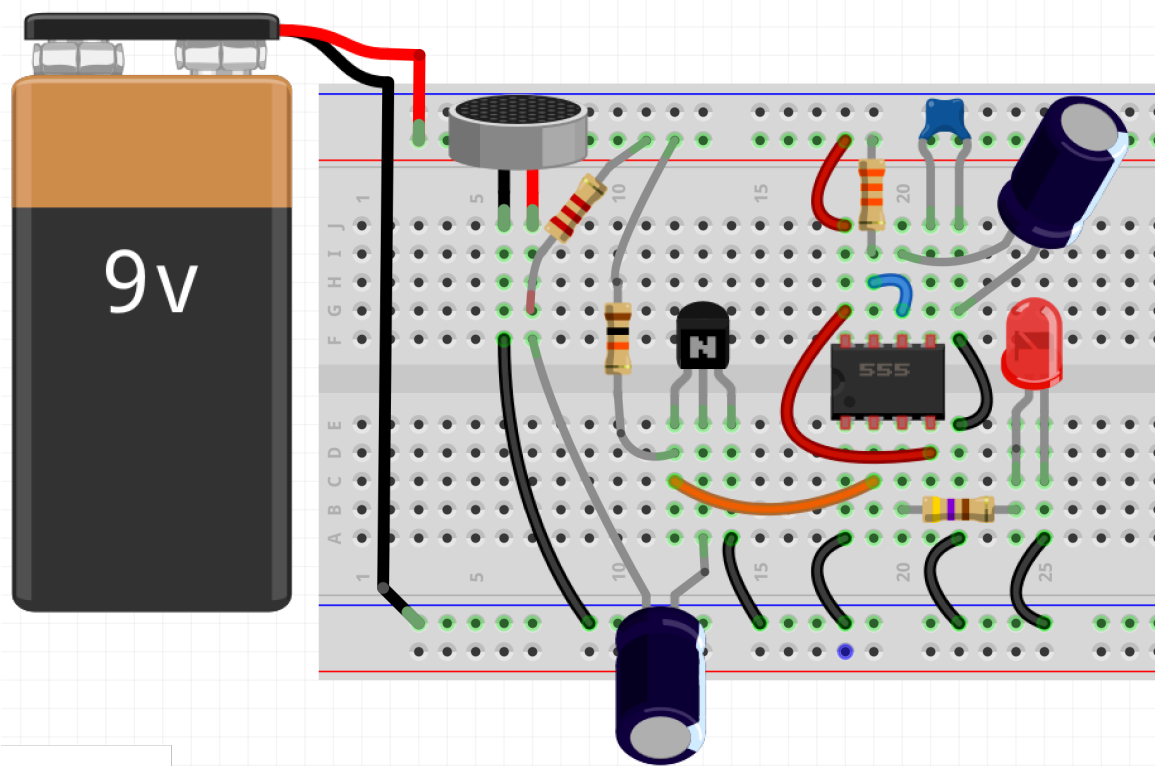
No exemplo abaixo, substituímos o botão por um circuito que amplifica o sinal recebido por um **microfone de eletreto**. O resistor de 10k mantém ALTO (positivo) o nível lógico do pino 2, mas quando som é captado pelo microfone, um **sinal** é enviado através do capacitor e o transistor **satura**, comportando-se como uma **chave fechada** entre os seus terminais C e E, conectando a entrada do pino 2 ao negativo (nível lógico baixo), desta forma disparando o temporizador.



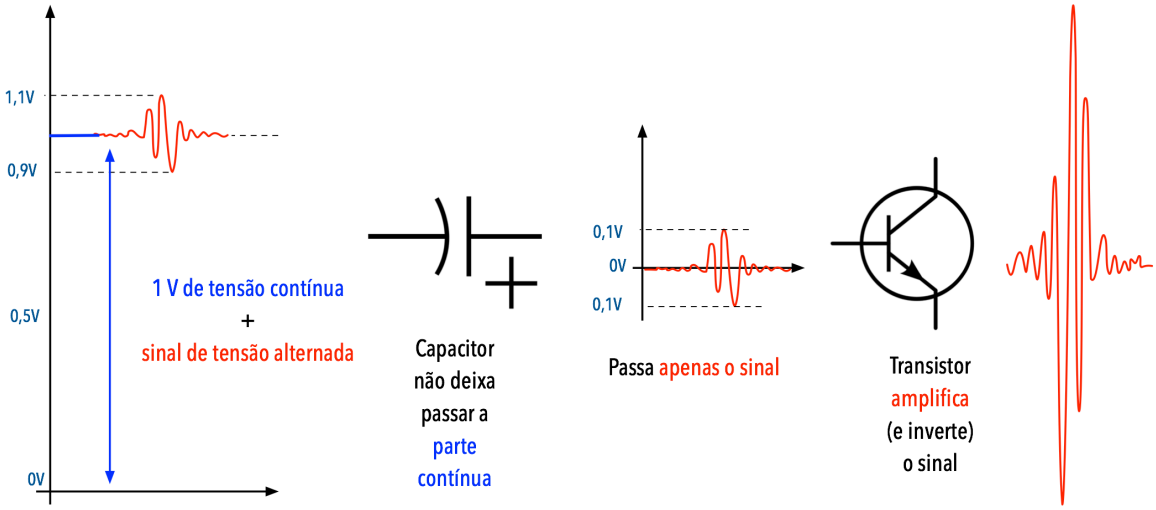
Observe que o microfone tem uma **polaridade** definida. (Pode ser necessário soldar terminais no microfone incluído no kit, se eles já não estiverem soldados, ou se forem muito pequenos. Veja o tutorial no final da apostila.)

Experimente gritar ou bater palmas perto do microfone. O som deve acionar o temporizador e fazer o LED acender por alguns segundos.

A ilustração abaixo mostra uma montagem possível usando o protoboard:



O capacitor de 1µF usado na base do transistor serve para que o componente de tensão contínua que alimenta o microfone não passe para o transistor. **O capacitor não deixa passar corrente contínua**, mas apenas transições e pulsos. Interessa amplificar **o sinal** recebido pelo microfone, mas ele sempre vem **sobreposto** à tensão do microfone, que produziria corrente para o transistor ligar mesmo não havendo som algum vindo do microfone. Usando o capacitor, apenas o sinal gerado pelo microfone passa para a base do transistor. Esse tipo de circuito é usado em muitas situações onde é preciso amplificar sinais e eliminar a componente contínua:



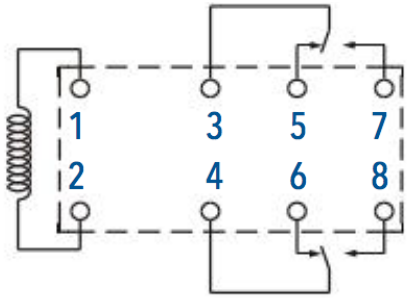
Alteração 18.2 – Substituindo o LED por um relé

Material adicional:

- Relé de 5V, dois polos, duas posições
- Diodo de propósito geral (1N4148 ou equivalente)
- Cigarra de 5V
- Resistor de 100 Ω (ou 220 Ω se fonte de 12V)

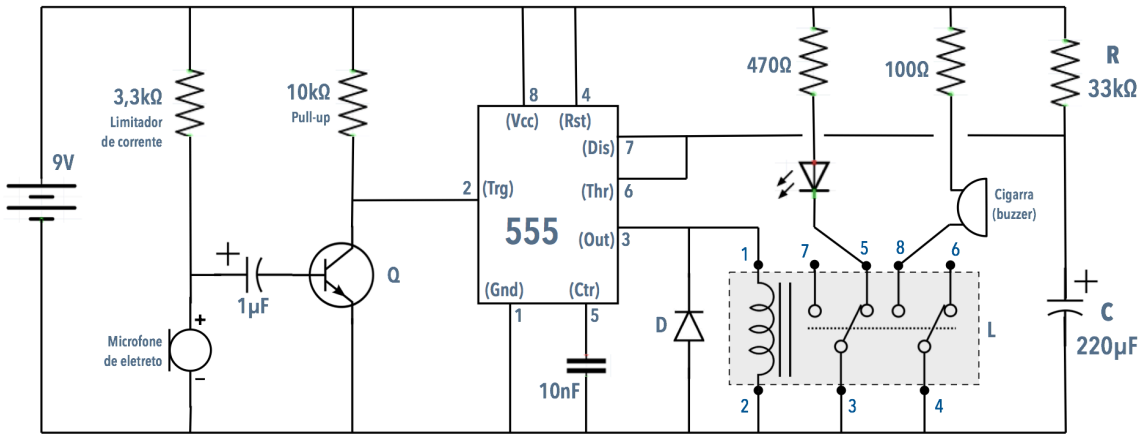
Finalmente o LED pode ser substituído por qualquer outro circuito. No exemplo abaixo, substituímos o LED por um **relé de duas chaves e duas posições**.

Um relé é uma chave acionada por um eletroímã e está descrito na referência no final da apostila. Ele tem oito terminais e deve ser posicionado no meio do protoboard, como os circuitos integrados. Internamente ele possui a seguinte estrutura:



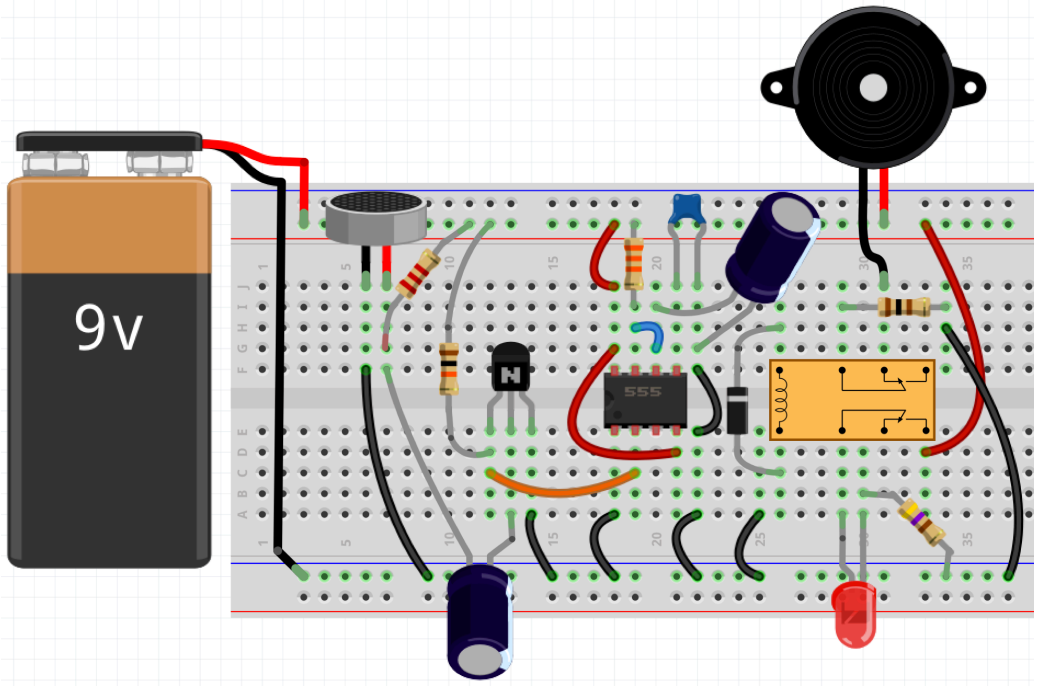
Os pinos 1 e 2 são os terminais do eletroímã. O relé contém **duas chaves**, representadas pelos pinos **3-5-7** e **4-6-8**. Os pinos 3-5 e 4-6 estão inicialmente conectados. Ao ligar o eletroímã (aplicando uma tensão de 4 a 10V nos terminais), os pinos 5 e 6 são desligados e a conexão muda para 3-7 e 4-8.

No circuito abaixo, o relé **L** foi colocado no lugar do LED. São usados os terminais conectados da primeira chave (3-5-7) para manter um LED aceso enquanto OUT estiver em nível lógico BAIXO (e consequentemente o relé estiver desligado). A segunda chave (4-6-8) conecta uma cigarra (buzzer) aos terminais inicialmente desconectados (4-8), que será ligada durante o período em que OUT mudar para nível lógico ALTO (e o eletroímã do relé estiver acionado).

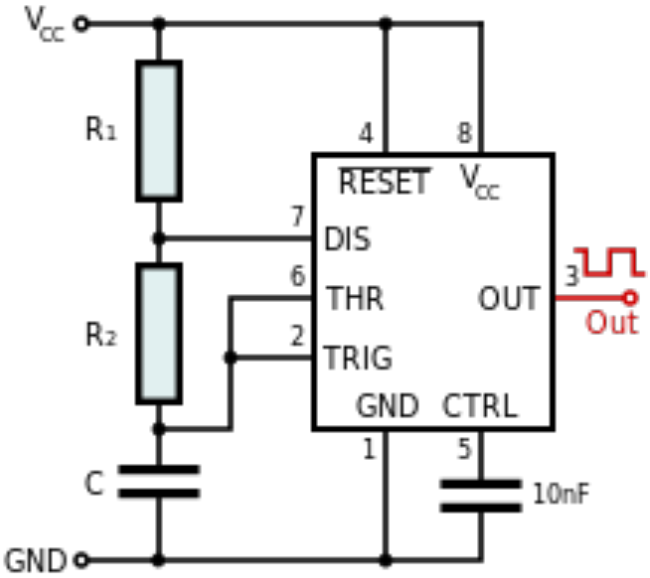


O **diodo D** é importante pois indutores em geral (relés, motores, transformadores) armazenam corrente que podem fluir no sentido contrário ao ligar o circuito. Esse pulso de corrente pode ter um valor muito elevado e queimar um componente. O diodo só permite a passagem de corrente em um sentido. Colocando-o da forma mostrada no circuito, impede-se que essa corrente reversa flua, protegendo o circuito. Diodos de proteção devem sempre ser usados ao incluir relés, transformadores, motores e outros indutores no circuito.

A ilustração abaixo mostra uma possível montagem com o protoboard. Observe que a cigarra também tem uma polaridade. A cigarra distribuída no kit tem valor nominal de 5V e pode ser usada com tensões entre 3 e 8V e corrente de 30mA, por isto incluímos um resistor para reduzir a tensão e a corrente que passa por ele.



4.1.3. 555 em modo astável



Neste modo, o 555 funciona como um **oscilador** gerando na saída OUT uma sequência contínua de pulsos alternados em uma frequência determinada pelos dois resistores (**R1** e **R2**) e capacitor (**C**), e pode ser usado para várias aplicações, por exemplo, piscar LEDs, produzir tons em um alto-falante, controlar a intensidade de LEDs e velocidade de um motor.

Pode-se usar um potenciômetro no lugar de R2 para variar a frequência dos pulsos gerados.

O mesmo capacitor é usado para acionar os pinos 6 (THR) e 2 (TRIG). Quando o capacitor carrega via R1 e R2 e atinge **2/3 de Vcc**, THR (6) dispara e encerra o ciclo, iniciando a descarga via R2 e DIS (pino 7), e fazendo o pino 3(OUT) = nível lógico BAIXO. Quando a carga do capacitor cai abaixo de 1/3 de Vcc, TRIG é acionado e OUT passa a ter nível ALTO, fechando a chave DIS e permitindo o reinício da carga do capacitor.

A fórmula para determinar a frequência (ciclos por segundo) é

```
1,44 / [ C x (R1 + 2 x R2) ]
```

A duração de cada pulso depende principalmente de R2. Para R1 pode-se escolher um valor fixo (ex: 10k Ω). Fazendo-se o valor de R2 muito maior que R1 garante pulsos positivos e negativos com duração semelhante (mas evite valores de R1 muito menores que 1k Ω). A fórmula para a duração de cada pulso é:

```
ALTO = 0,7 x C x (R1+R2)
BAIXO = 0,7 x C x R2
```

Pelas fórmulas, a **duração do pulso de nível lógico ALTO** (chamado de "ciclo de trabalho" ou "duty-cycle") sempre será um pouco maior que a duração do pulso BAIXO. Para obter pulsos iguais ou pulsos de nível logico baixo mais longos é preciso configurar o 555 de forma diferente (isto será mostrado mais adiante).

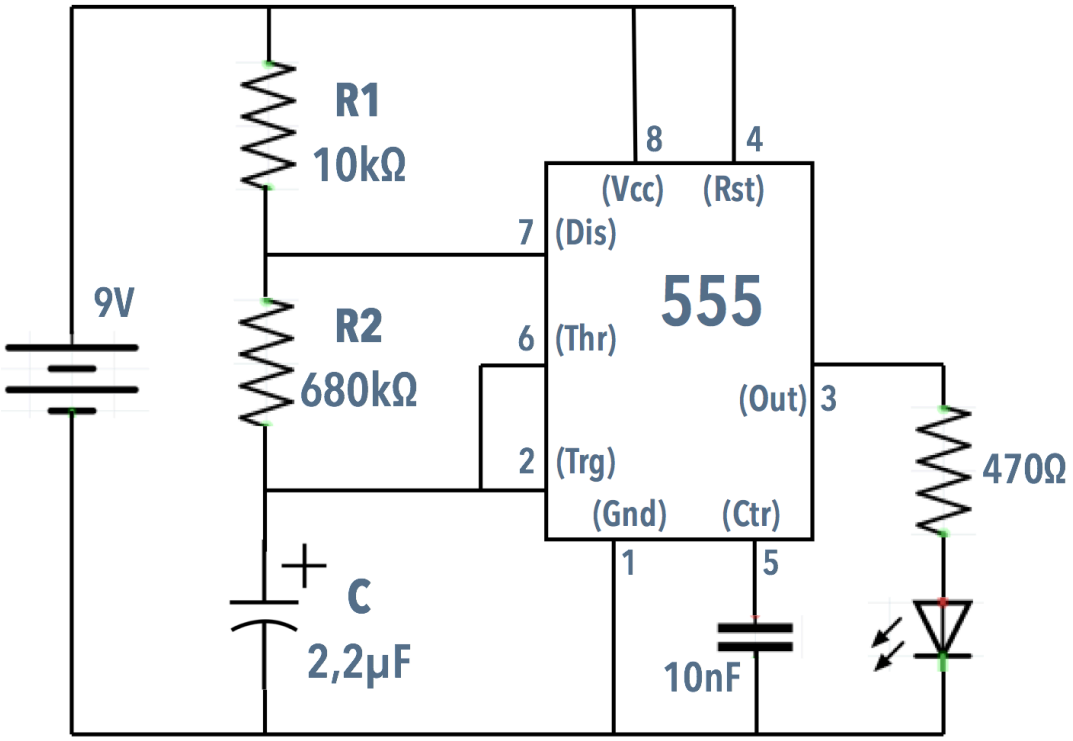
No final da apostila estão listados alguns sites que contém calculadoras para astáveis 555, onde você pode entrar com a frequência desejada, ciclo de trabalho desejado, e obter valores de capacitor e resistor. Também existem simuladores onde você pode experimentar alterar valores de capacitores e resistores na tela e ver a onda gerada na saída.

Experimento 19 – Pisca-pisca com LED usando 555

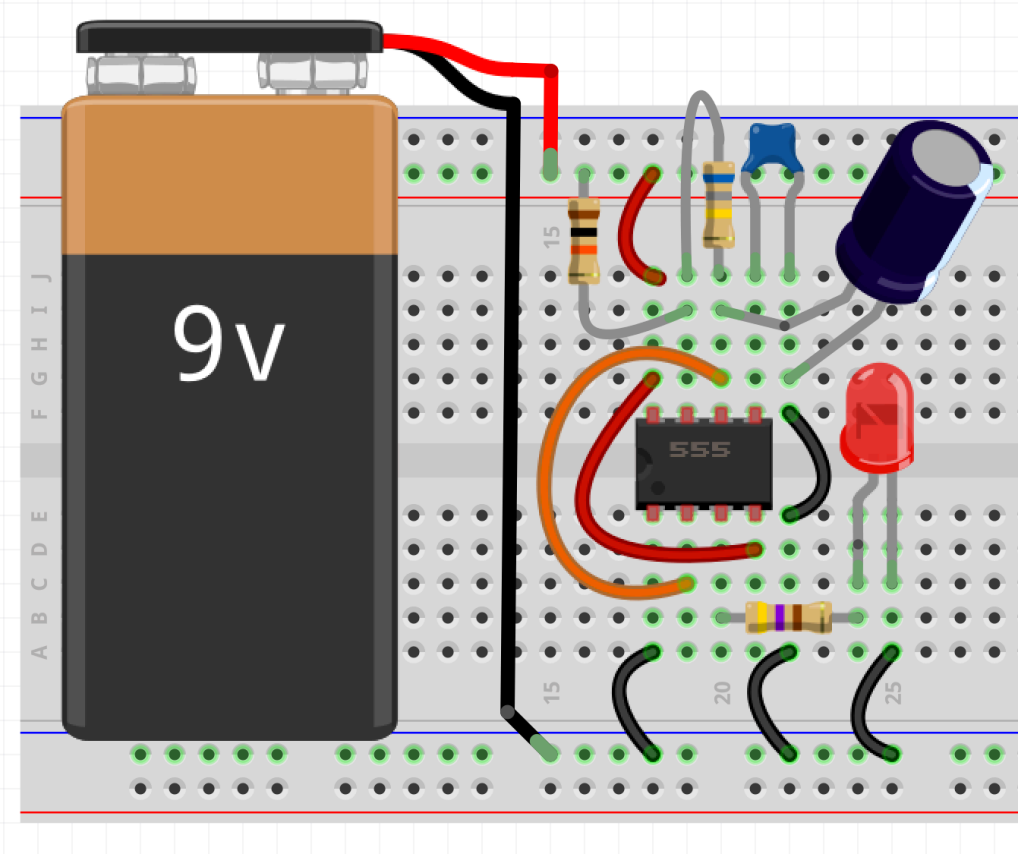
Material necessário:

- Fonte de 9 ou 12V, ou bateria de 9V
- Protoboard, fios e jumpers
- Resistor de 10k Ω
- Resistor de 680k Ω
- Capacitor de 2,2µF
- Resistor de 470 Ω (ou 560 Ω se fonte de 12V)
- LED
- Capacitor cerâmico de 10nF
- Circuito integrado 555

O circuito abaixo acende e apaga um LED por tempo determinado. Como R2 é muito maior que R1, é possível calcular a duração aproximada de cada pulso (ligado ou desligado) ignorando-o no cálculo e usando **0,7 x C x R2**.



A ilustração abaixo mostra uma possível montagem no protoboard:



Escolhendo valores menores de R2 e C, pode-se aumentar a frequência até atingir um valor audível (tipicamente entre 30 a 8000 pulsos por segundo, ou Hertz). Por exemplo, para produzir uma frequência de próxima de 440 Hz (frequência do diapasão usado para afinar instrumentos) pode-se usar um resistor de 47k Ω e um capacitor de 33nF, e ligar a saída do circuito a um alto-falante. Experimente usando os simuladores e calculadoras de astáveis 555 listados no final da apostila.

Experimento 20 (extra): Mini instrumento musical com 555

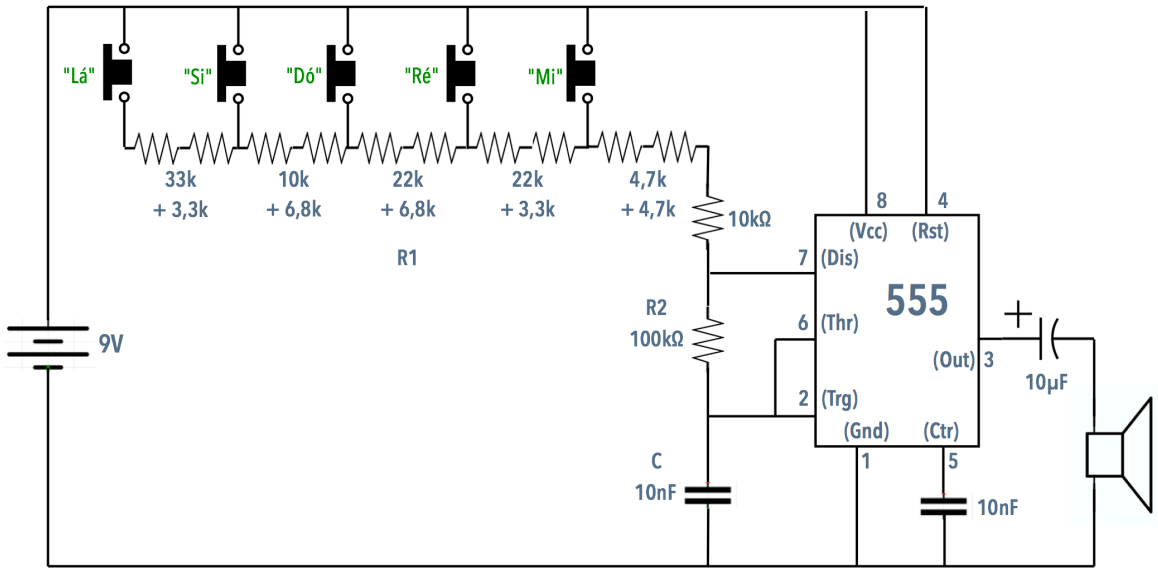
Este experimento parece complexo, mas é apenas uma variação do anterior. A maior complexidade é posicionar os componentes corretamente no protoboard.

Material necessário:

- Circuito integrado 555
- 5 chaves tácteis de pressão
- 1 resistor de 100k Ω
- 1 resistor de 33k Ω
- 2 resistores de 22k Ω
- 2 resistores de 10k Ω
- 2 resistores de 6,8k Ω
- 2 resistores de 4,7k Ω
- 2 resistores de 3,3k Ω
- 2 capacitores de 10nF
- 1 capacitor eletrolítico de 10µF
- 1 alto-falante de 8 Ω
- Fonte ou bateria de 9V
- Fios, jumpers e protoboard

O circuito abaixo é uma versão do multivibrador astável em frequência audível, similar ao circuito que montamos com transistores. Mas criamos 5 trechos RC (resistor-capacitor) separados, variando o resistor R1, de tal forma a produzir frequências que correspondem aproximadamente a notas musicais. Para acionar cada trecho é preciso apertar o botão correspondente.

Este circuito utiliza vários resistores **em série** para produzir os valores necessários para obter as frequências desejadas. Todos os resistores usados estão disponíveis no kit. Se você tiver outros resistores além dos resistores do kit, poderá usar os dados na tabela abaixo para tentar obter uma frequência mais próxima da esperada, ou usar menos resistores no circuito.



Cada botão, quando apertado, combina os resistores abaixo dele (em série) e faz o 555 apitar em uma frequência diferente, aproximadamente um tom ou semitom acima do botão anterior. As frequências calculadas são Lá(440Hz), Si(494Hz), Dó (523Hz), Ré (587Hz) e Mi (659Hz). Elas foram calculadas usando a fórmula:

```
1,44 / [ C x (R1 + 2 x R2) ]
```

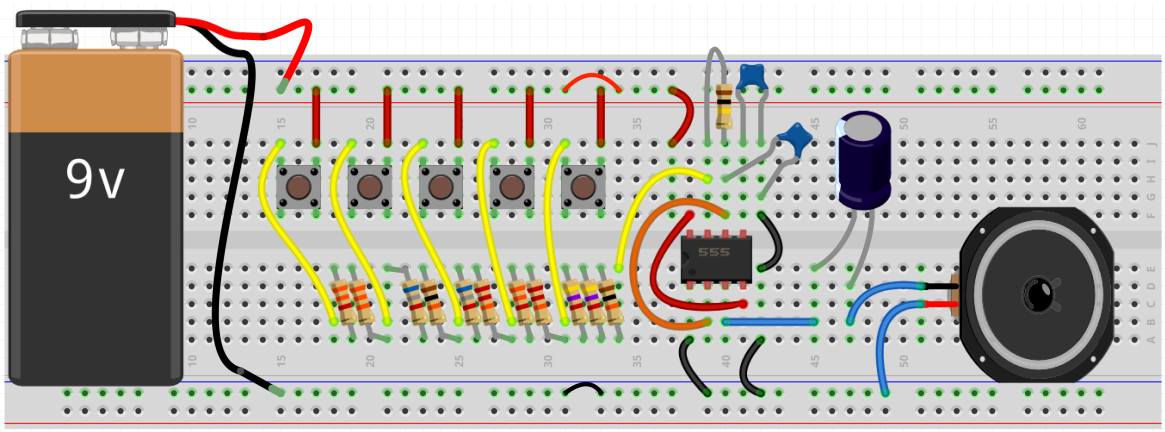
As combinações R1-R2-C foram construídos de acordo com a tabela abaixo. Apenas R1 varia:

Nota	**Frequência esperada**	**R1** (com resistores existentes no kit)	**R2**	**C**	**Frequência calculada**
Mi (5)	659 Hz	**19,4k** (4,7k+4,7k+10k)	100k	10nF	1,44 / (200k + 19,4k) * 10nF = **656 Hz**
Ré (5)	587 Hz	**44,7k** (19,4k+22k+3,3k)	100k	10nF	1,44 / (200k + 44,7k) * 10nF = **588 Hz**
Dó (5)	523 Hz	**73,5k** (44,7+22k+6,8k)	100k	10nF	1,44 / (200k + 73,5k) * 10nF = **527 Hz**
Si (4)	494 Hz	**90,3k** (73,5k+10k+6,8k)	100k	10nF	1,44 / (200k + 90,3k) * 10nF = **496 Hz**
Lá (4)	440 Hz	**126,6k** (90,3k+33k+3,3k)	100k	10nF	1,44 / (200k + 126,6k) * 10nF = **441 Hz**

O 555 produz na saída OUT uma **onda quadrada** e a fórmula usada não garante uma onda com pulsos de duração igual (o pulso de nível lógico ALTO sempre é igual ou maior que o pulso BAIXO, e geralmente é bem maior), portanto o som não é dos melhores.

Outra questão é a precisão dos resistores que é de 95%. Uma variação de 5% pode causar uma diferença de um semitom, que é significativa para a afinação. Mesmo assim, o som resultante deve ficar em uma frequência próxima da esperada. Você pode tentar melhorar o circuito incluindo um ajuste fino da afinação usando potenciômetros.

A ilustração abaixo é uma possibilidade de montagem em protoboard:



4.1.4. Controle da duração dos pulsos

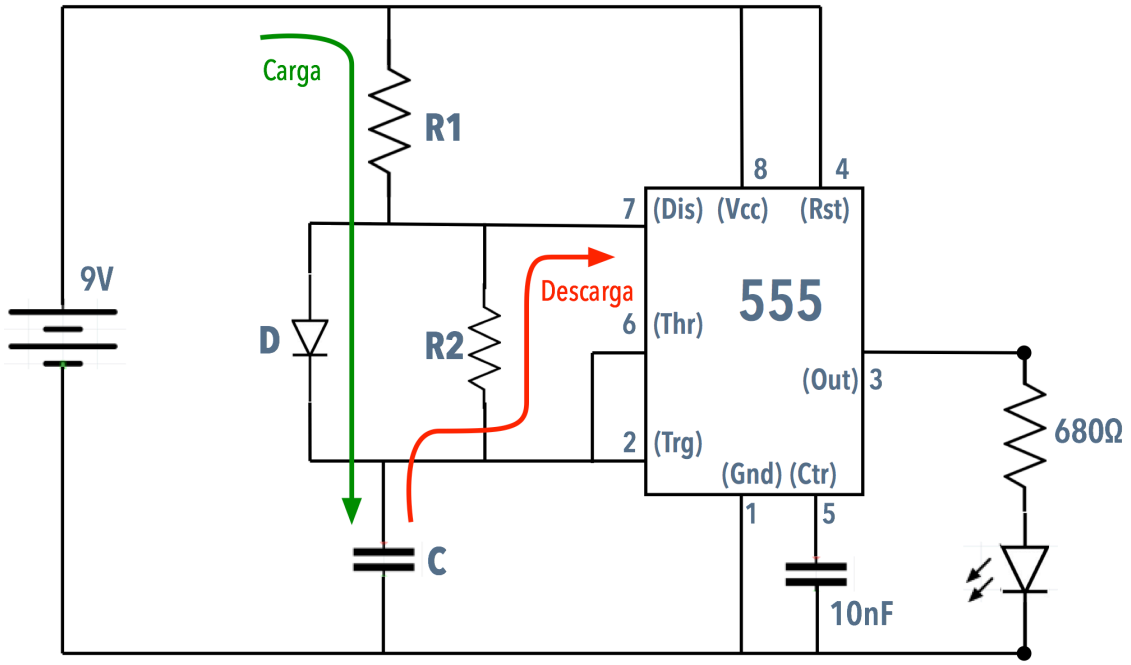
Nos exemplos acima, é **impossível** ter um pulso de nível BAIXO que seja **maior** que um pulso de nível ALTO. De acordo com as fórmulas, qualquer aumento de R2 que aumenta a duração do pulso em estado BAIXO, também aumenta o pulso em estado ALTO:

```
ALTO = 0,7 x C x (R1+R2)
BAIXO = 0,7 x C x R2
```

O pulso em estado **ALTO** acontece **durante a carga do capacitor**, com a corrente fluindo pelos dois resistores. Já o pulso em estado **BAIXO** corresponde à **descarga do capacitor** através do resistor R2 e pino 7 (DIS). Uma forma de evitar que R2 influencie no pulso ALTO é fazer com que ele **não participe da carga** do capacitor, apenas da descarga. Isto é possível usando um **diodo**.

Um diodo só permite a passagem de corrente em um sentido. Colocando um diodo em paralelo com o resistor R2, podemos deixar a corrente passar pelo resistor apenas quando ele estiver descarregando. Durante a carga o resistor será ignorado porque o diodo se comporta como um circuito fechado.

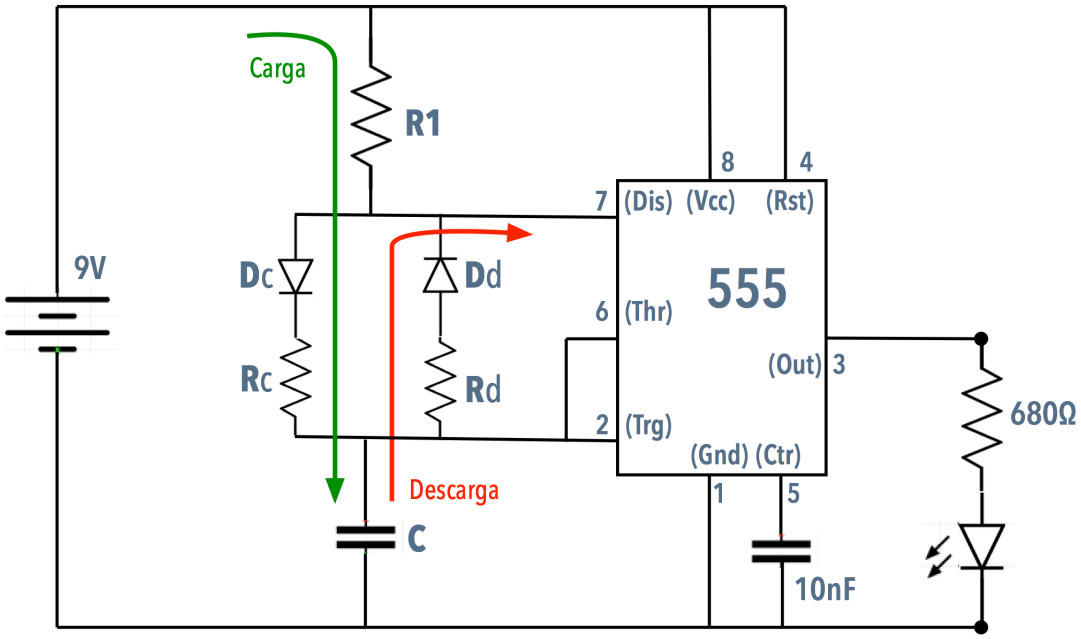
O circuito abaixo ilustra essa configuração:



Agora é possível calcular a duração de cada pulso usando apenas um resistor, e assim escolher suas durações de forma independente:

```
ALTO  = 0,7 x C x R1
BAIXO = 0,7 x C x R2
```

Existe uma variação do circuito acima que é mais comum por permitir variar a largura dos pulsos mantendo R1 fixo. Essa configuração é importante para o uso do 555 com **PWM** – uma técnica eficiente para **simular saída analógica** com o 555. Esta variação está mostrada abaixo:



As fórmulas para calcular a duração de caba pulso são:

```
ALTO  = 0,7 x C x (R1 + Rc)
BAIXO = 0,7 x C x Rd
```

As fórmulas são aproximadas e não levam em conta a queda de tensão nos diodos (que é de aproximadamente 0,7V).

4.1.5. PWM – Pulse Width Modulation

PWM significa **modulação de largura de pulso**. É uma das principais técnicas usadas para **simular uma saída analógica através de um circuito digital**. Consiste em gerar pulsos digitais em alta frequência, variando sua largura para produzir uma **valor médio** de tensão variável. Ou seja, o PWM é um pisca-pisca que varia os tempos aceso e apagado muito rapidamente, produzindo a **ilusão** de que a intensidade está variando.

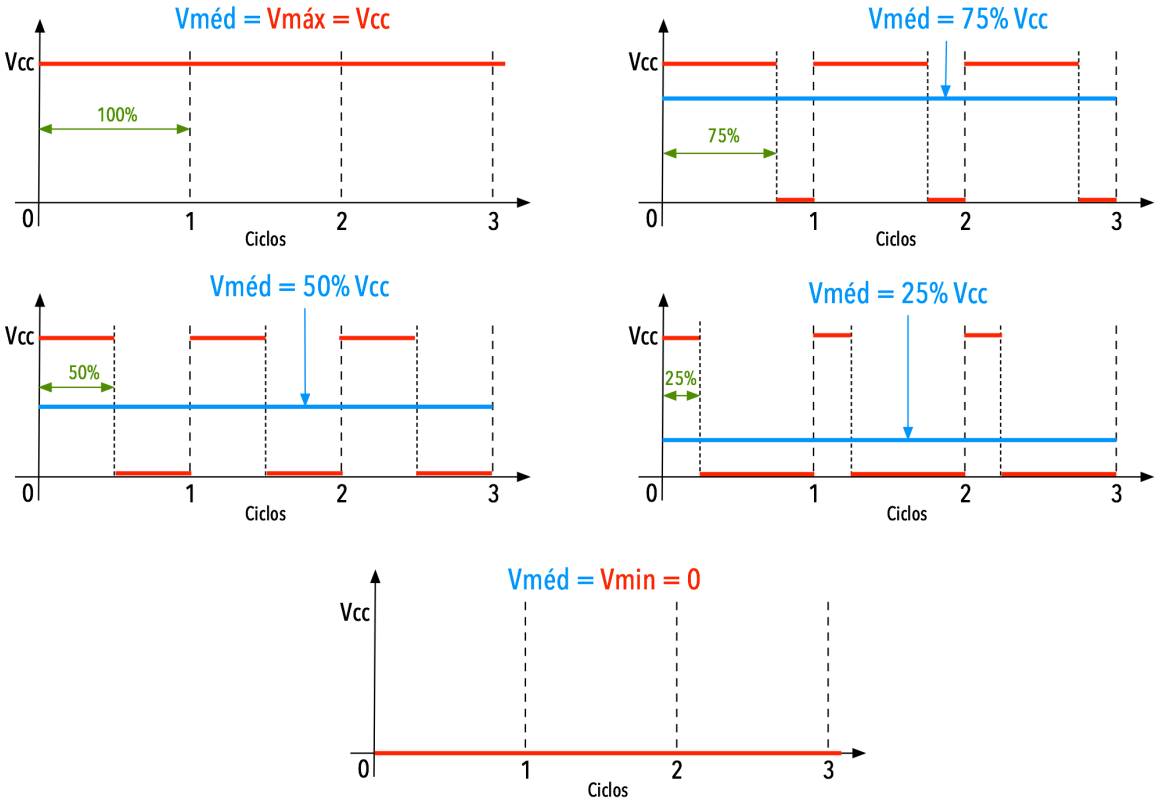
Por exemplo, se um pulso de **onda quadrada de 5V** fica **metade do tempo ligado**, e **metade do tempo desligado** (em 0V), o **valor médio simulado é de 2,5V**. Aplicando um sinal desses a um LED, ele piscará muito rapidamente ficando aceso metade do tempo, e irá **aparentar** ter metade do brilho que teria com um valor contínuo.

Se o pulso ficar 25% do tempo em 5V, o valor médio cai para 1,25V, e se ele ficar 75% do tempo em 5V e apenas 25% do tempo em 0, o valor médio aumenta para 3,75V. Comparada ao uso de um potenciômetro, que consome energia dissipando na forma de calor, PWM é uma técnica mais eficiente de variar tensão e corrente, já que a carga fica desligada (consumindo zero amperes) durante o tempo que o pulso é BAIXO.

Com PWM o 555 pode ser usado para implementar aplicações como controles de intensidade luminosa (*dimmers*) e controle de velocidade de motores. É bastante usado em circuitos com Arduino, que implementa suas saídas analógicas usando PWM.

Nem todo circuito funciona com PWM. Dispositivos que têm limite de tensão menor que o valor máximo produzido podem ser danificados com uma tensão simulada baixa (pois ela sempre alterna entre valores máximo e mínimo). Nessas situações é preciso construir um circuito retificador para converter as ondas quadradas em um valor contínuo.

Os gráficos a seguir ilustram o funcionamento de PWM descrito acima.



Para que o primeiro pulso possa ser menor que o segundo, é preciso usar a configuração mostrada anteriormente (usando dois diodos) para poder controlar isoladamente os tempos de carga e descarga do capacitor.

Nos experimentos a seguir, implementaremos PWM nessa configuração usando um **potenciômetro**, que irá permitir a variação do equilíbrio de resistências de um divisor de tensão, e desta forma variar os tempos de carga e descarga do capacitor, e consequentemente a largura dos pulsos na saída OUT (pino 3) do 555.

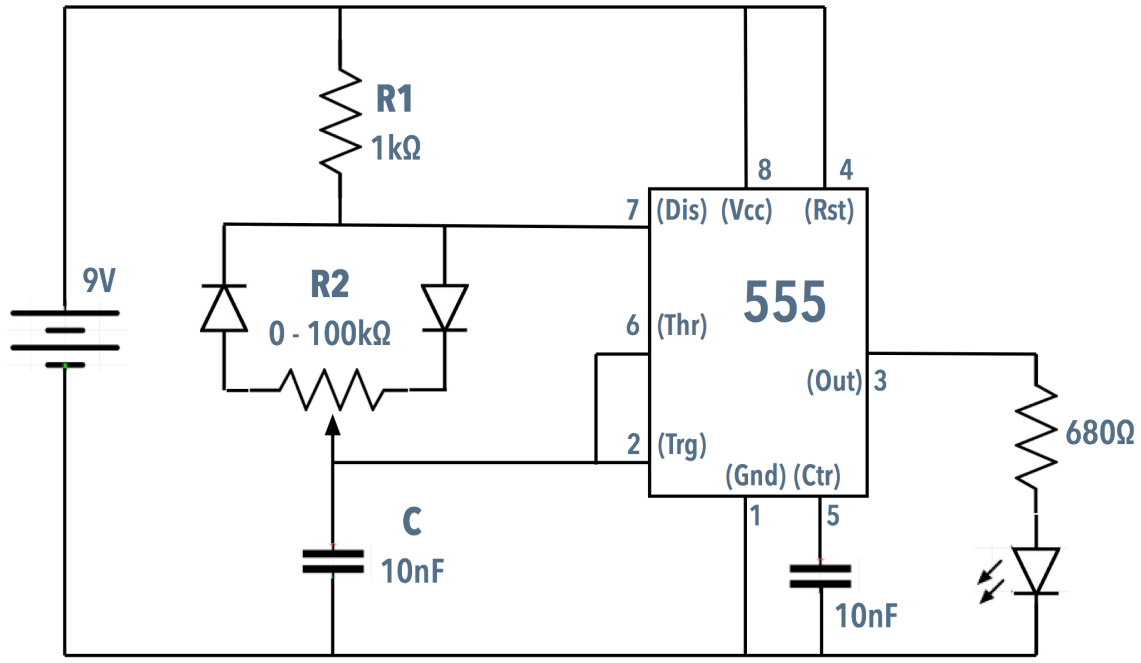
Experimento 21 (extra) – Dimmer usando PWM

Material necessário:

- Fonte ou bateria de 9V
- 1 resistor de 1k Ω
- 1 resistor e 680 Ω
- 1 potenciômetro de 100k Ω
- 2 diodos de propósito geral (1N4148 ou equivalente)
- 2 capacitores de 10nF
- 1 LED de qualquer cor
- 1 circuito integrado 555
- Protoboard, fios, jumpers

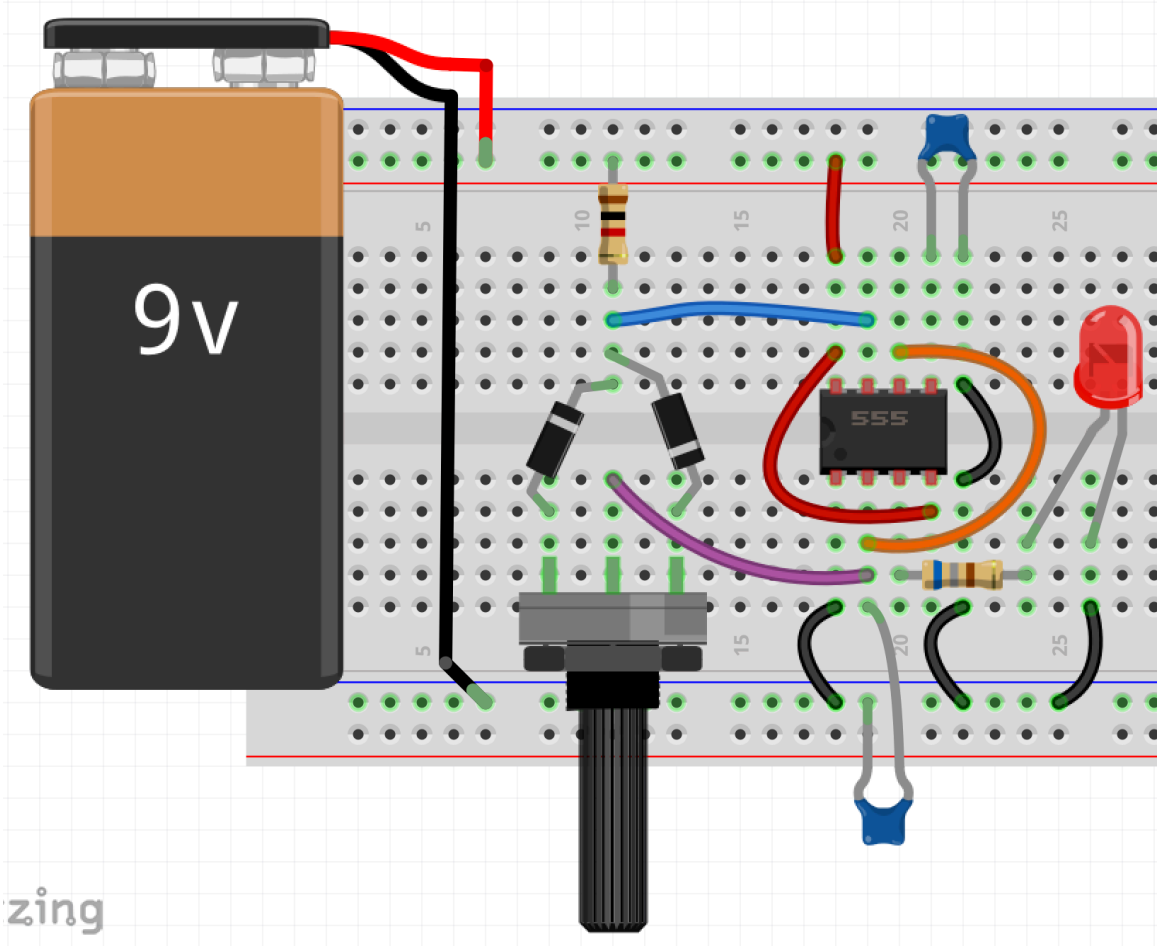
O circuito abaixo controla a intensidade do brilho do LED de forma eficiente usando PWM, em vez de desperdiçar energia dissipando o excesso de corrente com um resistor limitador (como fizemos no experimento com LED RGB).

Com PWM o LED permanece desligado parte do tempo. O LED acende e apaga completamente muito rapidamente, portanto a redução de brilho é uma ilusão causada principalmente pela persistência da visão (é o mesmo efeito que ocorre quando assistimos um filme no cinema), embora parte do efeito também se deva ao fato de que a onda quadrada não é perfeita.



O potenciômetro varia a largura de pulso para mais ou para menos, fazendo com que o brilho aparente do LED varie. O LED é a **carga** do circuito, e pode ser substituído por outro dispositivo analógico ou circuito que funcione com PWM.

A ilustração abaixo é uma possível implementação para o circuito acima.



Variação 21.1 – Controle de velocidade de motor com PWM

Material adicional necessário:

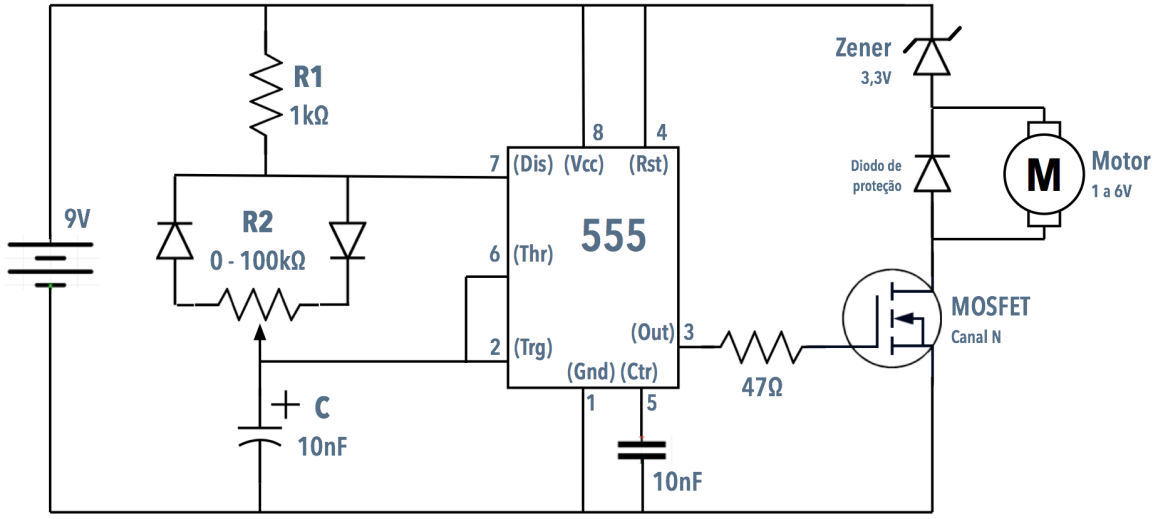
- 1 transistor MOSFET de canal N de propósito geral (2N7000 ou equivalente)
- 1 resistor de 47 Ω
- 1 diodo Zener de 3,3V e 1W (1N4728 ou equivalente)
- 1 diodo de propósito geral (1N4148 ou equivalente)
- 1 motor de 3V (opera com 1 a 6V)

O circuito abaixo é uma adaptação do anterior. Nesta versão PWM é usado para variar a velocidade de um motor. O circuito possui várias adaptações necessárias para lidar com algumas limitações impostas pelo 555, transistor e pelo motor:

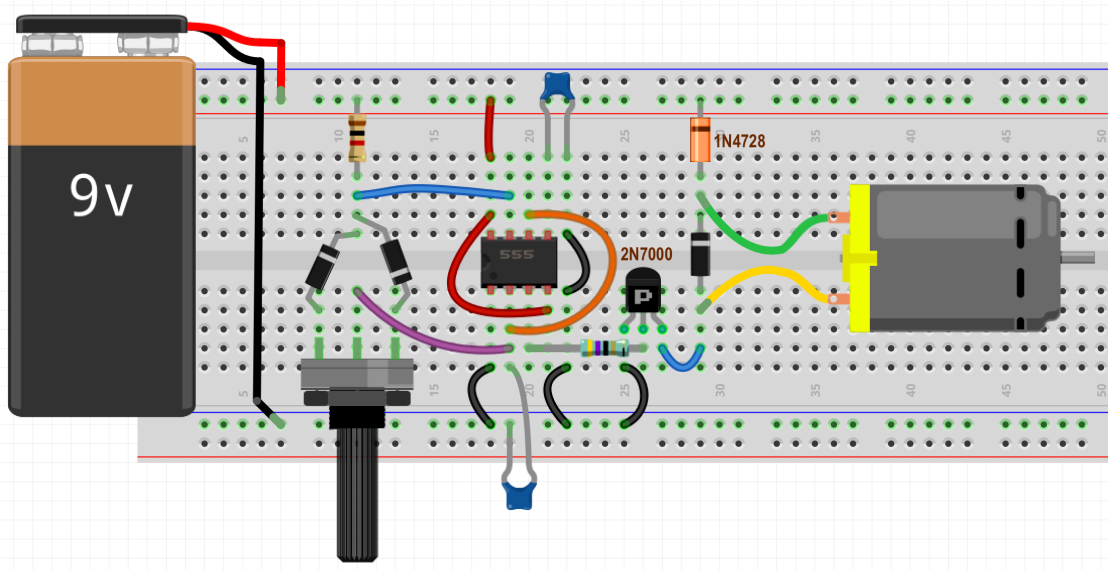
- **Pulsos de corrente gerados pelo motor:** uma bobina (indutor) **armazena** corrente, que é liberada em um pulso durante transições (ao ligar ou desligar a energia sobre ela). Um pulso de corrente reversa muito alta pode queimar um componente. Por este motivo usamos um **diodo de proteção** posicionado de forma **reversa** e em paralelo com o relé em um experimento anterior, para que esse pulso seja curto-circuitado e não afete o resto do circuito. O mesmo vale para motores, transformadores e indutores em geral.
- **Demanda de corrente do motor:** além de pulsos elevados, a demanda de corrente do motor pode ser maior que a suportada pelo 555 ou Arduino. Nestes casos, precisamos isolar o circuito com um transistor. Pode ser um transistor bipolar, mas a opção mais eficiente neste caso é usar um **transistor MOSFET**, que fornece um isolamento maior e quase não consome energia quando desligado (ele tem uma resistência interna altíssima na entrada).
- **Tensão máxima suportada pelo motor:** o motor do kit tem uma faixa de operação que varia de 1 a 6V. Deixá-lo funcionando muito tempo em 9V pode queimá-lo, portanto é preciso garantir que a tensão sobre ele não passe de 6V. Não adianta limitar a tensão via PWM, pois ela é simulada (o pulso sempre irá produzir valores máximos de 9V). Um divisor de tensão com resistores também não garante essa redução, pois a resistência interna do motor varia durante a operação. Uma solução para regular a tensão é usar um **diodo Zener**. Esse tipo de diodo é construído para manter uma **tensão reversa fixa** sobre ele, mesmo com grandes variações de corrente. No kit temos o diodo 1N4728, que mantém uma tensão fixa de 3,3V. Colocado em série com o motor, de forma reversa, ele irá subtrair 3,3V da tensão máxima produzida na saída (ou seja, do pulso máximo), garantindo que o motor nunca receba uma tensão maior que 6V (para uma alimentação de 9V).

Transistores MOSFET não têm terminais base, emissor e coletor, mas **Gate (G)** – que tem função similar à base, **Drain (D)**, similar ao coletor, e **Source (S)**, que tem um papel similar ao emissor. No circuito abaixo, a saída do 555 controla o terminal **G** que controla a passagem de corrente de **D para S** que aciona o motor.

Veja mais informações e links sobre diodos Zener, transistores MOSFET e motores na referência no final da apostila.



Uma possível implementação do circuito acima em um protoboard está ilustrado abaixo.



PWM funciona no experimento com motor devido à inércia, que impede que o motor pare completamente quando desligado. Portanto, o motor está sempre acelerando e desacelerando muito rapidamente, o que na prática produz uma velocidade constante reduzida.

4.2. Outros circuitos integrados

No kit há dois outros circuitos integrados que funcionam com lógica digital. Os experimentos extras a seguir exploram o uso deles.

4.2.1. Contador de década 4017

O circuito integrado 4017 é um **contador de década**. Basicamente ele sabe contar até 10 e dentre os seus 16 pinos possui 10 pinos que produzem uma saída em nível lógico ALTO ou BAIXO, dependendo dos controles que disparam sua contagem. Essas saídas podem ser usadas para ligar circuitos, acender LEDs, disparar cigarras, etc. A contagem é produzida por um pino de entrada que avança a contagem através do recebimento de um pulso (que pode ser produzido por uma chave, um sensor, um oscilador de cristal de quartzo ou gerado através de um temporizador como o 555).

Como o 555, o 4017 também é um circuito integrado muito antigo (tem quase meio-século de existência).

O experimento a seguir usa o 4017 para acender uma série de LEDs em sequência a cada pulso recebido. Uma alteração do experimento inclui o 555 para que os pulsos sejam gerados periodicamente.

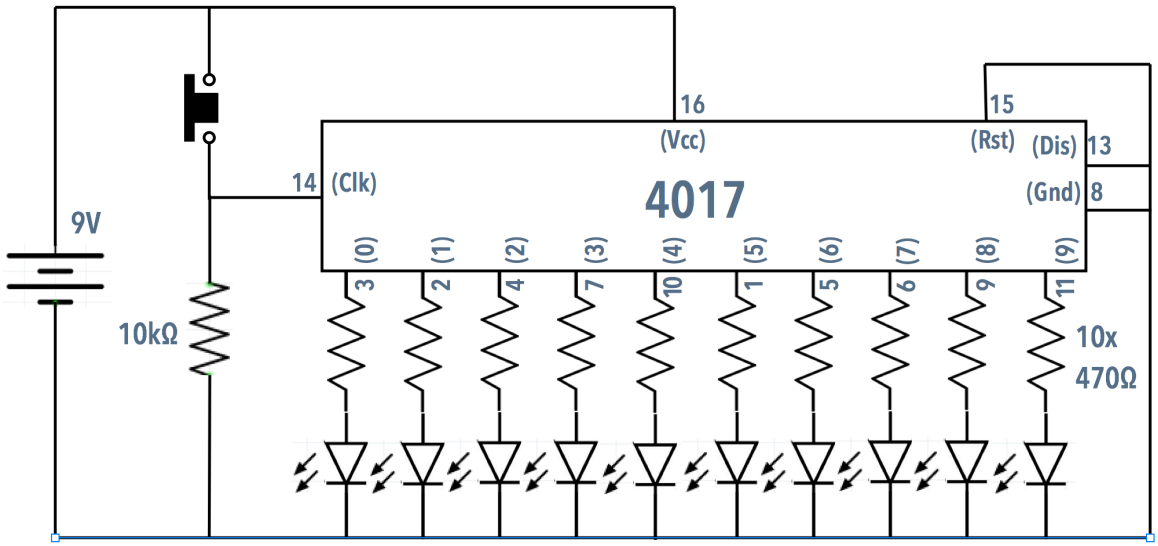
Experimento 22 (extra): sequenciador de LEDs com o 4017

Este experimento parece complexo apenas porque tem muitos fios, mas na verdade é muito mais simples que os outros experimentos montados anteriormente. A principal complexidade é verificar com cuidado as polaridades e o posicionamento dos fios e componentes.

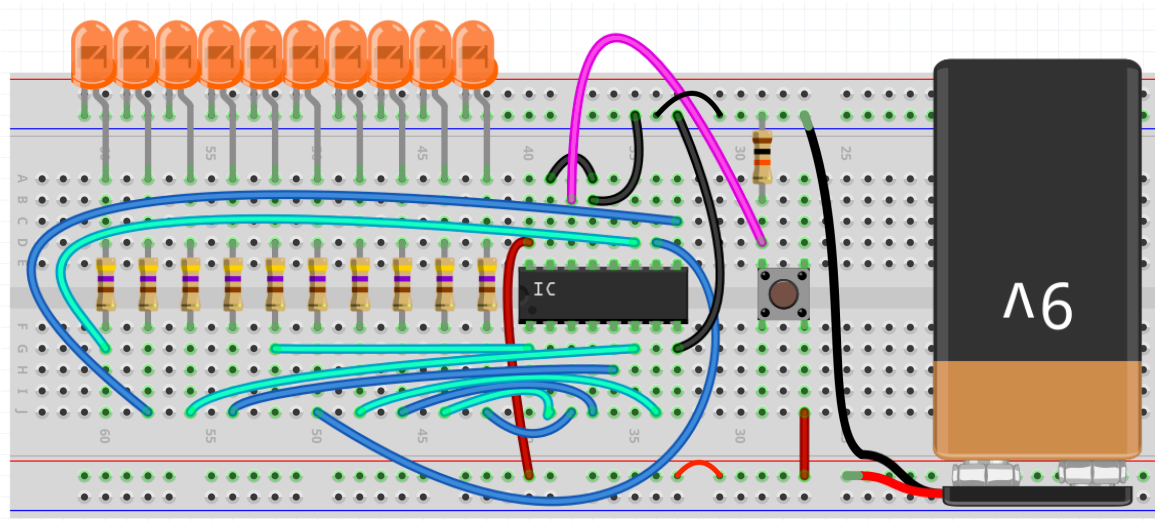
Material necessário:

- 1 circuito integrado 4017
- 10 resistores de 470 Ω
- 10 LEDs
- 1 resistor de 10k
- 1 chave táctil de pressão
- Fonte ou bateria de 9V
- Fios e jumpers

Monte o circuito abaixo. Lembre-se que os pinos são contados em sentido anti-horário a partir do chanfro em um dos lados, ou do ponto, que indica o pino 1. Veja o diagrama do 4017 e link para seu *datasheet* no final desta apostila.



Esta é uma possível implementação com protoboard.



Você também pode posicionar os LEDs formando um círculo, assim a contagem não acaba nunca e o LED fica girando para sempre. Isto pode ser feito em uma placa ou superfície qualquer, soldando (ou amarrando) os terminais dos LEDs.

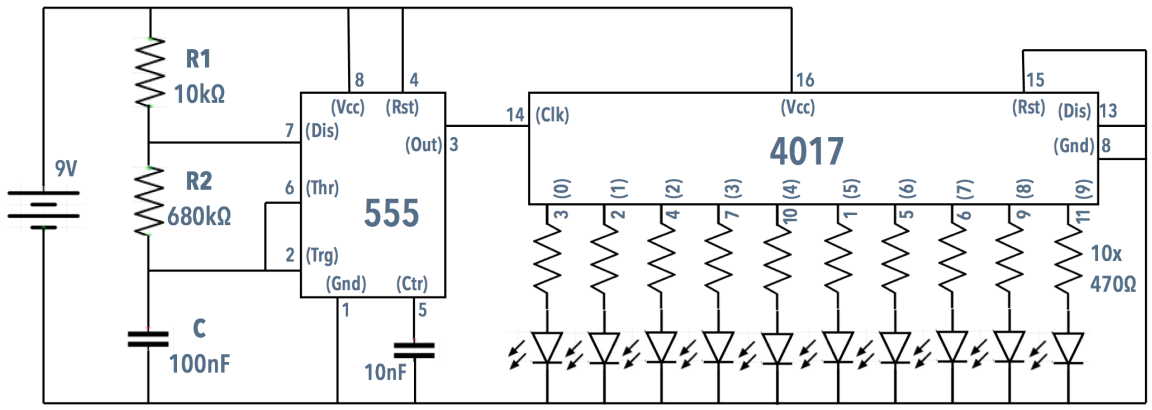
Alteração 22.1 – Sequenciador de LEDs automático com 555 e 4017

Este circuito adiciona um gerador de pulsos automático com 555, para o circuito anterior.

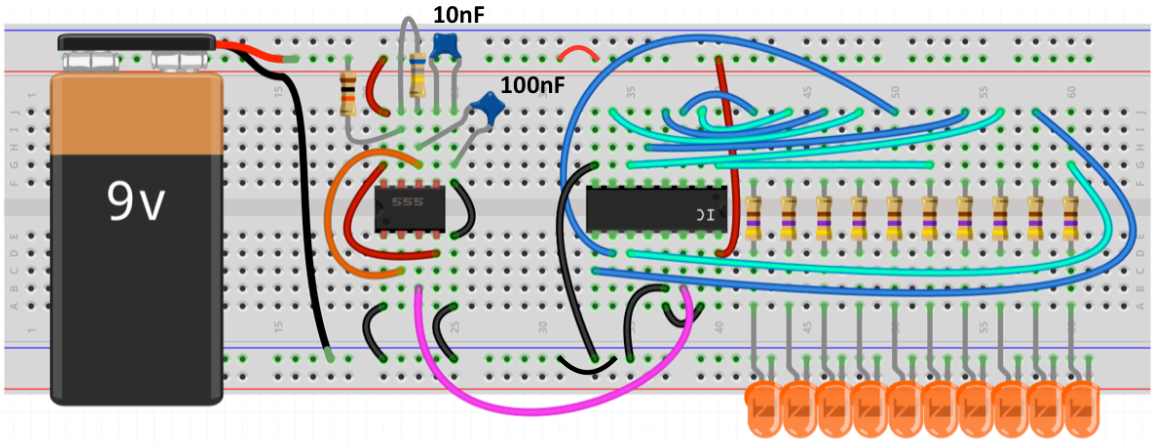
Material adicional necessário

- Circuito integrado 555
- Resistor de 680k Ω
- Capacitor de 10nF
- Capacitor de 100nF (104)

A função do 555 é apenas de produzir pulsos para a entrada do 4017. Você pode alterar os valores do capacitor (de 100nF) e do resistor (de 680k) para variar a frequência e fazer o LED se mover mais ou menos rapidamente.



Alguns protoboards têm uma interrupção no meio dos condutores laterais. O circuito abaixo assume essa possibilidade e faz a ligação entre as duas metades com um fio (na faixa positiva e negativa).



4.2.2. Decodificador para display de 7 segmentos 4026

Um **decodificador** é um circuito que adapta dados de entrada codificados para uma saída esperada por um dispositivo. A codificação pode ser uma sequência de pulsos. Assim como o 4017, o 4026 também sabe contar pulsos, mas em vez de ligar uma dentre 10 saídas, ele liga duas a sete saídas de uma vez. Essas sete saídas correspondem aos LEDs de um display de sete segmentos, que pode ser usado para representar dígitos.

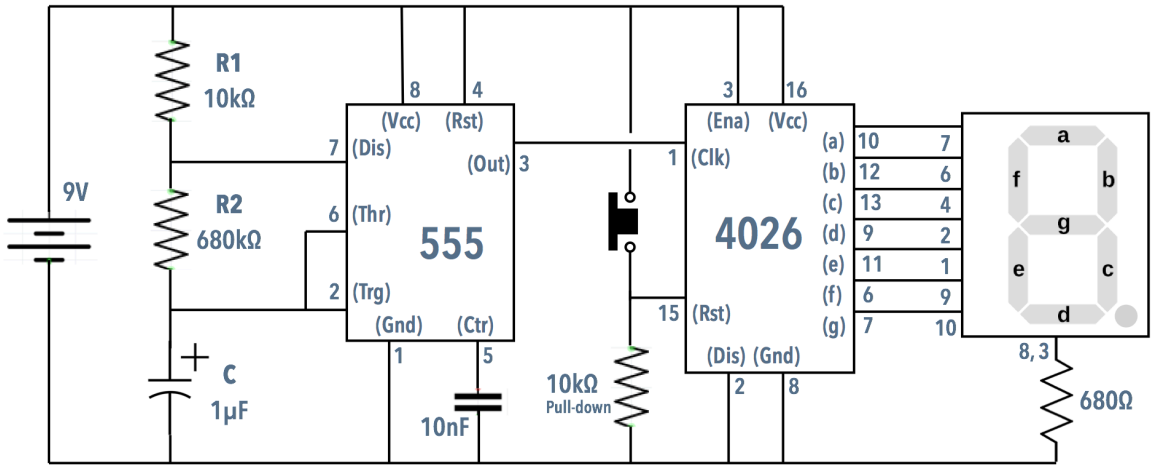
Experimento 23 (extra) – Contador de 0 até 9 com display de 7 segmentos e 4026

Este circuito é ainda mais simples que o anterior (basicamente ligar fios a terminais), e aproveita o gerador de pulsos automático criado acima.

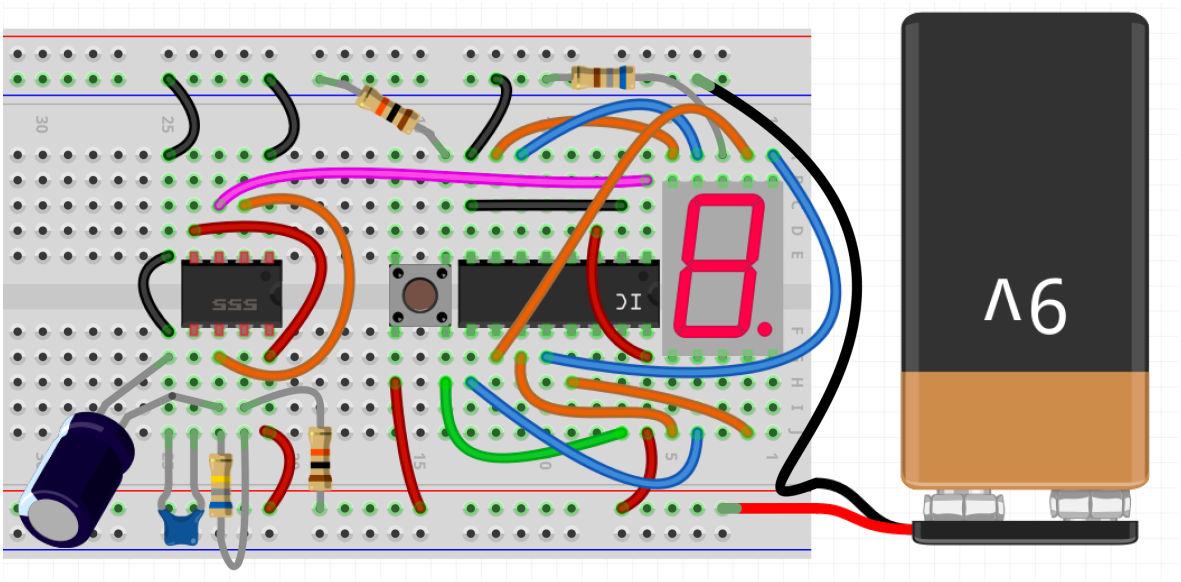
Material necessário

- Circuito temporizador 555 (construído na alteração do experimento anterior)
- 1 capacitor de 1µF (substituindo o de 100nF, para que a contagem seja mais lenta)
- 1 resistor de 10k Ω
- 1 resistor de 470 Ω
- Chave táctil de pressão
- Display de 7-segmentos
- Circuito integrado 4026
- Protoboard, fios e jumpers
- Fonte ou bateria de 9V

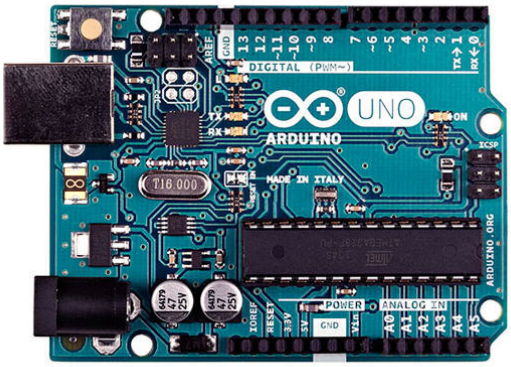
Monte o circuito a seguir. Observe que a parte do 555 é idêntica ao do circuito do experimento anterior com exceção do capacitor de 100nF que foi trocado por um de 1µF, para que a contagem seja mais lenta.



Veja detalhes sobre a pinagem do display e do 4026 na referência no final da apostila. Esta é uma possível implementação em protoboard.



5. Introdução ao Arduino

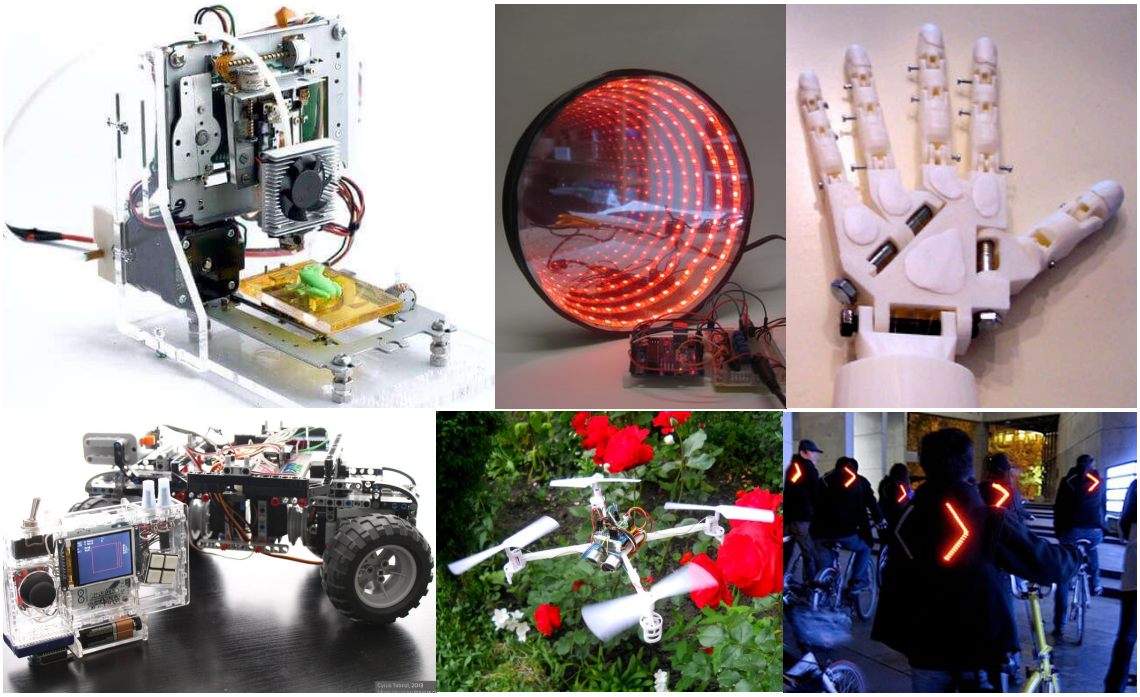


Arduino é o nome de um projeto que consiste na especificação de uma **linguagem** de programação e de um **circuito** baseados em um **microcontrolador**, como uma ***plataforma eletrônica*** para facilitar a construção de dispositivos interativos que detectam e controlam objetos no mundo físico. É um projeto de **fonte aberta**: a especificação do hardware e software são livres, permitindo que qualquer um fabrique e venda placas Arduino sem pagar royalties a ninguém. A plataforma original foi criada em Ivrea, na Itália, por estudantes italianos e colombianos, com a intenção de facilitar o uso da eletrônica por artistas e designers.

5.1. Projetos artísticos com Arduino

Arduino é ideal para projetos artísticos, pois facilita muito o projeto e construção de circuitos eletrônicos, eliminando grande parte da sua complexidade. Muitos projetos que normalmente requerem o cálculo de circuitos elaborados com resistores, capacitores e transistores para usar um sensor, podem ser construídos usando apenas este sensor e um Arduino. É necessário, no entanto, programar o comportamento desejado em uma linguagem de computador.

Arduinos são usados em inúmeros projetos, dos mais simples aos mais complexos. Luzes que piscam ao ritmo da música, alarmes que disparam quando percebem movimento, robôs que interagem com o ambiente e controlam máquinas pela Internet, roupas, óculos e bolsas multimídia, instalações visuais, sensoriais, cinéticas e sonoras, sistemas de automação residencial, sistemas de irrigação, sistemas de realidade virtual, próteses controladas por voz, jogos, drones, controladores MIDI, impressoras 3D são alguns exemplos de projetos já criados com Arduino.



Nesta seção faremos uma breve introdução ao Arduino através de alguns circuitos e programas simples que você pode usar como base para projetos mais sofisticados. No final da apostila estão listados alguns sites com tutoriais mais detalhados sobre a linguagem de programação e a construção de circuitos com Arduino.

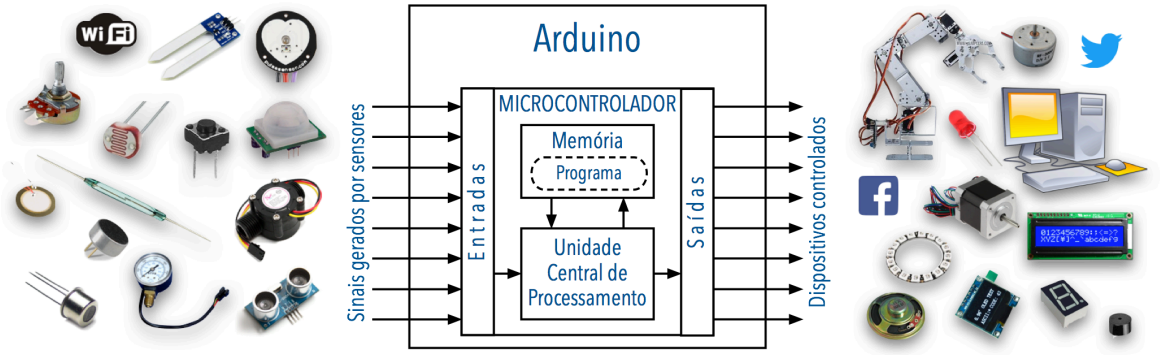
5.2. Arquitetura do Arduino

O circuito do Arduino é composto de um *microcontrolador* programável montado em uma placa, onde é configurado para operar e oferecer acesso seguro aos seus pinos de **entrada** e **saída**. Um microcontrolador é como um mini-computador. Ele tem **memória**, uma unidade central de processamento (**CPU**), **entradas** e **saídas**. Os programas gravados na memória de um microcontrolador controlam os sinais (correntes e tensões) enviadas e recebidas em seus pinos de entrada e saída, permitindo receber sinais de sensores externos e controlar dispositivos.

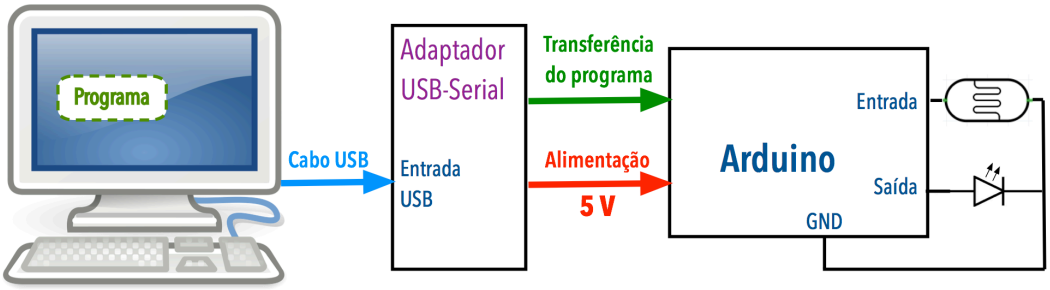
As **entradas** do Arduino recebem **dados**, que podem ser pulsos, tensões e outros sinais elétricos que ele interpreta como dados **digitais** (dois estados lógicos: ligado/ALTO ou desligado/BAIXO) ou **analógicos** (valores que variam). Os geradores desses sinais podem ser chaves, potenciômetros, sensores de luz, som e temperatura, outros circuitos, dispositivos conectados a redes, etc.

As **saídas** também produzem pulsos, tensões e sinais analógicos ou digitais, que podem ser usadas para diversas tarefas, como acender um LED, controlar um motor, ligar ou desligar um circuito, controlar um dispositivo externo, enviar um email.

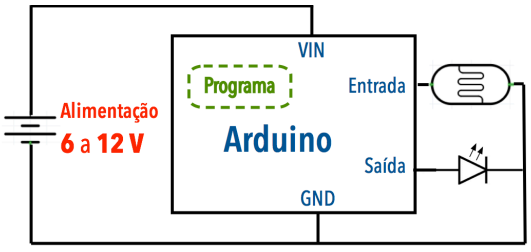
O microcontrolador **processa** os dados de entrada para gerar os dados de saída. Todo o processamento é feito via software, ou seja, através de uma **linguagem de programação**.



Um programa usa **instruções** para ler o **estado** (nível de tensão, nível lógico alto ou baixo) em pinos de **entrada**, e **produzir saídas** (tensão, nível lógico alto ou baixo) nos pinos de saída, que poderão ligar, desligar ou controlar componentes e dispositivos. O programa é um **arquivo de texto** digitado em um computador, em seguida traduzido para linguagem de máquina e depois transferido para o microcontrolador (através de um circuito de comunicação que controla o acesso a pinos de comunicação serial), onde será gravado na **memória** do chip.

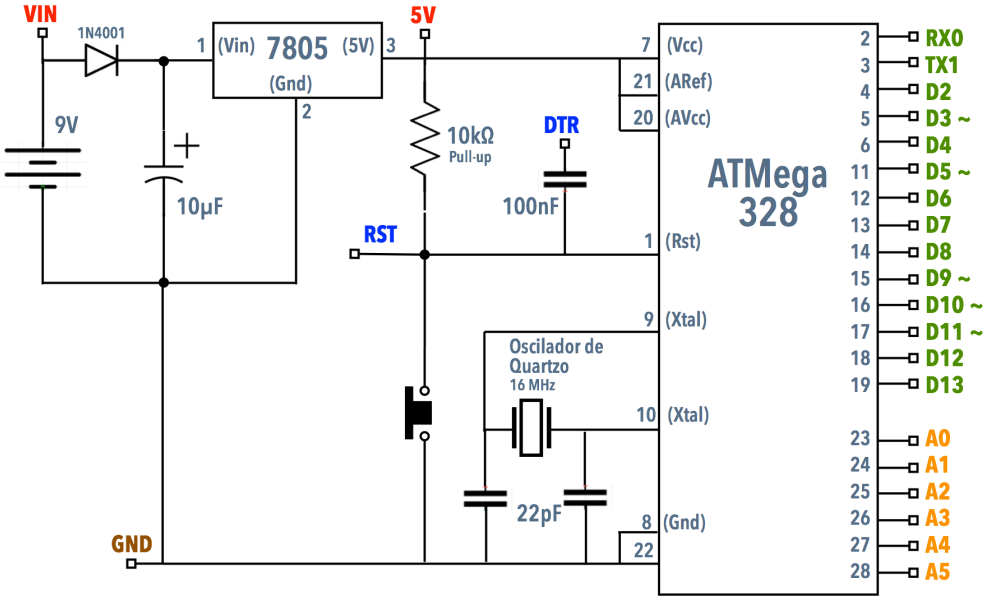


Uma vez gravado o programa, o microcontrolador pode ser desconectado do computador e usado em outro circuito para usar seus pinos de entrada/saída, ser alimentado por baterias, e funcionar de forma independente.

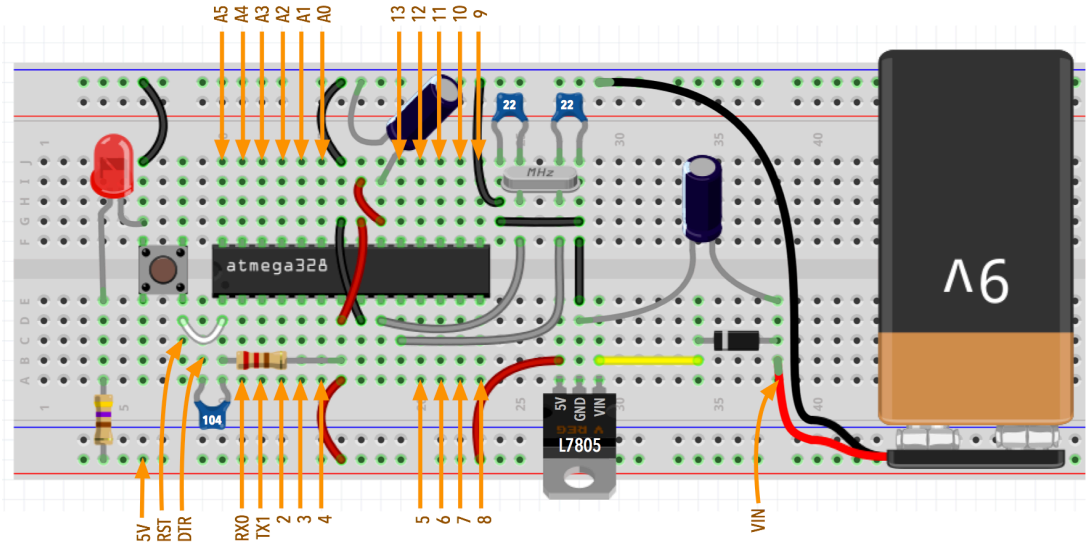


O microcontrolador usado pelo Arduino é um circuito integrado pertencente à **família AVR**. Microcontroladores AVR são populares em drones e impressoras 3D. Existem vários diferentes tipos, com diferentes capacidades de processamento e memória. Nos Arduinos, os mais populares são os **ATMega** e **ATTiny**. Eles têm diferentes formatos e tamanhos. A ilustração abaixo contém três circuitos integrados AVR usados em Arduinos (ATMega328 SMD, ATMega168, ATTiny85):

Um **circuito mínimo** do Arduino é simples e pode ser montado com um ATMega e meio-protoboard. Basicamente é um circuito que serve para **alimentar** o circuito integrado (pinos 7 e 8) e gerar os pulsos de **relógio** que o microprocessador precisa para operar. O ATMega328 do esquema abaixo usa um oscilador de cristal de quartzo nos pinos 9 e 10 para gerar 16 milhões de pulsos por segundo (16 mega Hertz). Este microcontrolador possui 14 pinos digitais (0 a 13) e seis analógicos (A0 a A6) que podem ser usados tanto como entrada ou saída (a finalidade é determinada via software).

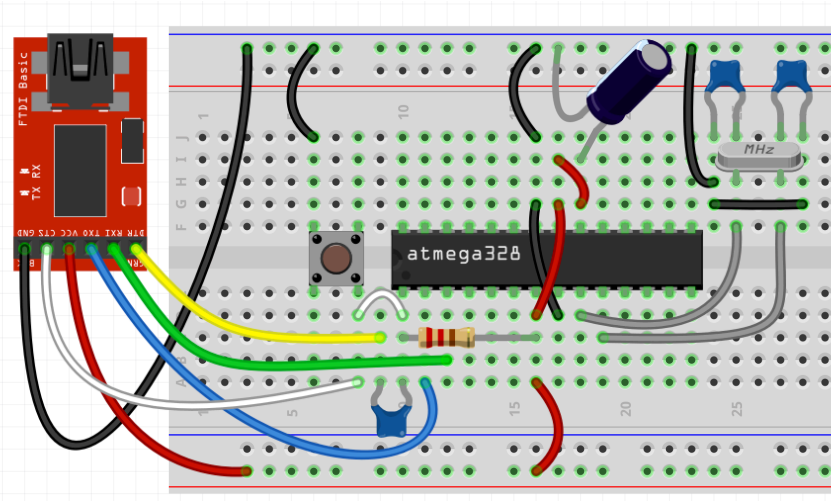


O ATMega precisa ser alimentado por 5V. O circuito acima usa um **regulador de tensão** L7805 que permite alimentar o circuito com 7 a 35V, garantindo que apenas 5V seja enviado ao ATMega. O desenho abaixo ilustra um Arduino montado em protoboard. É igual ao esquema acima, mas acrescenta um LED entre 5V e GND que acende quando o Arduino estiver sendo alimentado.



O circuito acima é apenas uma **plataforma básica** para operar o Arduino. Ele **não faz nada**. Para isto é preciso que tenha na memória um programa contendo instruções dizendo o que deve fazer, e sejam conectados sensores e/ou dispositivos a serem controlados aos seus pinos de entrada e saída.

A **transferência do programa** para a memória do ATMega é feita através dos pinos **RX0** (entrada) e **TX1** (saída). Para realizar essa transferência a partir de um computador é preciso conectar esses (e alguns outros) pinos a um **circuito adaptador USB-Serial**. Este circuito pode ser comprado separadamente como uma pequena placa, e às vezes já vem embutido em alguns cabos adaptadores. O diagrama abaixo mostra um circuito de Arduino mínimo conectado a uma pequena placa adaptadora USB-Serial. Através da placa ele poderá ser conectado a um computador, que também fornecerá a alimentação de 5V):

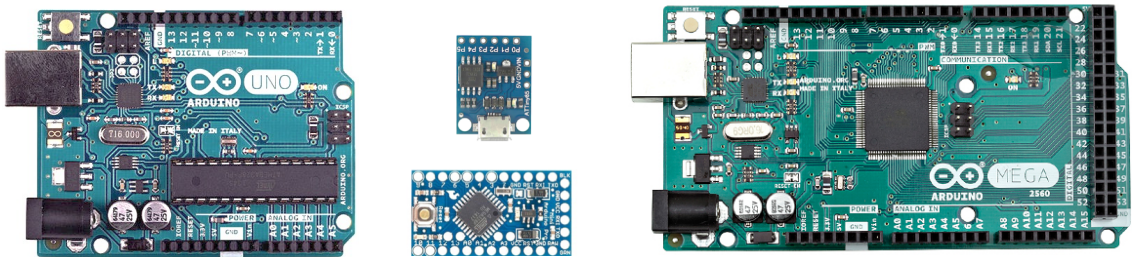


Construir um Arduino desta forma é bom como exercício didático para entender como funciona, mas normalmente usamos **placas prontas**, que já contém um adaptador USB embutido e regulação de tensão (com saídas fixas de 3,3V e 5V), permitindo alimentar o Arduino com tensões variáveis (de 1,8 até 20V, dependendo do modelo). Essas placas vêm em vários tamanhos, são mais práticas, fáceis de usar, e às vezes até mais baratas que montar um circuito Arduino como mostrado acima.

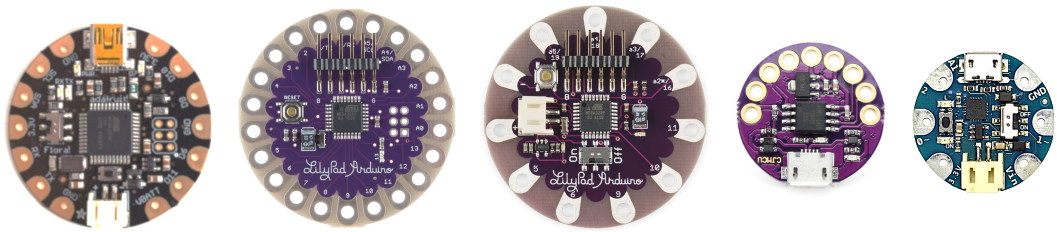
5.3. Placas Arduino

Existem dezenas de diferentes tipos de placas que podem ser chamadas de "Arduino". Todas, sejam as originais italianas ou clones, são baseadas nas famílias de chips AVR ATMega/ATTiny ou similar, e rodam programas escritos para a plataforma Arduino.

O **Arduino Uno** é um dos mais populares, e ideal para fazer protótipos, experimentar e programar. O **Arduino Mega** possui mais pinos, maior capacidade de processamento e memória e é indicado para projetos maiores (ex: impressoras 3D). Existem várias placas minúsculas como o **Arduino Pro Mini**, o **DigiSpark ATTiny**, ou o **Arduino Nano** (incluído no kit). Algumas não tem entrada USB (para economizar espaço e energia) e precisam de um adaptador USB-Serial para que sejam programadas.



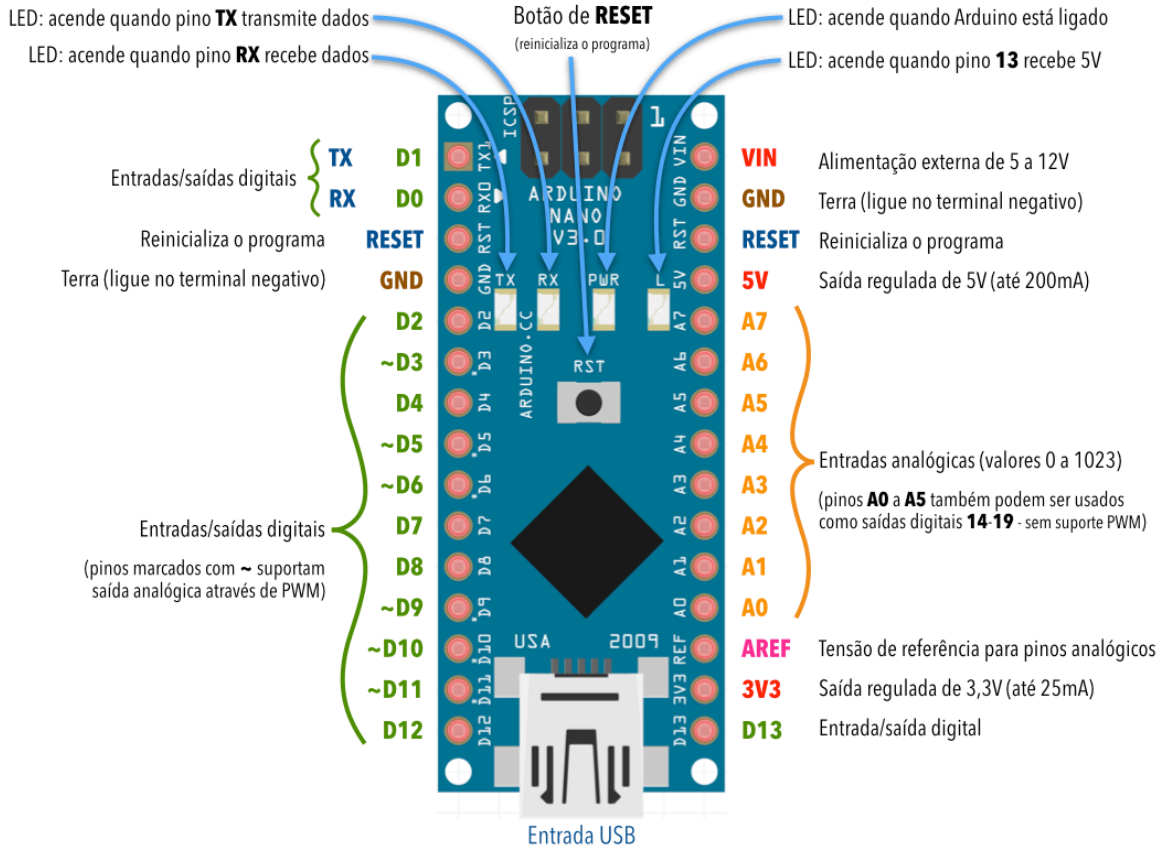
O **Arduino LilyPad** é a principal placa usada em eletrônica para vestir (*wearables*). Nessa linha existem várias placas, a maioria em formato circular com pinos em forma de ilhas que podem ser *amarradas* com linha de costura condutiva, e costuradas em tecido. Exemplos incluem **Arduino Gemma**, **AdaFruit Flora**, **Digispark LilyTiny**.



5.3.1. Arduino Nano

O Arduino que usaremos na oficina é um **Arduino Nano**. Tem dimensões de 43 x 15 mm. Ele possui uma entrada USB que permite a ligação direta a um computador (não precisa de adaptador), e que também fornece alimentação de 5V enquanto estiver conectado. Depois de programado e desconectado do computador, ele pode ser alimentado de forma independente por **7 a 12V** aplicados nos pinos **VIN** (ligado ao positivo da bateria ou fonte) e **GND** (ligado ao negativo).

O Arduino Nano também possui saídas de tensão reguladas em **3,3 V** (Pino 3V3) e **5V** (Pino 5V). Os pinos A0 a A7 são de **entrada analógica** (recebem valores *entre* 0 e 5V), e D0 a D13 suportam **entrada digital** (reconhecem dois valores: 0V – nível lógico BAIXO ou 5V – nível lógico ALTO). A **saída analógica** é *simulada* via PWM apenas através dos pinos digitais D3, D5, D6, D9, D10 e D11. Os outros pinos digitais, e também os pinos A0 a A5, podem operar como **saída digital**. O diagrama abaixo ilustra a pinagem do Arduino Nano:



As especificações de corrente e tensão referem-se ao clone chinês CH340 do Arduino Nano que está incluído no kit, e não ao Arduino Nano original italiano (que são um pouco diferentes).

Algumas observações e cuidados importantes:

- **Os pinos** do Arduino suportam **no máximo 40mA** (ligar em um circuito que deixa passar mais corrente pode queimar o pino). É necessário calcular resistores para limitar a corrente.

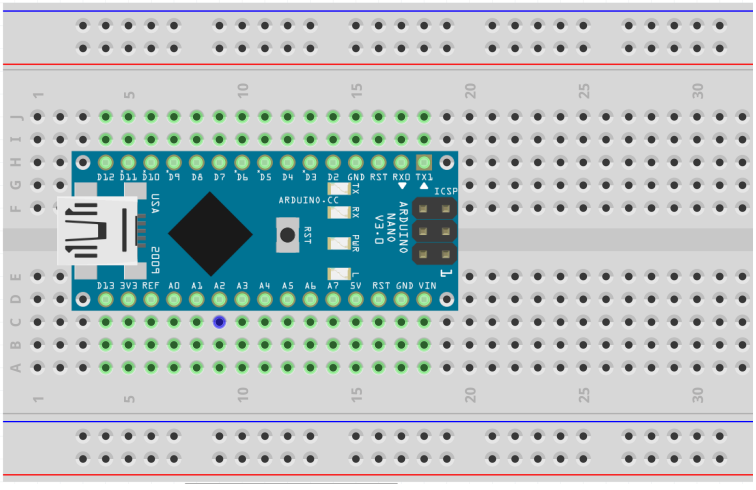
- **O Arduino inteiro fornece no máximo 200mA**. Mas é possível controlar circuitos que consomem bem mais corrente, desde os sinais enviados e recebidos pelos pinos sejam intermediados por circuitos que reduzam as correntes e tensões a níveis suportados. Isto pode ser feito com resistores, capacitores, transistores, relés e outros dispositivos.
- Também é necessário ter cuidado para não curto-circuitar as saídas (**5V** ou **3V3** ligadas diretamente em **GND**). Os pinos analógicos e digitais podem ser ligados diretamente em 5V ou 0V somente **se forem usados como entradas**. Esses valores são tratados como informação (nível lógico ALTO e BAIXO) pelo Arduino. Para usá-los como **saídas**, é necessário configurar essa funcionalidade na programação, e ter o mesmo cuidado que as saídas 5V e 3V3 (não ligar diretamente em GND), além de usar resistores para manter o fluxo de corrente dentro do limite.
- O pino AREF é usado para ajustar a tensão de referência usada para os pinos analógicos. Ela está internamente conectada ao pino 5V, mas pode ser desligada via programação. Ligar uma tensão qualquer neste pino sem primeiro fazer essa alteração via código irá queimar o regulador de tensão (e provavelmente a entrada USB).

Um programa escrito para um tipo de Arduino pode ser usado em outro tipo de Arduino. Pode-se aproveitar programas prontos e fazer pequenas adaptações sem que seja necessário entender todo o código. Portanto, sabendo o mínimo da programação do Arduino, você pode baixar programas da Internet e adaptar para seus circuitos. É preciso garantir que os números de pinos, **declarados no código dos programas**, e os pinos reais, **usados no circuito** estejam de acordo. Em geral qualquer pino digital ou analógico pode ser usado. Eles podem até ser reprogramados. Alguns pinos têm capacidades especiais. Por exemplo, os pinos digitais 3, 5, 6, 9, 10 e 11, no Arduino Nano, permitem gerar saída analógica usando PWM.

Não se preocupe se você não entendeu tudo. São muitos conceitos e é sempre mais fácil entender com um ou mais exemplos. Nas próximas seções mostraremos como instalar e configurar o Arduino, e depois como usá-lo através de vários experimentos. Depois que você fizer os experimentos, releia esta seção. Vários conceitos irão ficar mais claros.

5.4. Preparação e teste do Arduino

Como vamos construir circuitos, e o Arduino Nano não possui soquetes onde podemos inserir terminais de componentes, precisamos usar o protoboard. Encaixe o Arduino com cuidado ocupando a parte central, de forma que possamos ter acesso a todos os seus pinos através dos pinos laterais. Da forma mostrada abaixo, cada pino terá dois a três furos.



O protoboard deve estar livre de outros circuitos (principalmente, não deve haver nenhuma fonte de energia conectado a ele).

Depois de encaixado o Arduino, encaixe uma das pontas do cabo USB no Arduino, e a outra em alguma saída USB do seu computador. O LED PWR do Arduino deverá acender, indicando que ele está sendo alimentado pela porta USB. Para haver comunicação, no entanto, é preciso instalar o driver.

5.5. Instalação do ambiente de desenvolvimento

Para habilitar um computador para programar o Arduino Nano do kit são necessárias duas etapas:

1. Instalar o **driver** (programa que permite a comunicação com o Arduino via porta USB do computador) **do adaptador USB-Serial** (embutido no Arduino).
2. Instalar o programa com o **ambiente de programação** (Arduino IDE).

A **IDE** (aplicação com ambiente gráfico para programação) é distribuída pelo site oficial do Arduino (**arduino.cc**) e existe para Mac, Windows e Linux. Roda de maneira praticamente idêntica nas três plataformas.

O **driver** é mais complicado de instalar, e pode variar dependendo do Arduino usado, se é um clone ou se é um autêntico italiano. O Arduino original (italiano) não requer a instalação de drivers no Mac, mas a instalação ainda pode ser necessária em algumas versões de Windows.

5.5.1. Instalação do driver

O Arduino Nano incluído no kit é um clone e usa um **adaptador USB-Serial chinês** (chip CH341). Para que ele seja reconhecido pelo computador, seja Mac, PC ou Linux, ele **precisa** ter o **driver** instalado antes. O driver é um programa de instalação que deve ser baixado do site do fabricante e executado. Ele não faz nada além disso. A instalação termina depois que o computador for reiniciado. Veja as instruções abaixo. Elas podem ser diferentes dependendo do sistema que você estiver usando.

Windows

Se você usa **Windows 10** baixe o arquivo EXE disponível em

```
http://www.wch.cn/download/CH341SER_EXE.html
```

(clique no botão de Download), execute-o. Você deve ter as permissões para executar este programa no computador, pois ele vai gravar arquivos do sistema. Siga o passo-a-passo (em inglês). Depois é necessário reiniciar o computador para completar a instalação. Quando terminar e reiniciar, pule para a seção seguinte (IDE) para instalar o ambiente de programação.

Mac

Se usa **Mac OS Sierra (10.12)**, baixe o arquivo ZIP em

```
http://www.wch.cn/download/CH341SER_MAC_ZIP.html
```

(clique no botão de Download) e abra o ZIP. Dentro dele há um arquivo **CH34x_Install_V1.4.pkg**. Execute esse arquivo e siga as instruções (em inglês). Depois é necessário reiniciar o computador para completar a instalação. Quando reiniciar, pule para a seção seguinte (IDE) para instalar o ambiente de programação.

Linux

E se você usa Linux baixe o arquivo localizado em

```
http://www.wch.cn/download/CH341SER_LINUX_ZIP.html
```

(clique no botão de Download). Abra o ZIP em uma pasta. Abra uma janela do terminal e execute as linhas abaixo:

```
sudo make
sudo make load
```

Depois siga para a seção seguinte.

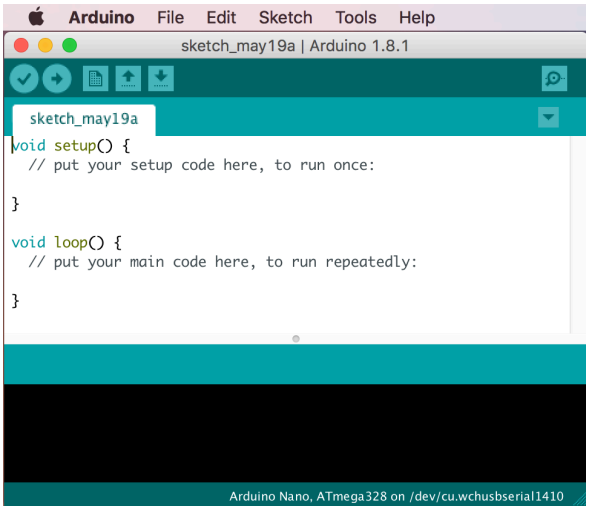
5.5.2. Instalação do ambiente de programação (IDE)

A programação do Arduino é feito na linguagem *Processing*, que é baseada na linguagem C e similar a linguagens de programação populares como C# e Java. Embora possam ser escritos programas bastante complexos usando essa linguagem, é possível fazer muita coisa escrevendo programas bem simples e fáceis de entender mesmo para quem é leigo em programação. Aprendendo o mínimo, você

conseguirá baixar programas disponíveis na Internet e adaptar para rodar com seus circuitos. Para isto, precisamos instalar o ambiente de desenvolvimento integrado (*IDE – Integrated Development Environment*) do Arduino. Baixe o programa de instalação para o seu sistema operacional (Windows, Mac ou Linux) na página

```
https://www.arduino.cc/en/Main/Software
```

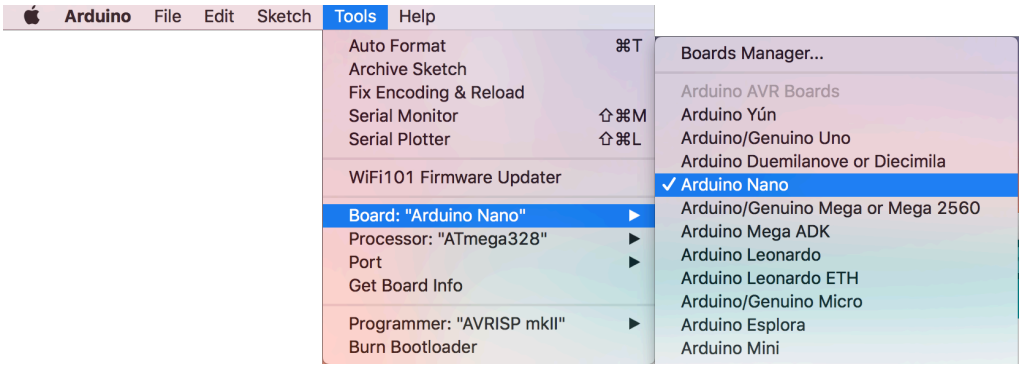
Execute o instalador e siga o passo-a-passo. Depois rode o programa. Ele deverá abrir a janela abaixo:



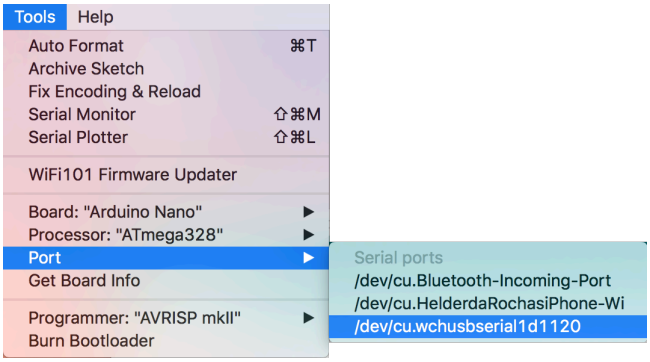
5.5.3. Comunicação do Arduino com o computador

Depois de instalados o driver e o IDE, é preciso ainda **selecionar a placa usada** e a **identificar a porta de comunicação** onde ela está conectada. Isto só precisa ser feito uma vez para cada placa diferente que você usar, mas requer que o Arduino esteja conectado. Portanto, se você ainda não conectou o Arduino a uma porta USB do seu computador, faça isto agora.

Selecione no menu **Ferramentas** (*Tools*), na opção **Placa** (*Board*). Na lista há várias placas. Selecione **Arduino Nano**.



Depois selecione a porta de comunicação. No Windows deve ser algo como **COM4**. No Linux e Mac, um caminho que inicia com **/dev/cu.wchusbserial** (ex: /dev/cu.wchusbserial123456).



Se você usar outro tipo de Arduino posteriormente, terá que modificar esses parâmetros ou a transferência do programa não será possível (o programa apresentará mensagens de erro informando isto). Alguns clones de Arduino são identificados diferentemente (ex: alguns clones chineses de LilyPad são identificados como Arduino Uno) e outros requerem a instalação de bibliotecas externas (que podem ser baixadas) para funcionar.

5.6. Programação do Arduino: fundamentos

Para fazer qualquer circuito com o Arduino, é preciso primeiro programá-lo. Este é o objetivo desta seção onde introduziremos, de forma prática, sua linguagem de programação.

Para que esta introdução seja curta, objetiva e mais simples possível, omitiremos detalhes e abordaremos conceitos de maneira incompleta e às vezes até imprecisa, para focar apenas no essencial necessário para entender os programas que serão construídos.

5.6.1. Estrutura básica de um sketch

Um programa Arduino é chamado de **Sketch**. Ele consiste de uma sequência de instruções escritas em uma linguagem chamada *Processing*. Para ser usado o programa depois precisa ser compilado (traduzido para linguagem de máquina) e transferido para o Arduino. Um sketch é também um arquivo de texto que pode ser gravado no computador, e possui a extensão **.ino**. O menor sketch contém no mínimo a seguinte estrutura (que não faz nada):

```
void setup() {

}

void loop() {

}
```

Se você está usando um Arduino pela primeira vez, e não sabe que programa está em sua memória, é uma boa prática transferir o programa acima para ele. Isto garante que ele não executará nenhuma tarefa que possa danificá-lo.

A estrutura acima possui **dois blocos**, que podemos chamar de **bloco setup()** e **bloco loop()**. A instrução *void setup()* define o bloco setup(), e a instrução *void loop()* define o bloco loop(). Essas instruções são **chamadas automaticamente** quando o Arduino estiver executando, e todas as instruções que forem digitadas **entre as chaves { }** serão executadas.

No programa acima, as chaves estão vazias, portanto quando o Arduino chamar *setup()* e *loop()*, ele **não vai fazer nada**.

Os blocos setup() e loop() funcionam de forma distinta. O bloco **setup()** é **chamado uma vez só**, portanto ele deve conter instruções que serão executadas **uma única vez**. Já o bloco **loop()** é **chamado eternamente**, e deve conter instruções que **repetem para sempre** (até que o Arduino seja desligado ou reiniciado).

Normalmente dentro de **setup()** serão colocadas **instruções de configuração** (por exemplo, especificar a função que um determinado pino irá assumir – se entrada ou saída). Em **loop()** ficam as **instruções que efetivamente programam o Arduino**, por exemplo, mandar nível lógico alto (5V) para pino 4, esperar meio segundo, e depois mandar nível lógico baixo (0V), e repetir isto sem parar.

Antes de programar qualquer coisa, vamos testar a transferência de programas para o Arduino.

Digite o programa acima. Para transferir para o Arduino, clique no ícone  (ou selecione o **menu Sketch/Upload**). Se houver erro, ele aparecerá na caixinha de status na parte inferior do programa, e a transferência não acontecerá. Um erro comum é esquecer de selecionar o modelo de Arduino e sua porta de comunicação (veja a seção 5.5.3). Verifique também se não esqueceu de fechar alguma chave, ou se digitou algo diferente do que foi listado. Os comandos precisam ser escritos **exatamente** como acima (letras maiúsculas e minúsculas são consideradas diferentes na linguagem do Arduino (ex: escrever **LOOP()** ou **Loop()** é um erro).

5.6.2. Sintaxe das instruções

As instruções usadas dentro dos blocos setup() e loop() têm uma sintaxe bem definida. Existem vários tipos. Usaremos principalmente **instruções de uma linha**. Essas instruções **sempre terminam em ponto-e-vírgula** e podem ser classificadas como **comandos** (que mandam o Arduino fazer alguma coisa) ou **expressões** (que calculam valores, guardam dados, etc.).

Comandos

Comandos são formalmente chamados de **funções** ou **métodos**. Eles fazem parte de uma **biblioteca** que define seus nomes e parâmetros. Os **parâmetros** são valores passados entre parênteses, às vezes entre aspas, e separados por vírgulas depois do nome da função. Há comandos que não têm parâmetros (apenas os parênteses vazios). Por exemplo, o comando:

```
delay(500);
```

manda o Arduino esperar meio segundo (500 milissegundos). Há apenas um parâmetro que é o número de milissegundos a esperar. Este outro comando:

```
analogWrite(9, 128);
```

manda o Arduino produzir no seu **pino digital 9** um sinal analógico de nível 128 (o nível varia de 0 a 255, e corresponde a **valores médios** (simulados) de 0 a 5V produzidos com PWM). A instrução, portanto, produz um pulso ligado 50% do tempo que resulta em um valor **médio** de 2,5V no pino 9.

Observe que as instruções terminam sempre em ponto-e-vírgula. Observe também que o "W" em *analogWrite* é maiúsculo (e assim deve ser escrito).

Expressões e variáveis

Pode-se escrever um programa apenas com comandos, mas alguns comandos retornam resultados que precisam ser processados. O processamento é feito através de **expressões**. Existem vários tipos de expressões: aritméticas, lógicas, etc. Expressões frequentemente são formadas por operações. Por exemplo, esta é uma expressão contendo uma **operação de soma** e uma **operação de atribuição**:

```
numero = 3 + 4;
```

O Arduino irá somar 3 com 4 e guardar o resultado na **variável** *numero*. O sinal de **=** é usado para fazer uma **operação de atribuição**, isto é, copiar um valor (o resultado da expressão) para uma área da memória associada à variável. Variáveis são palavras usadas para identificar, guardar e referenciar dados. Elas só guardam dados de um **tipo de dados** específico. A variável acima precisa **declarar** o tipo de dados que pode armazenar, **antes** que seja usada. A declaração é também uma expressão. Então algum lugar **antes** da linha acima, deve haver algo como:

```
int numero;
```

declarando que a variável *numero* é do tipo **int**. A palavra **int** significa **inteiro**, e é usada para declarar variáveis que só aceitam valores inteiros. Não seria possível, por exemplo, guardar um 3.14 na variável *numero*. Para isto ela teria que ser declarada como **float**, que é o nome usado para variáveis com parte decimal. Observe que a declaração da variável também termina em ponto-e-vírgula.

Nos programas em Arduino que faremos nesta introdução, declararemos apenas variáveis do tipo **int** e **float**. Muitas vezes, a declaração e a atribuição ocorrem **na mesma linha**, por exemplo:

```
int pino = 6;
float valor = 3.14;
```

Depois de declarada uma variável, provavelmente vamos querer **usá-la** depois. O **uso** de uma variável pode ser, por exemplo, a inclusão do valor que ela contém em alguma outra expressão ou comando:

```
float raio = 9.5;
float area = valor * raio * raio;
int tempo = 1000;
delay( tempo );
```

A última linha acima é um comando que está usando a variável *tempo*, que contém o valor inteiro 1000, que é passado como parâmetro da função *delay()*.

Alguns comandos **produzem um valor**, que geralmente é resultado do processamento executado por eles. Esse valor geralmente é guardado em uma variável. Por exemplo:

```
int duracao = analogRead(2);
```

O comando *analogRead(2)* é executado, e seu resultado é copiado (via operação de atribuição) para a variável *duracao*. Depois, este valor pode ser usado em outro comando, por exemplo:

```
delay(duracao);
```

A instrução *analogRead(2)* produz um valor (entre 0 e 1023) resultante da **leitura do nível da tensão no pino A2**. Por exemplo, se houver um potenciômetro com os pinos externos ligados entre 0 e 5V, e o pino do meio estiver ligado no A2 do Arduino, e este potenciômetro estiver com o seletor posicionado exatamente no meio, o valor recebido por *analogRead(2)* será 1024/2 ou 512.

Nomes de variáveis

Variáveis não podem ter qualquer nome. Não crie nomes com acentos, hífens, pontos. O ideal é usar nomes curtos e explicativos. Se você quer criar uma variável com mais de uma palavra, você pode distinguir as palavras usando maiúsculas, por exemplo:

```
int numeroDoPino = 6;
```

ou sublinhados:

```
int NUMERO_DO_PINO = 6;
```

Variáveis também não podem usar certas palavras, que são **reservadas**. Exemplos são as palavras *int* e *float*, que têm significado especial para o Arduino. É fácil saber quando uma palavra é reservada, pois ela aparece com uma cor diferente (azulada) no IDE.

Comentários

Nem tudo o que está escrito em um sketch é enviado para o Arduino. Para que os programas sejam mais fáceis de entender pelos humanos que irão lê-lo, é comum que tenham **comentários**. Os comentários **são texto ignorado pelo compilador** (mecanismo da IDE que traduz o programa para linguagem de máquina) e devem ser usados para explicar trechos do programa, ou incluir instruções de como usá-los. Há dois tipos: comentários de linha e comentários de bloco.

Comentários de linha geralmente aparecem antes de instruções, ou logo depois do ponto-e-vírgula, na mesma linha que a instrução. Tudo o que aparece depois do **//** é considerado um comentário. Por exemplo:

```
int PINO_DO_LED = 13;   // este é o pino do LED interno
```

Outra forma de escrever comentários no programa é usar **comentários de bloco**, que consiste em incluir o texto de uma ou mais linhas entre **/*** e ***/**. Use esse tipo de comentário se o que você pretende escrever tem muitas linhas:

```
/*
   Este programa faz um LED piscar.
   Ligue o Anodo do LED no pino 6.
   Ligue o Catodo em um resistor de 470 ohms, ligado a GND.
*/
void setup() { ... }
```

Comentários também são usados para temporariamente ignorar um trecho de código (que você não quer que execute, mas não quer apagar do sketch):

```
void loop() {
    // delay(500);
    delay(1000);    // a linha anterior será ignorada
}
```

Isto é suficiente como uma introdução à linguagem do Arduino. Com o que vimos até aqui já é possível fazer um primeiro programa para piscar um LED.

Experimento 24 – Piscando um LED

Material necessário:

- Arduino Nano + cabo USB + computador
- Um LED de qualquer cor
- Resistor de 220 Ω
- Protoboard, fios e jumpers

Para piscar um LED, é preciso liga-lo em uma saída que alterne entre dois níveis lógicos: **alto** e **baixo**, com um intervalo entre eles. Como precisamos de apenas dois estados, podemos usar um pino de saída digital. Há 14 deles. Podemos usar qualquer um. Vamos escolher o **pino digital 8**, identificado na placa do Arduino Nano com a **indicação D8**.

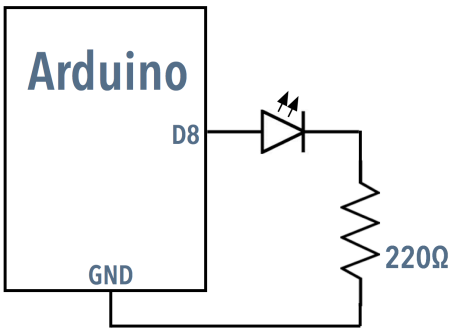
O nível lógico ALTO no Arduino é sempre 5 volts. Precisamos de um resistor para limitar a corrente do LED e temos informações suficientes para calcular seu valor. Se for um LED vermelho, com 2V de queda de tensão:

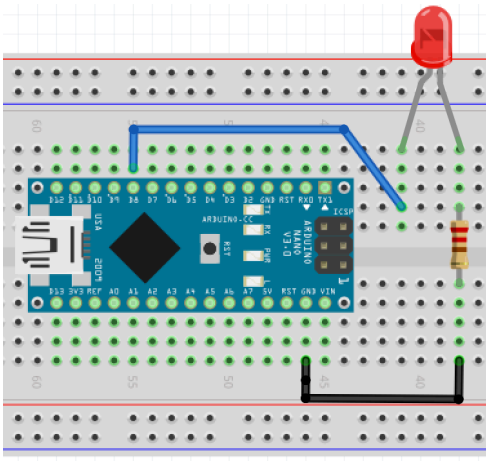
```
R = (5V – 2V) / 0,02 A = 150 ohms.
```

Não temos 150 ohms no kit, mas podemos usar 220 ohms que consome um pouco menos corrente, ou até mesmo arriscar um valor menor (100 ohms) já que ele não vai ficar ligado muito tempo. Podemos também usar um resistor de 100 ohms e trocar o LED vermelho por um de 3V (azul, rosa ou branco).

Antes de montar qualquer circuito, sempre desligue o Arduino do computador (desconecte o cabo USB). Monte o circuito abaixo, verifique as conexões, e **depois** ligue novamente o Arduino ao computador.

A porta USB é quem irá fornecer corrente para o circuito. Se houver um problema no seu circuito e ele tentar puxar corrente demais da porta USB, o computador desligará o acesso e desligará o Arduino (você terá que remover o cabo e reinserir novamente, depois de corrigir o problema).





O resistor ligado ao catodo do LED pode ser conectado a qualquer um dos dois pinos GND disponíveis no Arduino Nano (há um de cada lado). Internamente eles estão ligados entre si.

Agora vamos escrever um programa para piscar o LED. Abra um novo sketch (ícone  ou *Menu File/New*) e preencha os blocos setup() e loop() com as seguintes instruções:

```
void setup()
    pinMode(8, OUTPUT);     // declara pino 8 como uma saída
}

void loop() {
    digitalWrite(8, HIGH);
    delay(500);
    digitalWrite(8, LOW);
    delay(500);
}
```

Clique no ícone  para transferir o programa para o Arduino. Em alguns segundos o LED deverá começar a piscar ficando meio segundo aceso e meio segundo apagado. Experimente mudar o valor de *delay()* para que ele pisque mais rápido, ou que fique mais tempo aceso que apagado.

Explicaremos os comandos usados no programa nas seções a seguir.

5.6.3. Pinos digitais e estados HIGH e LOW

Um **pino digital** permite a entrada e saída de valores digitais correspondentes aos níveis lógicos **ligado** (5V), ou ALTO, e **desligado** (0V), ou BAIXO. Esses dois estados são representados na linguagem do Arduino pelas **palavras reservadas HIGH** (sempre em maiúsculas) e **LOW** (idem). No contexto do circuito, correspondem aos valores de tensão 5V e GND (0V).

(Na verdade **HIGH** e **LOW** são variáveis pré-definidas que, no contexto do programa, contém os números inteiros **1** e **0**, respectivamente.)

Um componente de dois terminais ligado a um pino de saída deve ter o seu outro terminal ligado a uma **referência** de tensão: o pino **GND** (negativo) ou o pino **5V** (positivo). Haverá corrente se houver **diferença de potencial** entre o pino de saída e a referência. Isto significa que, para que haja corrente fluindo por um componente, se ele estiver conectado a um pino que é acionado pelo valor **HIGH**, o outro terminal deve estar conectado a **GND** (é assim que o LED está configurado no nosso exemplo). Se o componente for acionado pelo valor **LOW**, o outro terminal deve estar conectado a **5V**.

É importante observar a polaridade do componente e posicioná-lo de acordo, e também **limitar a corrente**. Um pino e saída do Arduino **não suporta mais que 40 mA**. Ligar um pino de saída diretamente a 5V ou GND sem resistor limitador gera uma corrente muito alta que poderá queimá-lo quando houver uma diferença de potencial no pino.

A instrução pinMode()

Normalmente os pinos operam como **entrada**. Para usar um pino como saída digital é preciso executar uma instrução para declará-lo explicitamente. Isto normalmente é feito dentro do bloco **setup()** através da instrução **pinMode(número-do-pino, função)** (observe o "M" maiúsculo da instrução). Os dois parâmetros informam respectivamente o número do pino e o tipo de função que ele vai exercer (**OUTPUT**, para a função saída):

```
void setup()
    pinMode(8, OUTPUT);     // pino 8* é uma saída
}
```

* **Na placa** do Arduino Nano os pinos digitais são identificados pelo **prefixo D** (D0, D1, D2, D3, etc.) e os analógicos pelo **prefixo A** (A0, A1, A2, A3, etc.) **No programa**, **apenas os números** dos pinos digitais são usados (ex: 0, 1, 2, 3, etc.) Os pinos analógicos podem ser identificados com ou sem prefixo nos comandos que aceitam entradas analógicas.

5.6.4. Saída digital

Para produzir uma saída digital em níveis lógicos (HIGH/5V **ou** LOW/0V, sem valores intermediários) usa-se a instrução **digitalWrite(número-do-pino, nível-lógico)**, dentro de **setup()** (para rodar apenas uma vez) ou **loop()** (para rodar repetidas vezes). O número do pino precisa ter sido previamente declarado como OUTPUT através da instrução pinMode().

Por exemplo:

```
digitalWrite(8, HIGH); // aplica o valor HIGH (5 volts) no pino 8
```

O comando acima transfere 5V (**HIGH**) para a saída digital **D8**. Se no pino D8 houver um LED (alimentado entre o pino **8** e **GND**), ele irá receber 5V, e acender.

Para o LED piscar é preciso fazer o pino 8 ter valor HIGH, depois esperar algum tempo (mantendo o pino neste estado) e em seguida fazer o pino 8 ter valor LOW, esperar mais algum tempo (em que o LED ficará apagado) e repetir a sequência. A repetição acontece automaticamente para instruções digitadas dentro do bloco *loop(),* portanto para piscar o LED serão necessárias apenas quatro instruções:

```
void loop() {
    digitalWrite(8, HIGH);    // aplica 5V no LED+resistor
    delay(500);               // mantém em 5V por 0,5 segundos
    digitalWrite(8, LOW);     // aplica 0V no LED+resistor
    delay(500);               // mantém em 0V por 0,5 segundos
}
// repete tudo ad infinitum
```

5.6.5. Variáveis globais e #define

Quando se tem um programa que usa apenas um ou dois pinos, é fácil lembrar o que está conectado a cada um, mas se muitos pinos estiverem sendo usados o programa pode tornar-se difícil de ler e entender. E se for necessário mudar um componente para outro pino? O número teria que ser alterado em todos os lugares onde foi digitado. Uma solução para este problema é **declarar uma variável** para **identificar** o pino:

```
void loop() {
    int PINO_DO_LED = 8;    // Declarando variável PINO_DO_LED contendo 8
    digitalWrite(PINO_DO_LED, HIGH); // o primeiro parâmetro recebe 8
    delay(500);
    digitalWrite(PINO_DO_LED, LOW);
    delay(500);
}
```

Variáveis globais

Uma variável declarada **dentro das chaves { ... }** de um bloco (ex: *setup()* ou *loop()*) é acessível apenas dentro daquele mesmo bloco. Quando o bloco terminar, ela não poderá mais ser usada (causará erro no programa). Mas às vezes criamos uma variável exatamente para poder usá-la em blocos diferentes. Por isto é comum que a declaração de algumas variáveis ocorra **fora dos blocos loop() e setup()**. Variáveis declaradas fora dos blocos são chamadas de **variáveis globais**, porque elas podem ser usadas em qualquer um dos dois blocos.

No exemplo abaixo, criamos uma **variável global** para guardar o pino do LED:.

```
int LED = 8; // declarada fora de setup() ou loop() – é global

void setup()
    pinMode(LED, OUTPUT);     // reconhecida dentro de setup() – recebe 8
}

void loop() {
    digitalWrite(LED, HIGH);  // reconhecida dentro de loop() – recebe 8
    delay(500);
    digitalWrite(LED, LOW);
    delay(500);
}
```

Outra forma de declarar uma variável global para um pino

Você encontrará alguns programas que declaram variáveis usando o comando **#define** *antes* dos blocos setup() e loop(). Por exemplo:

```
#define LED 8
```

A sintaxe é diferente de uma declaração de variável comum. Não existe o sinal de igual (**=**) e nem ponto-e-vírgula (não pode ter ponto-e-vírgula). **Na prática o resultado é o mesmo.** Usar esta forma ou a outra é uma **questão de estilo**. Não vai alterar o funcionamento do programa. Mesmo que você escolha usar apenas a outra forma, é importante reconhecer essa sintaxe, pois muitos programadores preferem usar **#define** em vez de declarar variáveis globais.

As declarações **#define** geralmente aparecem no início do programa. Elas não podem aparecer dentro dos blocos **setup()** ou **loop()**.

Alteração 24.1 – Usando variáveis

Altere o programa do último experimento substituindo o número do pino por uma variável, e faça o upload novamente. Veja que o funcionamento não muda. Agora mude a posição do LED para o pino 7 no protoboard. Ele não pisca mais, mas você pode abrir o programa, fazer apenas uma alteração (mudando a variável LED para 7) e transferi-lo novamente, que ele voltará a funcionar.

Experimento 25 – Reagindo ao acionamento de chaves liga-desliga

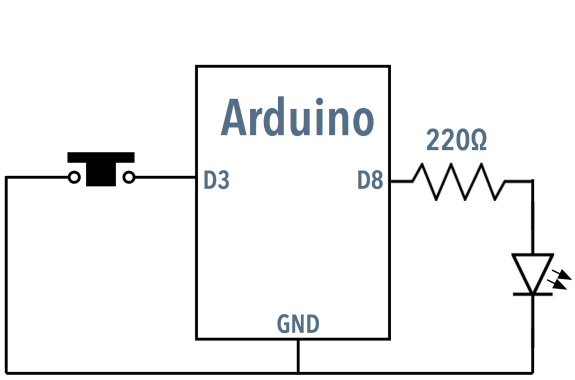
Material necessário:

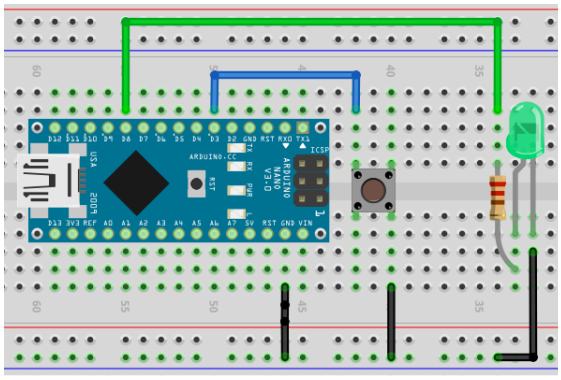
- Arduino Nano + cabo USB + computador
- Um LED de qualquer cor
- Resistor de 220 Ω
- 1 chave táctil de pressão
- Protoboard, fios e jumpers

Em um circuito Arduino, chaves não são usadas para ligar ou desligar um componente *diretamente* (como em circuitos eletrônicos tradicionais) mas para fornecer **um dado de entrada** para o programa, que poderá usá-lo para **tomar decisões** (a decisão pode ser inclusive para ligar ou desligar o componente.) Portanto, uma chave deve ser ligada diretamente a um pino digital de **entrada**.

Não é preciso chamar a instrução pinMode() no setup para configurar entradas (se for usada, deve declarar a funcionalidade do pino como **INPUT**.) Nesse modo, quando a chave estiver fechada ela deverá fornecer ou 5V ou 0V para o pino. Quando aberta, o estado do pino é **indefinido**.

Monte o circuito abaixo (é o mesmo circuito do experimento anterior, acrescentando a chave):





Ligamos uma chave de pressão (normalmente aberta) entre o pino 3 e GND. Quando ela não estiver pressionada, não haverá sinal algum no pino 3 (estado indefinido), mas quando ela estiver apertada, ela fará a conexão entre o pino D3 e GND e seu estado será **LOW**. Usamos uma **expressão condicional** para testar o estado do pino, com a seguinte regra: **se** o estado do pino 3 for LOW, o LED será aceso, caso controário (se for qualquer outro estado – indefinido ou HIGH), o LED será apagado:

```
#define CHAVE 3
#define LED   8

void setup() {
  pinMode(LED, OUTPUT);
}

void loop() {
  int estado = digitalRead(CHAVE);
  delay(10); // espera 10 milissegundos antes de testar
  if(estado == LOW) {
    digitalWrite(LED, HIGH);
  } else {
    digitalWrite(LED, LOW);
  }
}
```

O loop repete continuamente lendo o estado da chave e usando o estado lido para comparar com o valor LOW (0V).

Se a chave estiver aberta, o estado não é LOW. É indefinido, portanto, o conteúdo do bloco *if* é ignorado, mas o bloco *else* é executado. A instrução dentro do *else* mantém o LED apagado, já que fornece 0V (LOW) para a saída 8.

Se a chave estiver apertada, ela faz uma ligação direta entre o pino 3 e GND, fazendo-o ter o estado LOW. Neste caso, a instrução executada muda o estado do pino 8 para HIGH, e o LED acende. Soltando o botão, ele volta ao estado indefinido, e em pouco mais de 10 milissegundos, o estado do pino 3 será testado de novo, desta vez apagando o LED. Portanto, o LED só acende enquanto o botão estiver apertado.

5.6.6. Entrada digital

A instrução **digitalRead(número-do-pino)** serve para **ler o nível lógico de um pino de entrada**. O valor pode ser ALTO (**HIGH**) ou BAIXO (**LOW**). Normalmente HIGH corresponde a 5V e LOW corresponde a 0V (mas na prática o Arduino irá considerar como HIGH qualquer valor de tensão de 3 volts ou mais. Valores abaixo de 3V serão considerados nível lógico LOW.

HIGH e LOW são variáveis que guardam valores inteiros (respectivamente 1 e 0), portanto o valor lido por *digitalRead*() deve ser armazenado em uma variável declarada como **int**:

```
int valor = digitalRead(3);
```

Os pinos digitais são inicialmente configurados como entradas, portanto não é necessário usar *pinMode()* para declará-los como tal. Se for usada deve conter a opção INPUT:

```
pinMode(3, INPUT);      // pino 3 é uma entrada
```

5.6.7. Lógica condicional e bloco if-else

"**If**" significa "se". O bloco condicional **if(*condição*) {}** recebe entre parênteses uma **expressão lógica booleana**, e entre as chaves uma lista de instruções que devem ser executadas **somente se a expressão for verdadeira**.

Expressões lógicas podem ser igualdade (operador ==), diferença (operador !=) e desigualdade (operadores >, <, >= e <=).

Observe que para **testar** a igualdade usa-se um duplo igual ==, já que o sinal de igual isolado é usado como operador de atribuição.

Blocos **if()** devem ser usados dentro de blocos *loop()* ou *setup()*, portanto é uma boa prática, ao escrever programas, **endentar** o conteúdo do bloco para facilitar a leitura do código (ex: digitar quatro espaços antes, para cada novo nível de chaves {...}).

Por exemplo, as instruções que começam com "int" e "if" abaixo estão dentro de loop() e endentadas 4 espaços. A instrução "digitalWrite" está dentro de if, que está dentro de loop, e endentada 8 espaços:

```
void loop() {
    int estado = digitalRead(3);
    if (estado == HIGH) {          // testa se estado é 5V
        digitalWrite(8, HIGH);     // "acende" componente que está no pino 8
    }
}
```

Um bloco **if()** pode ser seguido por um bloco **else {}**, que significa "caso contrário" e executa quando a condição *não* for verdadeira:

```
if (estado == HIGH) {          // somente se o valor de estado for 5V
   digitalWrite(8, HIGH);      // "acende" componente que está no pino 8
} else {                       // caso contrário
   digitalWrite(10, HIGH);     // acende o componente do pino 10
}
```

Alteração 25.1 – Invertendo o estado de acionamento

Altere o programa do último experimento para que ele acione o LED quando:

- O estado do pino **não for** LOW (requer apenas alteração no código).
- O estado do pino for HIGH (que alteração precisará ser feita no circuito?)

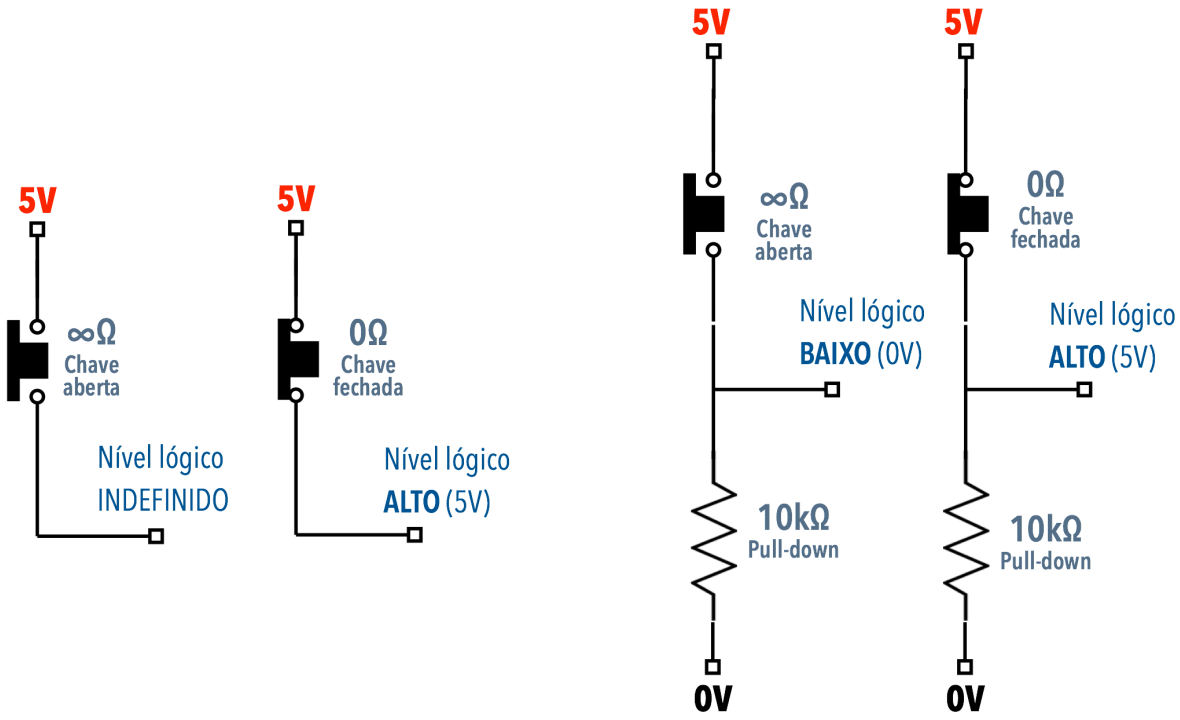
Experimento 26 – Entrada com resistores pull-up

Material necessário:

- Arduino Nano + cabo USB + computador
- Um LED de qualquer cor
- Resistor de 220 Ω
- 2 chaves tácteis de pressão
- Protoboard, fios e jumpers

Há situações em que um programa precisa saber o estado da chave **também quando ela não estiver sendo apertada**. Nesses casos, é preciso conectar um **resistor de pull-up** ou **pull-down** entre o pino e o estado desejado (oposto ao estado quando a chave estiver acionada) **para que o estado inicial seja definido.** Este resistor liga o pino a um **estado inicial**. Se o pino for depois ligado **diretamente** ou através de uma **resistência menor**, a 5V ou GND, haverá um caminho mais curto para a corrente e seu estado será invertido.

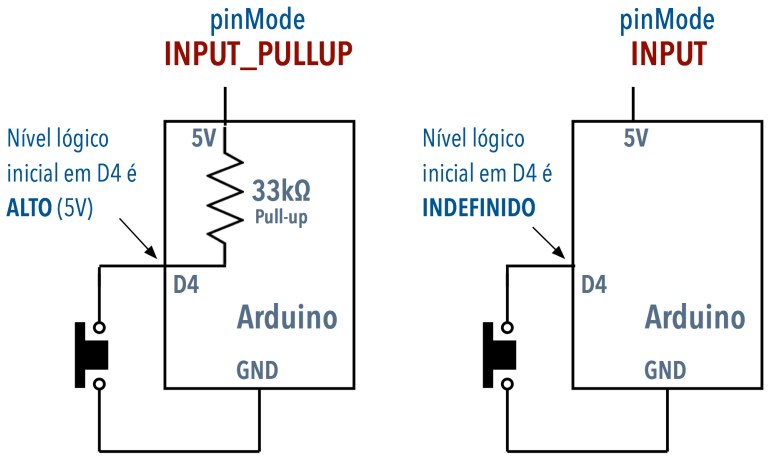
Ligar o pino a 5V ou GND é irrelevante, pois o programa poderá escolher o que fazer em cada caso. Por exemplo, se escolhermos ligar uma chave a 5V, o valor no pino será HIGH quando a chave for pressionada, mas indefinido quando ela estiver aberta (veja ilustração abaixo). Usando um resistor de pull-down (tipicamente de 10k) ligando o pino inicialmente a GND, garantirá ao pino um estado inicial LOW que mudará para HIGH quando a chave for pressionada.



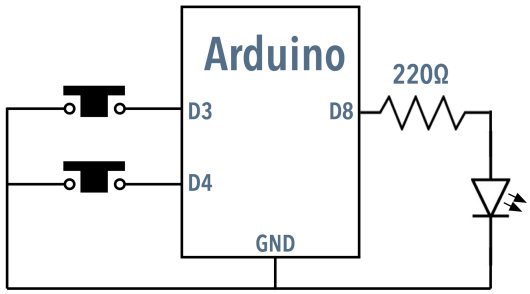
Se usarmos a lógica oposta (conectar com GND/LOW em vez de 5V/HIGH) não precisaremos dos resistores, pois o Arduino **possui internamente resistores de pull-up** (ligados em 5V) **para cada pino**. Esta opção pode ser ativado configurando o *pinMode()* com **INPUT_PULLUP**:

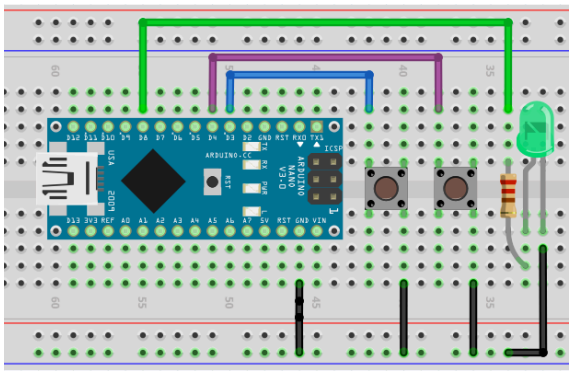
```
pinMode(CHAVE, INPUT_PULLUP);
```

Nessa configuração, o pino terá sempre como estado inicial o nível lógico HIGH. No fechamento da chave, o estado mudará para o nível lógico oposto (LOW). O diagrama abaixo ilustra as duas formas de configurar entradas digitais em Arduino:



Construa o circuito abaixo. Ele apenas acrescenta mais uma chave ao circuito anterior.





Usaremos um programa que depende dos dois estados (ligado e desligado) de cada chave para decidir quando acender o LED.

```
#define PINO_LIGAR    3
#define PINO_DESLIGAR 4
#define LED 8

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(PINO_LIGAR, INPUT_PULLUP);
  pinMode(PINO_DESLIGAR, INPUT_PULLUP);
}

void loop() {
  int acender = digitalRead(PINO_LIGAR);
  int apagar  = digitalRead(PINO_DESLIGAR);

  if(acender == LOW) {
    digitalWrite(LED, HIGH);
  }
  if(apagar  == LOW) {
    digitalWrite(LED, LOW);
  }
}
```

O loop() é executado repetidas vezes e cada vez: 1) o estado de cada pino é lido, e 2) dois blocos condicionais **if** são executados. Se a condição testada for verdadeira, o conteúdo é executado. Se não for, o conteúdo é ignorado.

Os blocos condicionais apenas testam se cada botão está em estado LOW, se não estiver, eles são ignorados. Mas os botões raramente estão no estado LOW. Isto acontece apenas quando forem apertados. O estado normal de cada botão é HIGH, já que estão conectados via pull-up. Se por uma

fração de segundo você apertar qualquer um dos botões, seu estado será momentaneamente LOW e o bloco será executado. O primeiro **if** acende o LED, o segundo **if** apaga. Enquanto nenhum botão está pressionado, o estado anterior é mantido.

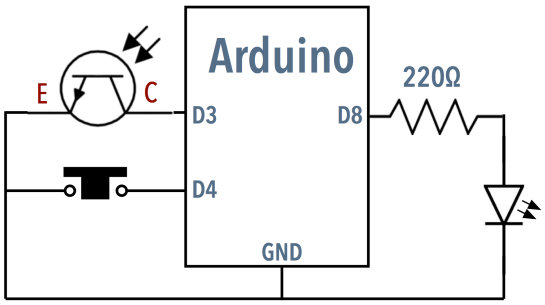
O circuito, portanto, tem uma **memória** que representa o estado do último botão apertado no acendimento ou apagamento do LED. Funciona como um **alternador de estado** (faz o mesmo que o 555 bi-estável do capítulo anterior, mas sem precisar calcular capacitores nem resistores). Se você trocar a chave por um sensor que baixe a tensão no pino 3 a um nível abaixo de 3V, você pode fazer o LED acender com um evento externo, por exemplo, apagar ou acender uma luz, ou bater palmas. O botão em D4 seria usado apenas para apagar o LED. Isto é proposto nas alterações abaixo.

Alteração 26.1 – Substituindo uma chave por um sensor de luz

Material adicional:

- Um fototransistor (TIL 78) ou LDR

Troque a chave em D3 por um LDR ou um fototransistor (esquema abaixo), que se comporta como uma chave fechada quando recebe luz. Um pulso curto de luz é suficiente para acender o LED.

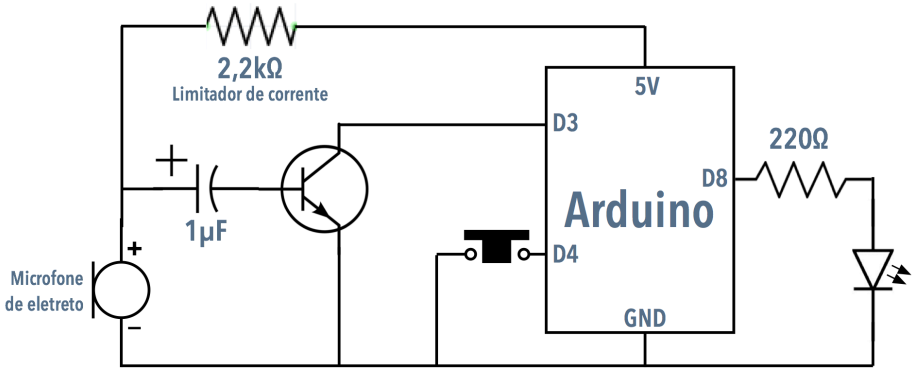


Alteração 26.2 – Substituindo uma chave por um sensor de som

Material adicional:

- 1 microfone de eletreto
- 1 transistor BC549 ou equivalente
- 1 capacitor de 1µF
- 1 resistor de 2k2 Ω

O circuito sensor de som abaixo irá gerar pulsos altos e baixos. Um pulso que faça o transistor conduzir por um instante fará o LED acender. O microfone pode ser alimentado através da saída 5V.



5.6.8. PWM e analogWrite

O Arduino Nano não produz sinal analógico verdadeiro, mas apenas os **simula** através de PWM (*Pulse Width Modulation – veja capítulo anterior*) através dos pinos digitais 3, 5, 6, 9, 10 e 11. PWM permite *simular* valores **médios** de tensão que variam entre os níveis lógicos **LOW** e **HIGH**. Esses valores são representados por número entre **0 a 255** retornado pela instrução **analogWrite()**.

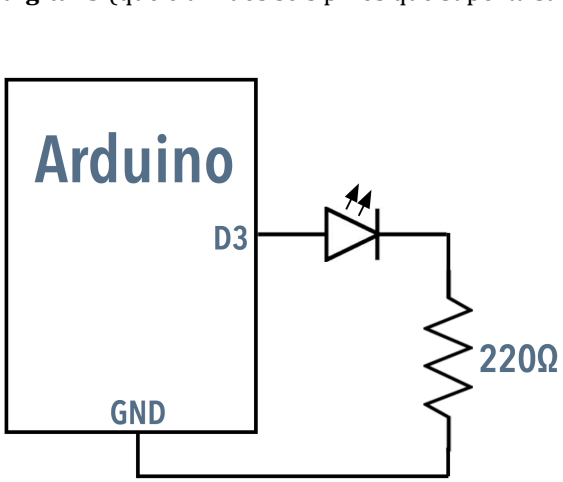
```
analogWrite(5, 64);  // envia 5V 25% do tempo 5V para a saída digital 5
```

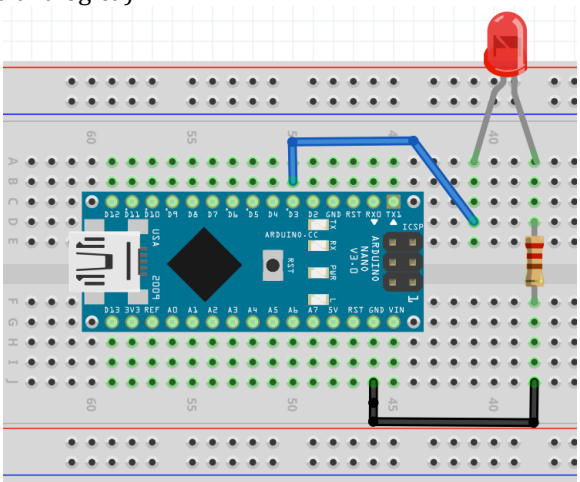
Experimento 27 – Piscando suavemente

Material necessário:

- Arduino Nano + cabo USB + computador
- LED
- Resistor de 220 ohms
- Protoboard, fios e jumpers

Este é uma variação do primeiro experimento que pisca um LED. O circuito é praticamente o mesmo, mas como o pino 8 usado no programa anterior não suporta PWM, **transferimos o LED para o pino digital 3** (que é um dos seis pinos que suporta saída analógica).





O programa também mudou. Em vez de piscar em dois estados digitais, este pisca suavemente, variando entre o totalmente aceso e totalmente apagado.

```
int LED = 3;
int brilho = 255; // inicia com brilho máximo
int direcao = -1;  // 1 = aumentando, -1 = diminuindo

void setup() {
    pinMode(LED, OUTPUT);  // opcional com analogWrite (mas é boa prática)
}

void loop() {
    analogWrite(LED, brilho);
    delay(2);
    brilho = brilho + direcao;
    if(brilho <= 0 || brilho >= 255) {
        direcao = -direcao;
    }
}
```

O programa define duas variáveis (além do pino): **brilho**, que guarda o valor do brilho do LED (entre 0 e 255) e **direcao**, que contém o número 1 ou -1, que será somado ao brilho cada vez que *loop()* for executado, fazendo com que o brilho aumente ou diminua lentamente.

Assim que loop() inicia, o pino onde está o LED recebe o valor analógico 255 (correspondente a 5V). Depois espera 10 milissegundos e executa a instrução

```
brilho = brilho + direcao;
```

Uma expressão de atribuição primeiro executa a expressão do lado direito do "=", substituindo as variáveis por seus valores. Portanto, a expressão executada será calculada da seguinte forma:

```
brilho = 255 + (-1)
```

que irá gravar o valor 254 na variável **brilho**.

No bloco seguinte temos uma expressão condicional **if** que testa se brilho é maior ou igual a zero **OU** se é maior ou igual a 255. O símbolo **||** conecta duas expressões através de uma **proposição lógica OU**. Isto significa que a condição do **if** será verdadeira **se uma ou ambas as expressões forem verdadeiras**. A condição é testada para os valores limite 0 e 255. Quando o valor de brilho chegar a um desses valores, o sinal da variável **direcao** é trocado (se era 1, passa a ser -1; se era -1 passa a ser 1). Assim o brilho que estava diminuindo, passa a gradualmente aumentar, e vice-versa.

5.6.9. Entrada analógica

As entradas analógicas podem ser usadas para ler valores produzidos por sensores e potenciômetros. A instrução é **analogRead(Número-do-pino)**. O valor lido é de **0 a 1023** (correspondente a valores intermediários de tensão entre 0 e 5V). Os pinos de entrada analógica são **A0 – A7**. Eles podem ser identificados no programa com ou sem prefixo A (ex: pode-se usar A0 ou simplesmente 0):

```
int valor = analogRead(3); // lê do pino A3
```

É uma boa prática usar sempre o prefixo A ao declarar pinos analógicos, já que deixa o código mais legível e fácil de entender, evitando confusão com pinos digitais.

Os pinos de entrada analógica não podem ser usados para saída analógica. Eles não suportam PWM. Mas os seis primeiros (A0 a A5) podem ser usados para saída digital (*digitalWrite*), se necessário. Neste caso eles podem ser identificados com o nome analógico *prefixado* (A0, A1, etc.) ou com os números 14 a 19 (que correspondem aos pinos A0 a A5, respectivamente). Ou seja,

```
digitalWrite(A5, HIGH);
```

é o mesmo que

```
digitalWrite(19, HIGH);
```

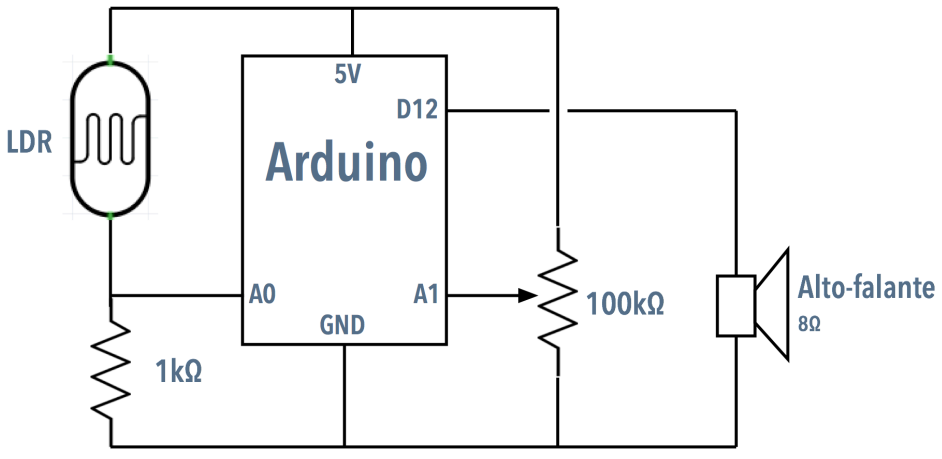
Experimento 28 – "Theremin" com LDR e potenciômetro

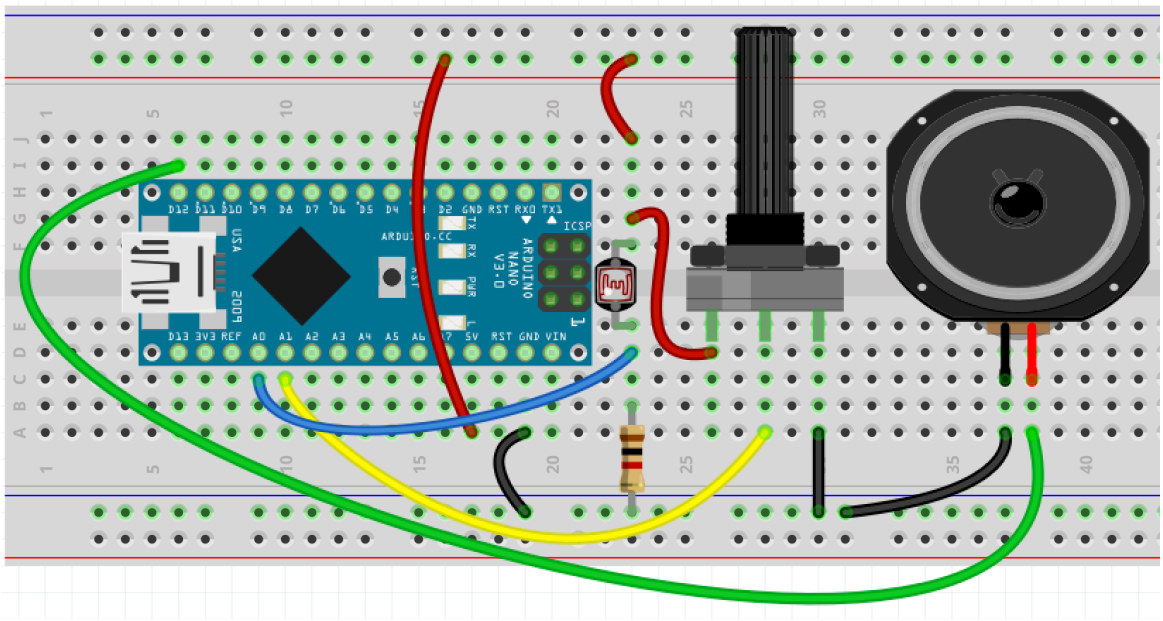
Material necessário:

- Arduino Nano + cabo USB + computador
- Potenciômetro de qualquer valor (ex: 10k, 100k)
- Resistor de 1k ohms
- LDR
- Alto-falante
- Protoboard, fios e jumpers

O circuito abaixo usa duas entradas analógicas. Uma para obter a leitura de luz, e a outra para obter a posição do potenciômetro que será usada para especificar um intervalo de tempo. A quantidade de luz no LDR irá variar a tensão sobre o resistor de 1k (e consequentemente no pino A0). O mesmo ocorre no pino A1, que obtém seu valor do divisor de tensão formado pelo potenciômetro.

A saída PWM varia de 0 a 255, portanto em um circuito perfeitamente calibrado, dividiríamos o valor lido em qualquer uma das entradas analógica por 4 (1024/4 = 256) para usá-lo diretamente na saída.





No programa abaixo dividimos por 4 o valor lido no potenciômetro. Mas a tensão no pino A1 que varia com a luz aplicada ao LDR tem valor indefinido (que provavelmente nunca será zero ou 1023).

```
#define FTE 12
#define LDR A0
#define POT A1

void setup() {}  // pinMode é opcional com analogRead

void loop() {
  int luz    = analogRead(LDR);
  int pausa  = analogRead(POT);
  delay(pausa * .25);
  int tom = luz * luz * luz / 16;  // experimente outros valores
  tone(FTE, tom);
}
```

O comando **tone()** gera uma onda quadrada em frequência especificada como parâmetro. As frequências suportadas (até 65kHz) incluem frequências audíveis (20Hz a 20kHz). O valor proporcional ao cubo do valor lido pelo LDR permite que a frequência varie bastante alternando entre sons agudos e graves.

5.6.10. Serial monitor

O **monitor serial** é uma tela que aparece no computador quando o Arduino está conectado via USB e que permite imprimir texto enviado pelo programa. Ele pode ser usado para depurar programas, ou exibir dados lidos por sensores.

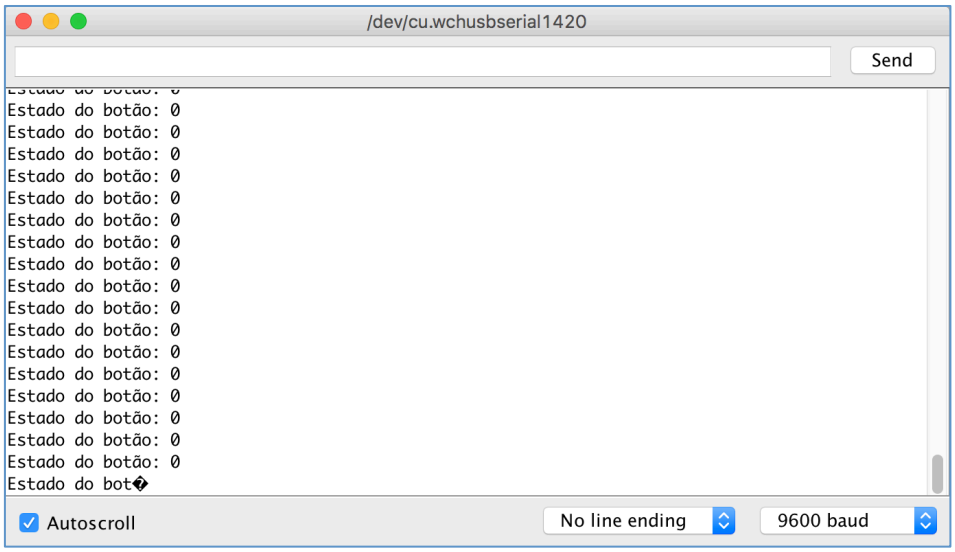
Para usar o monitor serial, é preciso declarar dentro do bloco **setup()** a taxa de leitura. Para depuração simples use o comando:

```
Serial.begin(9600);
```

Dentro de **loop(),** quando quiser imprimir algo no terminal use **Serial.print()** (que imprime texto ou valor de uma variável) ou **Serial.println()** (que imprime uma **nova linha** no final). Para imprimir texto, ele deve ser informado entre aspas. Por exemplo, considere o trecho abaixo:

```
void loop() {
    int estado = digitalRead(9);
    Serial.print("Estado do botão: ");
    Serial.println(estado);
}
```

O monitor serial pode ser aberto clicando no ícone  ou através do menu **Ferramentas** (ou **Tools**) do IDE Arduino, ou. Rodando o programa acima, ele irá imprimir **"Estado do botão: 0"** se o botão estiver no estado LOW, ou **"Estado do botão: 1"** se ele estiver no estado **HIGH**.



É mais interessante usar o monitor serial para ler dados gerados por dispositivos analógicos. Faremos isto no experimento a seguir.

Experimento 29 – Termômetro

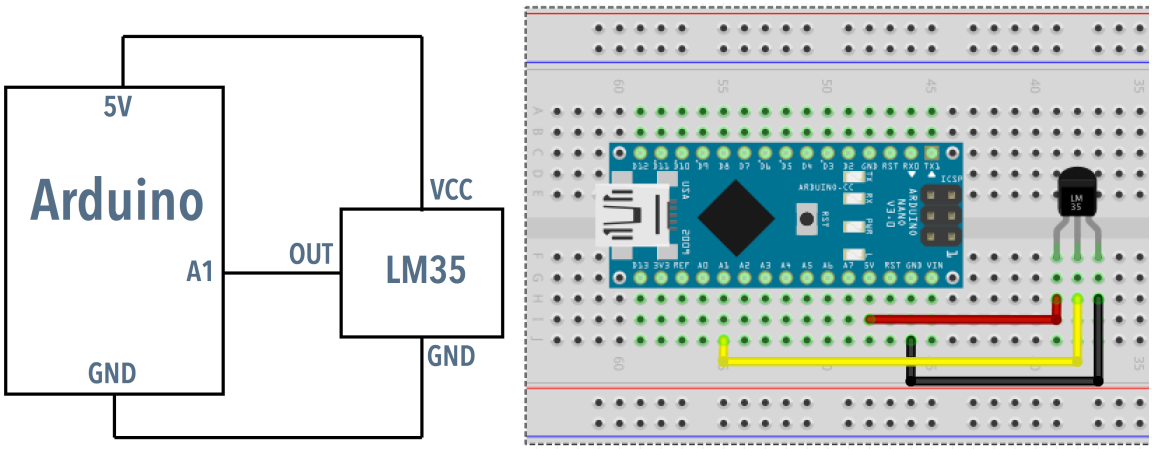
Material necessário:

- Arduino Nano + cabo USB + computador
- Circuito termômetro LM35DZ (veja referência no final da apostila)
- Protoboard, fios e jumpers

Neste experimento conectamos um componente **LM35** ao Arduino para obter a temperatura do ambiente e imprimir o seu valor no monitor serial.

O LM35 é um termômetro de precisão. Ele tem a mesma embalagem (TO-92) que um transistor BC549. Uma vez alimentado com uma tensão entre 5 e 15 V nos seus terminais externos, o terminal central apresentará uma tensão relativa ao terminal negativo proporcional à temperatura ambiente. Aos 25 graus Celsius essa tensão medirá 0,25V, e varia 0,01 volts para cada grau acima ou abaixo com margem de erro de 0,5 graus dentro da faixa 2 a 100 graus Celsius.

Veja a pinagem do LM35 na referência ao final da apostila. Ligue o seu pino VCC no pino 5V do Arduino, e o GND do LM35 em qualquer um dos dois GND do Arduino. O pino central OUT fornecerá a medida da tensão e deve ser ligado a qualquer uma das entradas analógicas (A0 a A7) do Arduino.



No programa abaixo ligamos terminal OUT do LM35 na entrada analógica A1.

```
#define TERMOMETRO A1

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

void loop() {
  int leitura = analogRead(TERMOMETRO);
  float volts = (leitura / 1024.0) * 5.0;
  float celsius = (volts) * 100.0;

  Serial.print("Temperatura: ");
  Serial.println(celsius);

  delay(2000);
}
```

A leitura analógica do Arduino varia de 0 (0 volts) a 1024 (5 volts). Portanto é preciso dividir por 1024 e multiplicar por 5 para obter o valor real em volts que está no terminal OUT do LM35. A leitura em graus Célsius será este valor multiplicado por 100.

Neste programa a temperatura é impressa a cada 2 segundos no monitor serial. O circuito funciona apenas conectado ao computador. Para tornar o termômetro independente do computador teríamos que elaborar um circuito de saída capaz de indicar a temperatura. Poderia ser um par de displays de 7 segmentos, um display de cristal líquido, uma série de leds, etc.

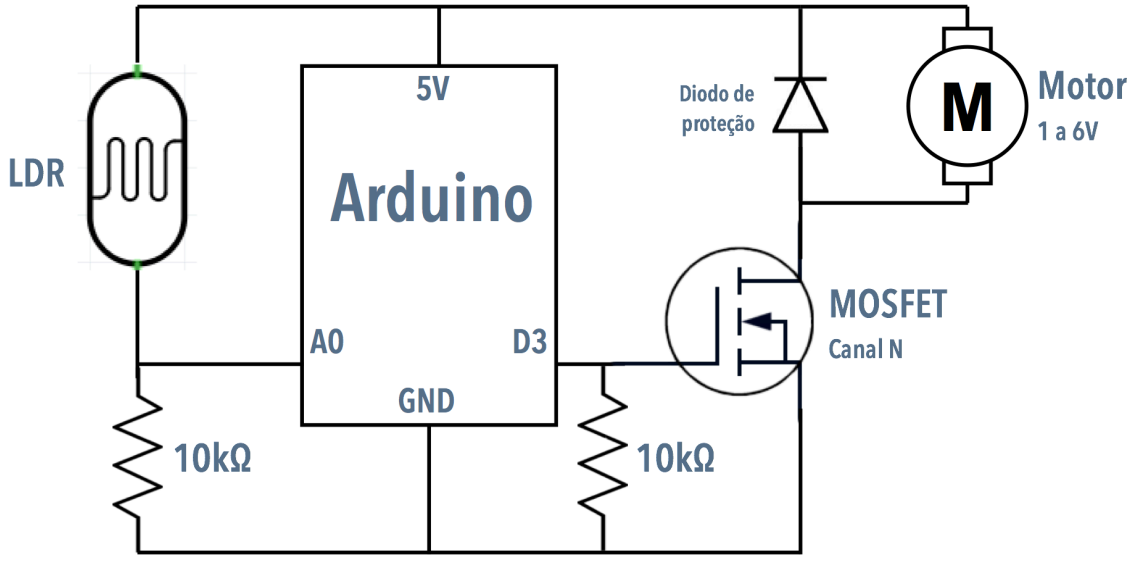
O próximo experimento combina o uso de entradas e saídas analógicas para controlar a velocidade de um motor.

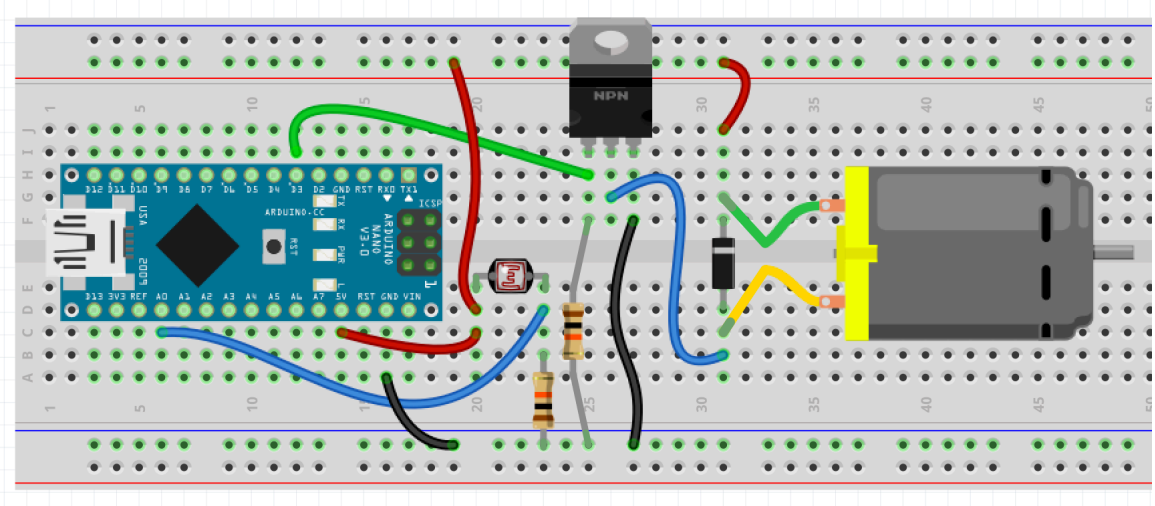
Experimento 30 (extra) – Acelerando e desacelerando o motor com luz

Material necessário:

- Arduino Nano + cabo USB + computador
- Motor de 3V
- 1 diodo de propósito geral (1N4148 ou equivalente)
- 2 resistores de 10k ohms
- LDR
- 1 transistor MOSFET IRL540
- Protoboard, fios e jumpers

Variando a tensão média aplicada no motor podemos fazê-lo acelerar ou desacelerar. Já fizemos isto usando o circuito integrado 555. Com Arduino o circuito é bem mais simples. Monte o circuito abaixo:





O circuito requer uma entrada analógica que irá selecionar a velocidade desejada para o motor, e uma saída, também analógica (PWM), para variar a intensidade da tensão aplicada no motor e assim fazê-lo girar mais rápido ou mais devagar.

A leitura dos dados de entrada para controlar a velocidade do motor poderia ser feita variando um potenciômetro, mas é bem mais interessante usar um sensor de intensidade luminosa como o LDR. O circuito pode ser usado para construir um carrinho que anda quando o ambiente está iluminado, e que para quando entra em um ambiente escuro.

Como um motor pode demandar muito mais corrente que o pino do Arduino é capaz de fornecer, precisamos **isolá-lo**. Uma forma de fazer isto é através de um transistor. Usaremos neste exemplo um **transistor MOSFET de potência,** que consome pouca energia mas suporta bem mais corrente direta que o motor será capaz de exigir. Motores também causam pulsos reversos de corrente quando o motor liga ou desliga. Protegeremos o circuito desses pulsos com um diodo em paralelo com o motor.

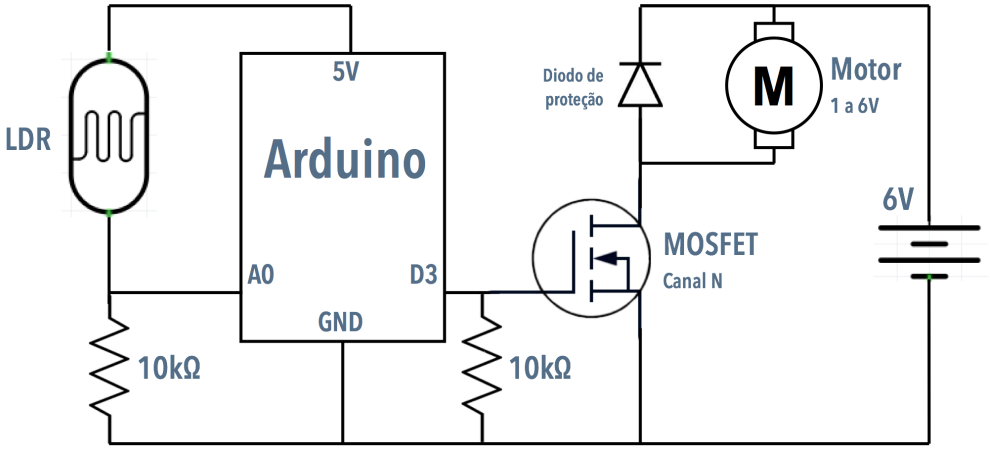
Este MOSFET que usamos só irá conduzir (entre os seus terminais D e S) quando a tensão no terminal de controle (terminal G) tiver mais de 4,5V. Como usamos PWM, o pino 3 irá gerar pulsos de onda quadrada em valores absolutos de 0 ou 5V (os valores intermediários são simulados através da largura dos pulsos, ou seja, através de PWM). Assim o MOSFET irá ligar e desligar muito rapidamente. Quando o tempo ligado aumentar, o motor irá acelerar. Quando o tempo desligado aumentar, o motor irá desacelerar.

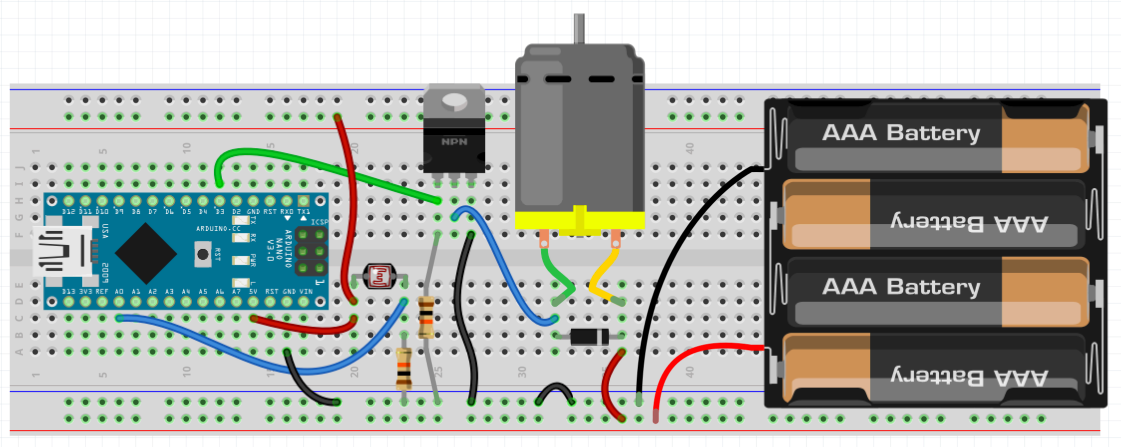
O programa é simples. Consiste na leitura do valor do LDR (0 a 1023) dividido por quatro para que possa ser enviado para a saída PWM (0 a 255):

```
#define PINO_MOTOR 3;
#define PINO_LDR A0;

void setup() {
  pinMode(PINO_MOTOR, OUTPUT);
}

void loop() {
    int velocidade = analogRead(PINO_LDR) / 4;
    analogWrite(PINO_MOTOR, velocidade);
}
```

Alteração 30.1 – Usando uma fonte externa para alimentar o motor

Este experimento funciona adequadamente com o motor distribuído no kit, que consome muito pouca corrente, mas se você usar um outro motor ele poderá exigir demais do Arduino. **Normalmente um motor não deve ser alimentado com a fonte interna de 5V do Arduino**. Ele deve usar uma fonte externa (a fonte pode até ser a mesma usada pelo Arduino, se o motor suportar). Isto não impede que o Arduino continue a controlá-lo. O circuito abaixo ilustra esta configuração (o Arduino pode ser alimentado via USB ou pelo pino Vin):





5.7. Programação do Arduino: funções, listas e bibliotecas

Com o que vimos de programação do Arduino até aqui, que foi basicamente como usar suas entradas analógicas e digitais, já podemos construir todos os circuitos que vimos nas seções anteriores usando transistores e circuitos integrados 555, de forma muito mais simples e sem precisar calcular circuitos RC (resistor-capacitor), e nem mesmo medir tensões e correntes. Não será sempre assim. O conhecimento de eletrônica básica ainda será importante para fazer projetos mais interessantes, porém o Arduino de fato simplifica o uso da eletrônica.

Esta é uma seção opcional. Aqui exploramos alguns tópicos mais avançados de programação com o Arduino que permitirão que você use programas escritos por outras pessoas e faça alterações neles sem necessariamente entender tudo o que fazem. Se você nunca programou antes os conceitos podem parecer complexos. Mas não é preciso entender tudo para fazer os experimentos. Monte os circuitos (que são muito simples) e copie os códigos. Você pode deixar para ler a teoria depois, quando já tiver mais familiaridade com programas em Arduino.

5.7.1. Declarando funções

Os comandos *digitalWrite*, *analogRead*, etc. que chamamos dentro dos blocos *setup()* e *loop()* são chamadas de **funções**. Elas foram **definidas** nas **bibliotecas** do Arduino. Essas bibliotecas são automaticamente incluídas em todos os programas.

Uma função, portanto, para que possa ser **chamada**, precisa ser **definida** em algum lugar. A chamada poderá ser feita dentro do bloco loop() ou setup(). A definição de novas funções poderá ser feita no próprio sketch, **ao lado** dos blocos loop() e setup() que são definições de funções (chamadas automaticamente pelo Arduino).

Portanto, para definir uma função, escolha qualquer lugar antes ou depois dos blocos loop() e setup(), e crie um novo bloco com a estrutura abaixo:

```
void nomeDaFuncao() {
    // coloque aqui as instruções que sua função irá executar quando chamada
}
```

Vejamos um exemplo. Abaixo **definimos** uma função chamada **piscar()**:

```
void piscar() {
    digitalWrite(8, HIGH);
    delay(500);
    digitalWrite(8, LOW);
    delay(500);
}
```

Ela tem o mesmo código dentro de *loop()* do circuito que pisca o LED. Agora podemos **chamar** a função que acabamos de definir como se fosse um comando (terminando em ponto-e-vírgula):

```
void loop() {
    piscar();   // chama a função piscar()
}
```

O código acima funciona igual ao código original que pisca o LED. A vantagem de definir funções é que podemos chamá-las várias vezes, sem precisar repetir o código que ela contém. Isto é mais fácil de perceber se definirmos funções com **parâmetros**.

Quando **definimos** uma função, os parâmetros são **declarados** como variáveis. Como toda variável, o cada parâmetro tem um tipo de dados que faz parte da declaração. Essas variáveis declaradas irão receber valores quando a função for chamada, e podem usar esses valores dentro da definição da função. No exemplo abaixo definimos uma função piscar() contendo dois parâmetros inteiros:

```
void piscar(int pino, int tempo) {
    digitalWrite(pino, HIGH);
    delay(tempo);
    digitalWrite(pino, LOW);
    delay(tempo);
}
```

Observe que as variáveis são usadas pelos comandos que estão dentro da função. Para chamar a função acima precisamos passar para ela dois parâmetros inteiros. O valor do primeiro parâmetro será copiado (atribuído) à variável pino, e o valor do segundo será copiado à variável tempo. Por exemplo, veja a chamada abaixo dentro de loop():

```
void loop() {
    piscar(8, 500);
}
```

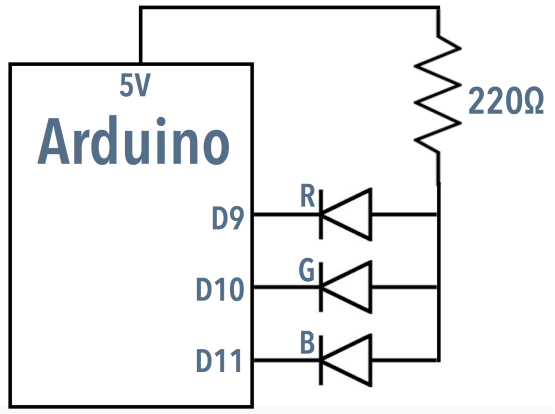
Isto irá copiar o valor 8 para pino, e o valor 500 para tempo. E dentro da função, esses valores serão novamente copiados para funções do Arduino (*digitalWrite* e *delay*). Qual a vantagem disso? Agora podemos chamar a mesma função várias vezes, alterando os parâmetros que passamos para ela. Por exemplo, podemos fazer um loop() que pisca LEDs em pinos diferentes e em tempos diferentes sem precisar escrever 12 linhas de código:

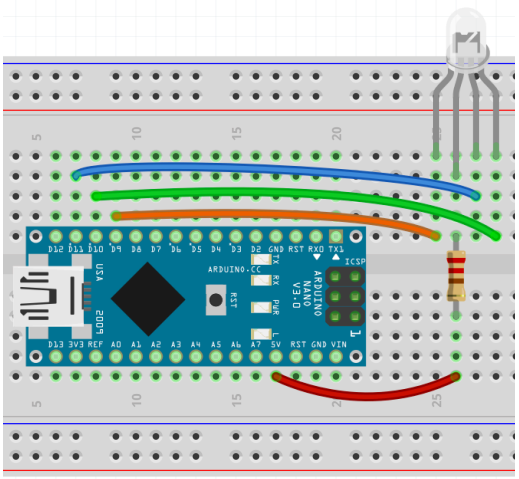
```
void loop() {
    piscar(8, 500);
    piscar(9, 250);
    piscar(8, 1000);
}
```

Experimento 31 (extra) – Definindo funções para controlar um LED RGB

Material necessário:

- Arduino Nano + cabo USB + computador
- LED RGB de anodo comum (ou três LEDs: um vermelho, um verde e um azul)
- Resistor de 220 ohms
- Protoboard, fios e jumpers





O programa abaixo define uma função **cor(r,g,b)** que permite declarar cores usando comandos RGB que são traduzidos em cores de um LED RGB cujos terminais estão conectados a saídas PWM do Arduino. O circuito pode ser usado com 3 LEDs ou um LED RGB. Como o Led RGB usado tem o **anodo comum**, ele foi ligado em 5V e analogWrite(pino, 0) provocará o brilho máximo, já que a diferença de potencial nesse valor é máxima. Para compensar isto sem modificar o circuito podemos subtrair 255 do valor de cada componente de cor. Isto é feito *dentro* da função cor();

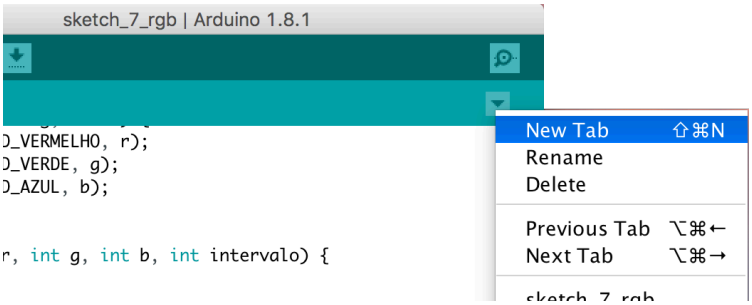
```
#define LED_VERMELHO 9
#define LED_VERDE 10
#define LED_AZUL 11

void setup() {}

void cor(int r, int g, int b) {
  analogWrite(LED_VERMELHO, 255 - r);
  analogWrite(LED_VERDE, 255 - g);
  analogWrite(LED_AZUL, 255 - b);
}

void acender(int r, int g, int b, int intervalo) {
  cor(r, g, b);
  delay(intervalo);
}

void loop() {
  acender(255,0,0,1000);      // vermelho
  acender(0,255,0,1000);      // verde
  acender(0,0,255,1000);      // azul
  acender(190,255,0,1000);    // amarelo
  acender(190,0,255,1000);    // magenta
  acender(0,255,255,1000);    // ciano
  acender(255,255,255,1000);  // branco
  acender(0,0,0,1000);        // apagado

  acender(63,0,0,1000);  // vermelho com 1/4 da intensidade
  acender(0,63,0,1000);  // verde com 1/4 da intensidade
  acender(0,0,63,1000);  // azul com 1/4 da intensidade
}
```

Além da função **cor()**, o programa acima também define uma função **acender(r,g,b,duracao)**, que estabelece a cor (chamando a função *cor()*) e o tempo em que ela ficará acesa. O *loop()* chama várias vezes a função acender() com diferentes parâmetros, para que o circuito exiba uma sequência de cores diferentes, mantendo cada uma acesa por um segundo.

5.7.2. Bibliotecas e arquivos .h (arquivos de cabeçalho)

Suponha que você pretenda usar LEDs RGB em vários circuitos diferentes e queira reusar as funções criadas no experimento anterior. Uma maneira de fazer isto é recortar e colar o texto no outro programa. Mas há alternativas melhores. Uma delas é armazená-las em um arquivo separado que possa ser incluído em outros programas: **um arquivo de cabeçalho**, que tem a extensão **.h**

Para criar um arquivo de cabeçalho no IDE do Arduino, clique no menu que aparece logo abaixo do ícone do monitor serial, e depois selecione "New Tab":



Depois escolha um nome para o arquivo. Por exemplo, **"ledsrgb.h"**, e grave-o.

Agora vamos transferir as funções *cor()* e *acender()* para o arquivo *ledsrgb.h*. Recorte-as do sketch, e cole as duas funções abaixo no arquivo *.h*:

```
void cor(int r, int g, int b) {
  analogWrite(LED_VERMELHO, 255 - r);
  analogWrite(LED_VERDE, 255 - g);
  analogWrite(LED_AZUL, 255 - b);
}

void acender(int r, int g, int b, int intervalo) {
  cor(r, g, b);
  delay(intervalo);
}
```

Como a função cor() **depende** de variáveis que ainda estão no arquivo original, precisamos redefini-las localmente (no arquivo .h) e depois copiar os valores. Acrescente o seguinte código:

```
int pino_R;  // foram declaradas sem valor inicial
int pino_G;
int pino_B;

void configurarRGB(int pr, int pg, int pb) {
  pino_R = pr;  // o valor inicial passado quando esta função for chamada
  pino_G = pg;
  pino_B = pb;
}
```

E altere a função cor() para que ela use os novos valores:

```
void cor(int r, int g, int b) {
  analogWrite(pino_R, 255 - r);
  analogWrite(pino_G, 255 - g);
  analogWrite(pino_B, 255 - b);
}
```

A função *configurarRGB()* **será chamada a partir do sketch original**, para passar os valores dos pinos ao arquivo .h. Para incluir o arquivo .h no seu sketch, use uma expressão **#include**:

```
#include "piscaled.h"
```

(observe que ela **não tem ponto e vírgula no final** – é uma diretiva como #define).

Dentro do *setup()* do seu sketch, chame a função *configurarRGB()* passando as variáveis que contém os números dos pinos. Desta forma, os valores serão copiados para as variáveis do arquivo .h:

```
void setup() {
    configurarRGB(LED_VERMELHO, LED_VERDE, LED_AZUL);
}
```

Agora é possível rodar o programa alterado, que consiste de dois arquivos, e ele funcionará igual, mesmo não contendo a definição de acender(). Se você quiser usar piscaled.h em outro sketch, precisa apenas copiá-lo para a pasta onde está o arquivo .ino do sketch, declarar o #include e chamar o configurarRGB(). Depois disso você pode chamar as funções cor() e acender() como se elas tivessem sido definidas localmente.

Se você achar mais fácil, pode baixar o projeto contendo os dois arquivos usados neste exemplo a partir do repositório GitHub disponível na Internet. Veja o *link* na introdução da apostila.

Experimento 32 (extra) – Usando bibliotecas para produzir notas musicais

Neste experimento usaremos uma mini-biblioteca (arquivo .h) de notas musicais para fazer o Arduino executar uma música.

Material necessário

- Arduino Nano + cabo USB + computador com conexão à internet
- Alto-falante
- Protoboard, fios e jumpers

O comando **tone()** gera pulsos de onda quadrada na frequência passada como parâmetro. Por exemplo, pra produzir um lá central usado para afinação, podemos mandar para um pino PWM que esteja ligado a um alto-falante um comando:

```
tone(440);
```

Para tocar uma música precisamos saber a frequência de cada nota, mas o ideal seria poder simplesmente chamar as notas pelo nome.

Existe um arquivo .h para isto. Ele atribui nomes de variáveis para cada uma das 88 notas do piano, guardando a sua frequência. É um programa de domínio público que faz parte dos exemplos da IDE do Arduino. Este é um trecho:

```
...
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
...
```

Ou seja, usando-o podemos mandar comandos:

```
tone(NOTE_A4);
```

e fazer soar um lá sem precisar lembrar da sua frequência.

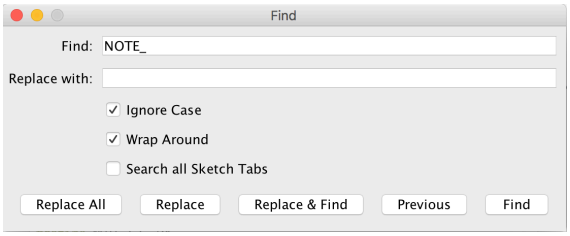
NOTE_C3 corresponde ao terceiro Dó do piano. **NOTE_A4** ao quarto lá, e assim por diante. O comando **noTone(PINO)** desliga o sinal no pino (aplica valor LOW, 0V).

Criaremos o arquivo .h localmente. Abra um sketch novo, e em seguida abra um tab (como mostrado acima no exemplo sobre arquivos .h), e cole nele conteúdo do arquivo abaixo:

https://www.arduino.cc/en/Tutorial/ToneMelody?action=sourceblock&num=2

Grave o arquivo como **notas.h**.

Vamos alterar o nome das variáveis para que fiquem mais curtos e seja mais fácil digitar notas. Com o tab notas.h aberto, digite Control-F (Command-F no Mac) ou selecione o item de menu Edit/Find. Abrirá a janela abaixo.



No campo "Find:" digite NOTE_ e clique no botão Replace All. Isto removerá o prefixo NOTE_ de todas as notas. O resultado deve ficar da forma abaixo:

```
#define B0  31
#define C1  33
#define CS1 35
#define D1  37
#define DS1 39
#define E1  41
...
```

Agora acrescente mais uma definição, em uma linha antes de B0:

```
#define ZZ 0
```

Esta será a nota de pausa (frequência zero) que não produzirá som algum.

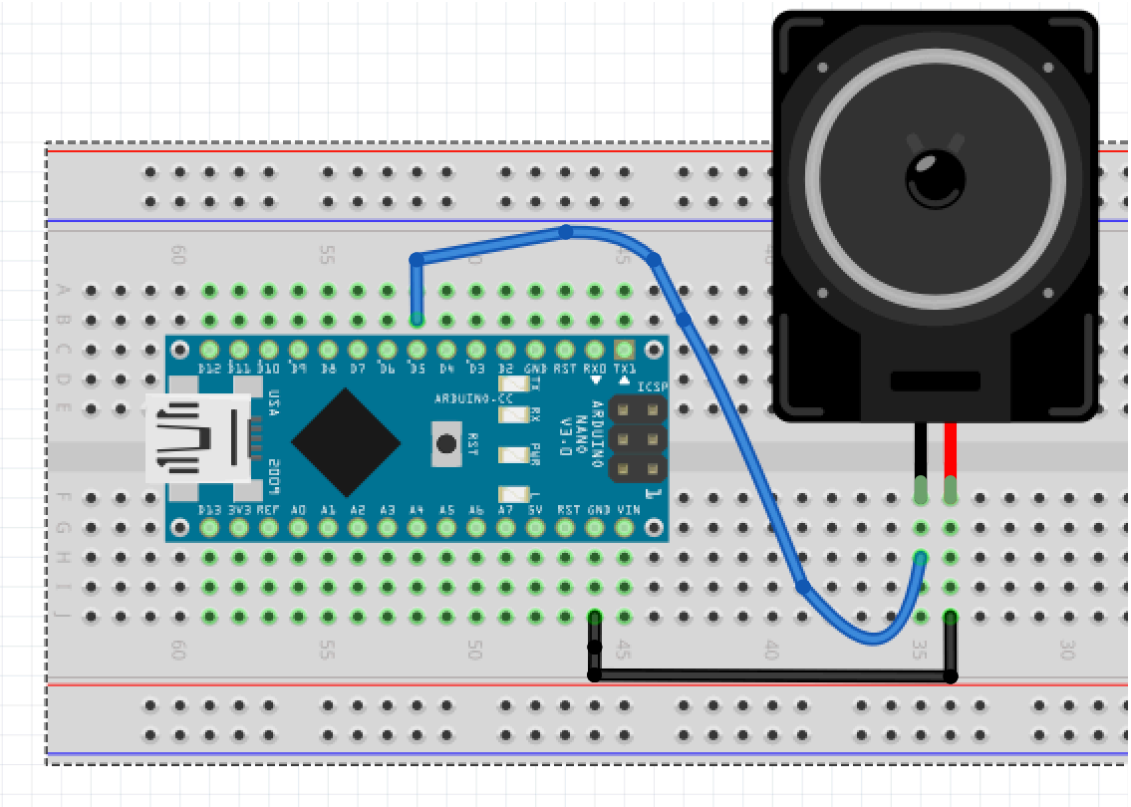
Digite o programa abaixo. Ele foi adaptado dos exemplos do Arduino ("toneMelody" criado por Tom Igoe). Ele usa essa biblioteca de notas musicais para tocar uma pequena música. Nesta versão tocamos uma música diferente. As notas da música estão dentro do bloco melodia[], e os tempos correspondentes de cada nota em durações[]:

```
#include "notas.h"

int ALTO_FALANTE = 5; // coloque alto-falante ou buzzer entre pino 5 e GND
int NUM_NOTAS = 8;    // número de notas incluídas em melodia[]

int melodia[] = {
  G4, G4, G4, DS4, AS4, G4, DS4, AS4, G4,
  D5, D5, D5, DS5, AS4, FS4, DS4, AS5, G4,
  G5, G4, G4, G5, FS5, F5, E5, DS5, E5, ZZ,
  GS4, CS5, C5, B5, AS5, A5, AS5, ZZ,
  DS4, FS4, DS4, AS4, G4, DS4, AS5, G4};

// duração: 1 = semibreve, 2 = mínima, 4 = seminima, 8 = colcheia, etc.:
int duracoes[] = {
  4, 4, 4, 6, 8, 4, 6, 8, 2,
  4, 4, 4, 6, 8, 4, 6, 8, 2,
  4, 6, 8, 4, 6, 8, 16, 16, 8, 8,
  8, 4, 6, 8, 16, 16, 8, 8,
  8, 4, 6, 8, 4, 6, 8, 2
};

void setup() {
  for (int nota = 0; nota < NUM_NOTAS; nota++) {
    int duracao = 1000 / duracoes[nota]; // divide 1000 por cada duração
    tone(ALTO_FALANTE, melodia[nota], duracao);
    int pausa = duracao * 1.30;
    delay(pausa);
    noTone(ALTO_FALANTE);
  }
}
void loop() {}
```

O programa roda apenas uma vez no bloco setup(). Se quiser que ela toque sem parar, você pode transferir todo o código para o bloco loop(), incluindo um delay(1000) no final para que haja um intervalo entre as execuções.

O programa usa um bloco de repetição **for()** para executar cada nota (será explicado mais adiante.)

Construa o circuito abaixo (que talvez seja o circuito mais simples criado no curso), ligando um dos fios do alto-falante no pino 5, e o outro terminal em GND.



Logo após a transferência, a música começará a tocar. Experimente criar outras músicas usando essa biblioteca.

O código-fonte deste experimento (o sketch e arquivo **notas.h** já alterado) também está disponível para download em repositório GitHub (veja link na introdução da apostila).

5.7.3. Listas

O código do programa anterior possui várias estruturas que ainda não vimos. Uma delas é a **lista** (também chamado de *array*, ou vetor). Há duas listas no código do experimento anterior, representadas pelas variáveis **durações[]** e **melodia[]**. Listas são declaradas com seus itens entre chaves, separados por vírgula. O tipo de dados se refere ao tipo de cada elemento da lista:

```
float precos = {23.56, 9.99, 11.99, 25.49};
```

Cada item da lista é referenciado por um numero que é seu índice. A contagem começa em zero, portanto, na lista acima, os índices são 0, 1, 2 e 3. Eles são usados para referenciar itens da lista. Por exemplo, **precos[1]** é 9.99 e **precos[3]** é 25.49.

As duas listas do programa do experimento anterior têm 8 elementos, e seus índices variam de 0 a 7.

5.7.4. Repetição com for

O bloco **for** é uma estrutura de repetição. Consiste de uma declaração entre parênteses e um conteúdo (a ser repetido) entre chaves:

```
for( declaração ) { /* conteúdo a ser repetido */ }
```

A declaração tem **três partes** (separadas por ponto e vírgula):

- **O valor inicial** de uma variável
- A condição **envolvendo a variável** que deve ser verdadeira enquanto o bloco é repetido
- Uma instrução para **mudar o valor da a variável**

Blocos de repetição for() são frequentemente usados com listas. Para imprimir no monitor serial cada um dos itens da lista **precos** (de 4 elementos) acima, pode-se usar o seguinte **for()**:

```
for(int i = 0; i < 4; i = i + 1) {
    Serial.println( precos[i] );
}
```

A cada passada no for(), o valor de **i** será diferente. Iniciará em zero, e será 1, depois 2, depois 3. Na próxima passada **i** é 4, a condição se tornará falsa, e o bloco será encerrado.

No código do experimento anterior, o for() incrementa o índice (variável **nota**) usando **nota++**. Isto é o mesmo que fazer:

```
nota = nota + 1;
```

O for() executa **apenas uma vez** a primeira parte da declaração (inicialização da variável que controla a repetição), mas **a cada repetição** a condição da segunda parte é testada, o bloco entre chaves é executado, e por fim a expressão da terceira parte é executada, nesta ordem.

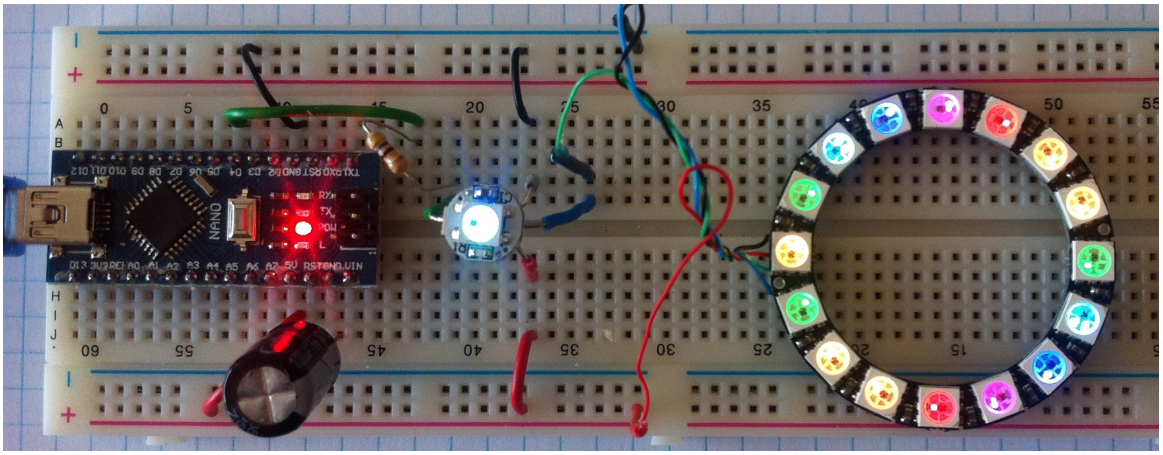
É importante que o valor da variável mude de forma que faça o **for()** terminar em algum momento (a menos que você queira um loop infinito – mas para isto você já tem a função *loop()* que é mais simples). Normalmente as expressões usadas na terceira parte do for() são apenas para incrementar (somar um) ou decrementar (subtrair um) da variável de controle.

Experimento 33 (extra) – Usando LEDs RGB endereçáveis

Neste último experimento iremos programar o Arduino para que ele execute uma sequência luminosa em um **LED endereçável WS2812**. Baixaremos as bibliotecas e os programas para executar as sequências da Internet através da IDE do Arduino.

LEDs WS2812 são pixels RGB frequentemente usados em painéis digitais coloridos de alta-definição. Eles também são muito populares em produtos de *wearables* (eletrônica para vestir), usados em roupas, jóias e calçados. Geralmente eles são distribuídos em conjuntos contendo vários LEDs, organizados em matrizes quadradas, sequências circulares, fitas e painéis flexíveis.

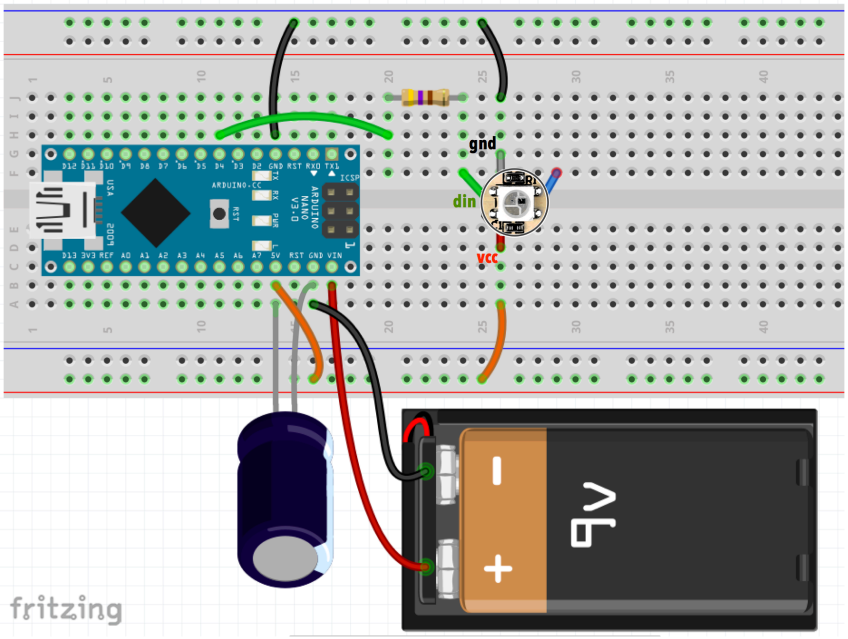
A foto abaixo mostra um circuito usando dois conjuntos de LEDs WS2812 (1 + 16) piscando em uma sequência programada no Arduino. No kit foi incluído um LED WS2812.

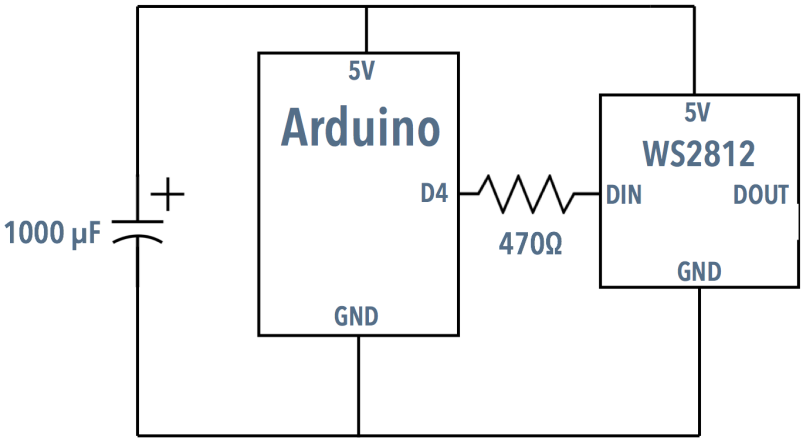


Material necessário

- Arduino Nano + cabo USB + computador
- LED RGB endereçável WS2812
- Resistor de 470 ohms
- Capacitor de 100uF
- Protoboard, fios e jumpers

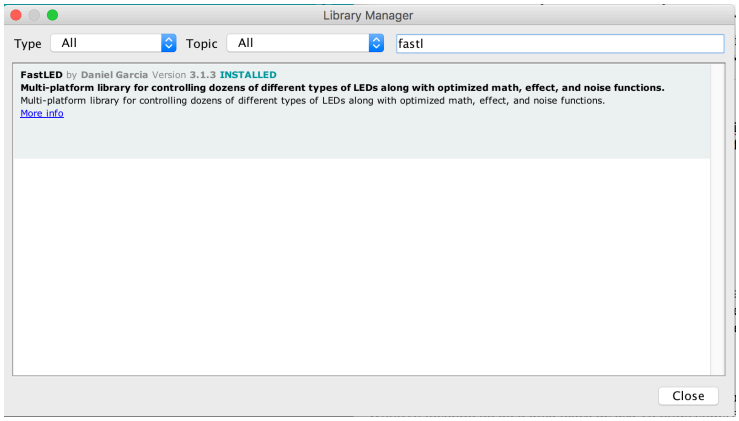
Monte o circuito abaixo. É necessário soldar terminais no LED WS2812 distribuído no kit. Veja como fazer isto no tutorial de soldagem no final da apostila. Você também pode adquirir LEDs WS2812 montados em placas com vários LEDs e adaptados para uso em projetos de eletrônica para vestir.





Existem várias bibliotecas para usar LEDs endereçáveis. As mais populares são as bibliotecas **NeoPixel** e **FastLED**. Todas têm diversos programas de exemplo que você pode usar imediatamente, fazendo poucas alterações. Vamos instalar uma delas e rodar seus exemplos.

Selecione o menu **Sketch/Include Library/Manage Libraries** no IDE do Arduino. Você verá a janela abaixo. No campo de pesquisa digite **FastLED**, e a janela filtrará a biblioteca FastLED que usaremos para programar os LEDs:



Clique no botão **Install**, que aparece do lado direito. Quando a instalação terminar, você poderá abrir os exemplos que usam a biblioteca selecionando o menu **File/Examples**. No final há uma seção **"Examples from Custom Libraries"** e você encontrará **FastLED**.

Abra o exemplo **Blink**. Verifique que a **variável NUM_LEDS** contém o valor **1** (um LED), e altere **DATA_PIN** para **4** (que é o pino onde conectamos o LED). Depois faça upload para o Arduino. O LED deverá piscar na cor vermelha. Você pode mudar a cor alterando o código em loop(), por exemplo, troque por:

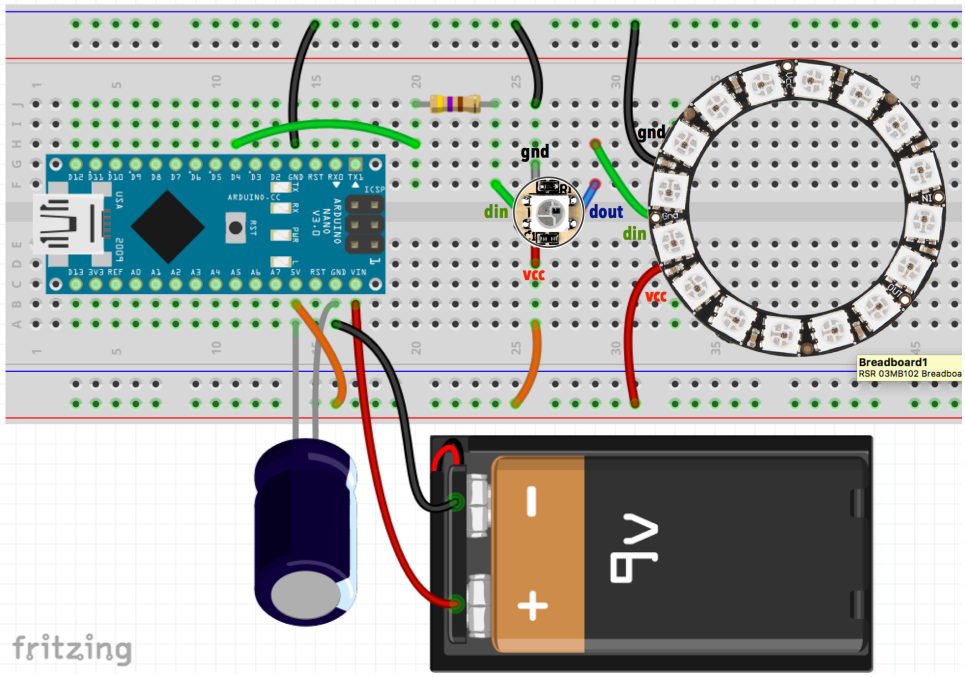
```
leds[0] = CRGB::Blue;
```

para piscar na cor azul.

Abra o exemplo **ColorPalette**. Este programa realiza uma série de animações passando por várias cores. Como temos apenas um LED, mude **NUM_LEDS** para **1**, e **LED_PIN** para **4**. Faça upload para o Arduino e observe as variações de cor e pulsos que o LED vai fazer.

É muito mais interessante usar esse programa com vários LEDs WS2812. Se você adquirir um outro LED endereçável, ou melhor ainda, uma placa com diversos LEDs WS2812 montados, você pode conectá-los em sequência ligando o pino **Dout** de uma placa à entrada **Din** da seguinte. Você então deve alterar os programas que fazem o sequenciamento de LEDs e **informar a quantidade total de LEDs** que serão controlados.

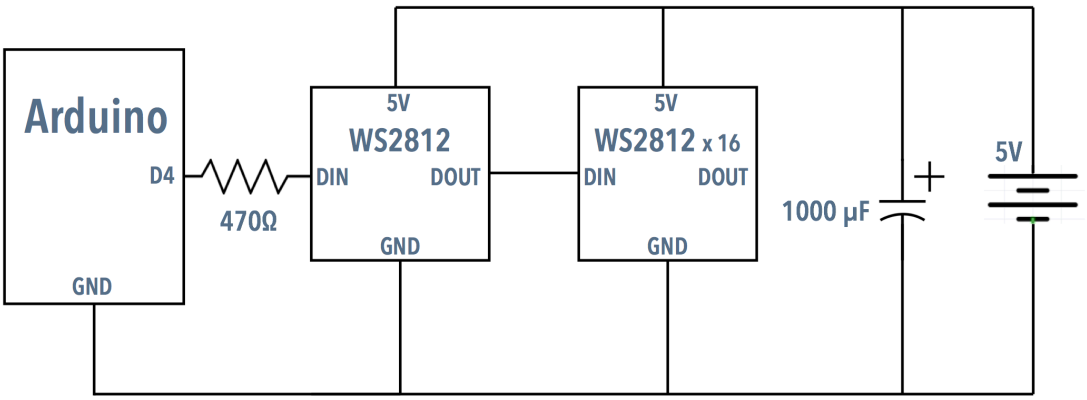
Por exemplo, no circuito abaixo adicionamos um anel de 16 LEDs em série com o WS2812 que já estava ligado ao circuito, resultando no total de **17 LEDs endereçáveis**. Altere a variável **NUM_LEDS** novamente (para **17**) e faça upload para ver os LEDs piscarem e mudarem de cor sequencialmente.



O circuito acima só deve ser usado em Arduino estabelecendo um limite de brilho para os LEDs (controlado pelo programa), já que o uso de 17 LEDs em brilho máximo exige muita corrente da saída 5V do Arduino Nano chinês (limite máximo de 500mA via USB e 800mA via fonte externa conectada a Vin). O LED com brilho máximo consome 60mA, portanto 17 LEDs com brilho máximo podem ultrapassar 1 ampere. O ColorPalette usa apenas ¼ do brilho (64) máximo. Há um fusível nos reguladores de tensão que desliga temporariamente o fornecimento de energia se ela ultrapassar esses limites.

O ideal é alimentar circuitos que tenham 7 ou mais LEDs WS2812 usando uma fonte externa. Nessa configuração, o Arduino fornece apenas o sinal de controle, através de seu pino de saída. A fonte externa pode até ser a mesma bateria, se ela fornecer até 5V.

O esquema abaixo mostra como construir o mesmo circuito acima de maneira mais segura, usando uma fonte externa. Com essa configuração, como a corrente não passará por dentro do Arduino, podemos acender os LEDs na intensidade máxima e alimentar muito mais LEDs:



No esquema acima o Arduino pode estar sendo alimentado por USB ou por outra fonte a partir da sua entrada Vin. Mas é possível também usar a mesma fonte que fornece energia para os pixels WS2812, se ela tiver capacidade de fornecimento de corrente suficiente. Por exemplo, 3 pilhas AAA de 1,5V cada poderiam alimentar o Arduino e os LEDs.

Com 4,5V não seria possível alimentar o Arduino pelo pino Vin, já que o limite mínimo é de 6V. A alimentação do Arduino teria que ser feita diretamente via pino 5V. É preciso tomar cuidado ao utilizar esse pino como entrada, já que ele está ligado diretamente ao microcontrolador (sem fusível), e poderá queimar o Arduino se a tensão passar de 5,5V.

6. Técnicas

Técnicas, informações e links para construir obras eletrônicas permanentes (sem protoboard).

6.1. Soldagem

Para construir circuitos permanentes, a forma mais comum de unir fios e componentes é através da soldagem. Os tutoriais abaixo demonstram a técnica da soldagem através de exemplos usando componentes do kit.

6.1.1. Ferramentas para soldagem

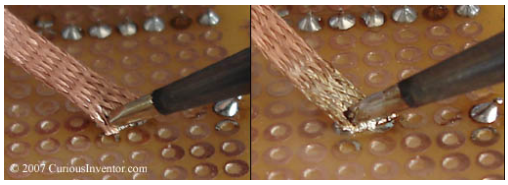
Para soldar circuitos eletrônicos o mínimo necessário é um **ferro de soldar de baixa potencia** (entre 20 e 30W) e **fio de solda** (liga de estanho que será derretida pelo ferro). Além disso, é útil ter também um pouco de **pasta de solda** (que ajuda na aderência da solda), **um suporte para o ferro de soldar com esponja** (para limpar o bico), e um **suporte com garras jacaré** para fixar componentes e placas, que precisam estar firmes na hora da soldagem. Um desses suportes é chamado de **terceira mão**, e geralmente contém também uma lupa para ampliar os objetos a serem soldados.

A soldagem consiste no aquecimento da solda junto aos contatos a serem soldados pelo tempo mínimo necessário para que a solda derreta, criando a aderência e juntando as partes. Componentes eletrônicos suportam calor por alguns poucos segundos. Se você errar a soldagem, o ideal é deixar esfriar e tentar remover a solda em uma segunda aplicação do ferro, para não danificar os componentes. Você também pode prender garras jacaré nos terminais durante a soldagem pois eles funcionam como dissipadores e reduzem a quantidade de calor que chega ao componente.

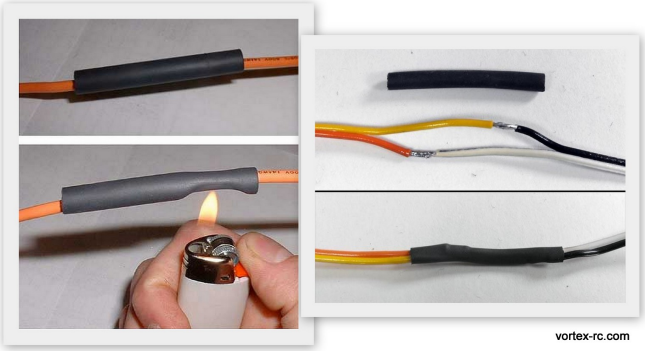
Se você vai soldar componentes mais sensíveis, o ideal é ter pelo menos uma **estação de solda com controle de temperatura** (foto abaixo, à esquerda). Um equipamento desses é um pouco mais caro, mas é necessário, principalmente se você precisar soldar **componentes SMD**, que são muito pequenos e muito mais vulneráveis ao calor.

Se você precisar soldar muitos componentes SMD em uma placa, o ideal é usar **uma estação de retrabalho** (foto acima, à direita), que contém ferro de soldar com temperatura controlada e ferramentas de solda a vapor (**sopradores térmicos**) que permitem colar componentes na placa e soldar com ar quente. Depois disso, ainda existem **fornos** que são usados para soldar vários componentes ao mesmo tempo. Uma estação de retrabalho também facilita a remoção de solda, que às vezes é muito mais difícil que a soldagem em si.

Mas existem alternativas mais baratas. Para remover solda aplicada em excesso, você pode passar uma malha de cobre chamada de **malha dessoldadora**, que absorve a solda extra. Em circuitos muito pequenos usar essa malha é inclusive uma técnica de soldagem: a solda é aplicada em excesso, para que se espalhe rapidamente, e depois ela é aquecida com a malha para remover o excesso. A solda adere apenas ao cobre, então a retirada separa as ilhas unidas pelo excesso de solda. Pode-se usar também um **sugador de solda** (abaixo, à direta), que utiliza vácuo para sugar a solda de um circuito depois que ela é aquecida.



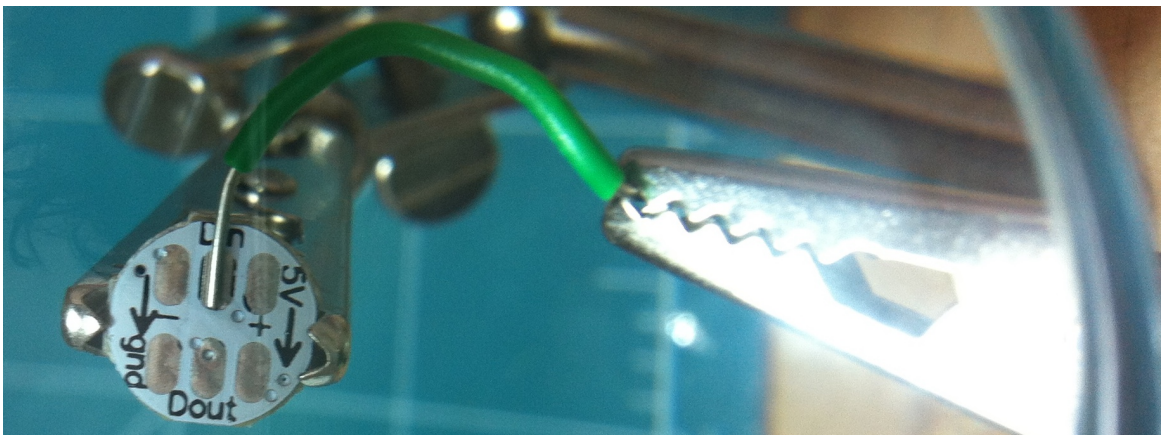
Para isolar fios, depois de soldá-los, você pode usar **fita isolante**. Mas existem alternativas mais seguras e limpas. Uma delas é usar **espaguete termo-retrátil**, que é um tubo de plástico que diminui seu diâmetro pela metade quando aquecido. Depois de fazer a ligação, passe o tubo de espaguete sobre a conexão, e aqueça rapidamente com um isqueiro. O tubo irá se retrair e isolar a conexão.



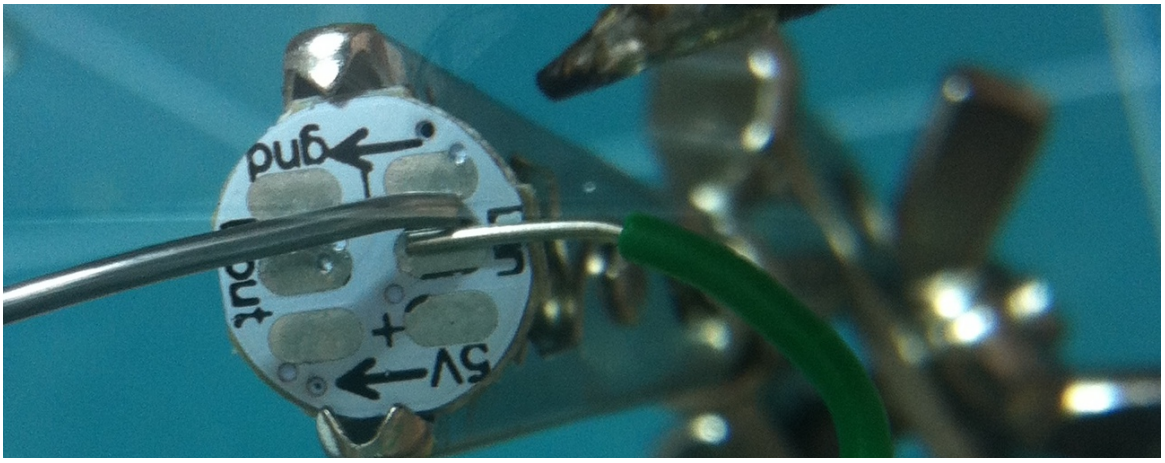
Alternativas também incluem **cola epóxi** (ex: Araldite), **cola quente** e **borracha de silicone**, que é ideal em circuitos flexíveis. **Colas "puff"** também podem ser usados em circuitos têxteis.

6.1.2. Soldagem de terminais no LED endereçável WS2812

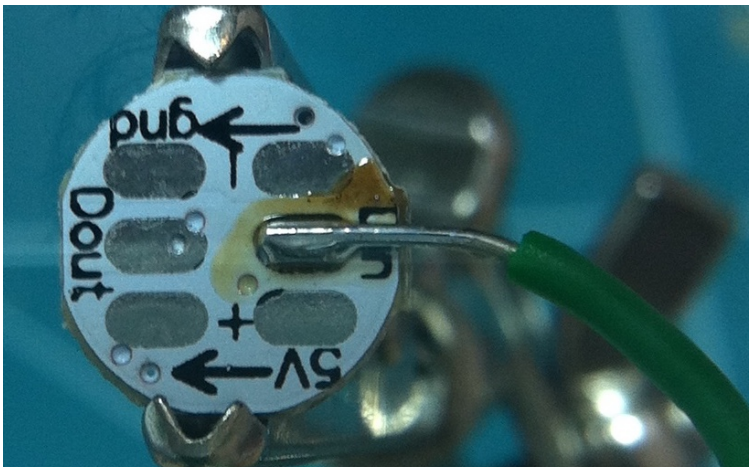
Antes de iniciar, prenda a placa dissipadora do LED em um lugar firme. Na foto abaixo usamos um suporte para placas chamado de "terceira mão", que tem uma lupa e duas garras jacaré. Mas você também pode usar outros meios, por exemplo, prender a placa numa mesa com fita crepe (deixando livres as partes que serão soldadas) ou com fita dupla-face do outro lado (protegendo a lente do LED com fita adesiva). Depois é importante fixar o fio a ser soldado sobre a superfície.



Os pontos de soldagem do LED WS2812 não precisam ser lixados. Eles já têm uma fina camada de solda que irá facilitar a aderência. Primeiro derreta um pouco de solda na ponta do ferro. Depois encoste a solda no fio de um lado e o ferro de solda por alguns segundos do outro lado. Assim que derreter, afaste o ferro e espere secar.



A foto abaixo ilustra o resultado esperado.

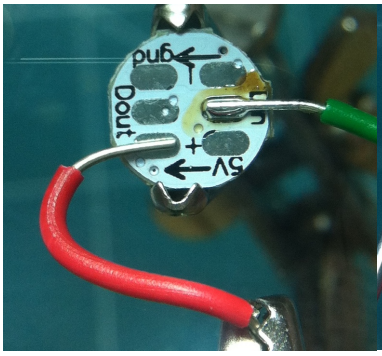


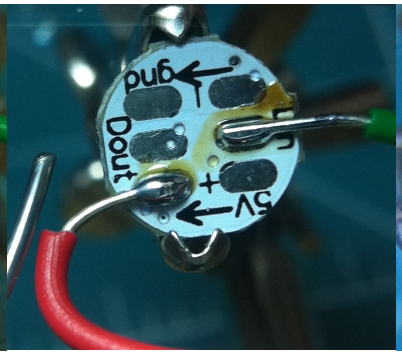
Se a solda for excessiva, limpe o ferro de solda e depois encoste-o novamente na junção, que ele absorverá o excesso.

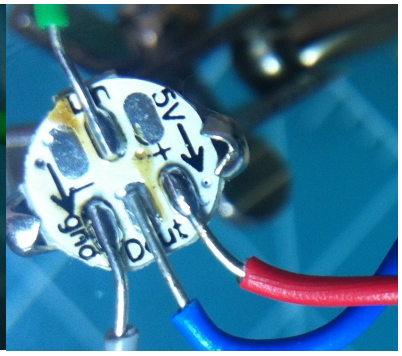
Continue soldando os outros fios. Reposicione a placa se necessário.

Para usar o LED isoladamente só é necessário fixar três terminais: **Din**, **5V** e **GND**. Se você quiser conectar o LED a outros (para fazer uma sequência) solde também um terminal em **Dout**.

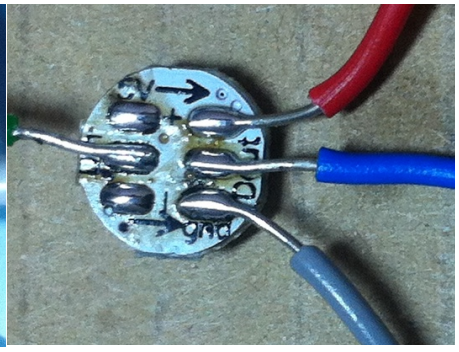
As fotos a seguir ilustram o processo.







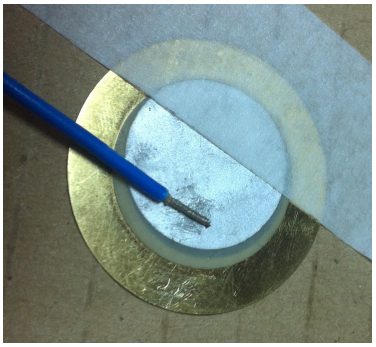
Não se preocupe se você soldar dois terminais por engano. Limpe o ferro de solda e passe-o sobre a junção novamente para que ele absorva o excesso. Se não for suficiente, você pode aquecer a junção com o ferro e uma **malha de cobre dessoldadora** para remover o excesso.

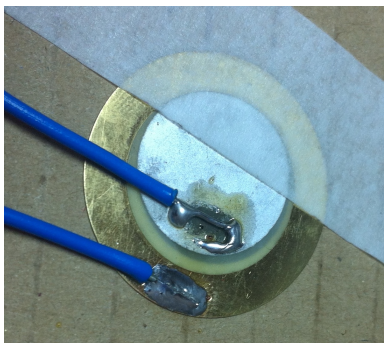


6.1.3. Soldagem de terminais na célula piezoelétrica

A célula piezoelétrica não tem pontos destinados a soldagem. Para que a solda tenha a aderência necessária para grudar na célula é necessário lixar um pouco a superfície.

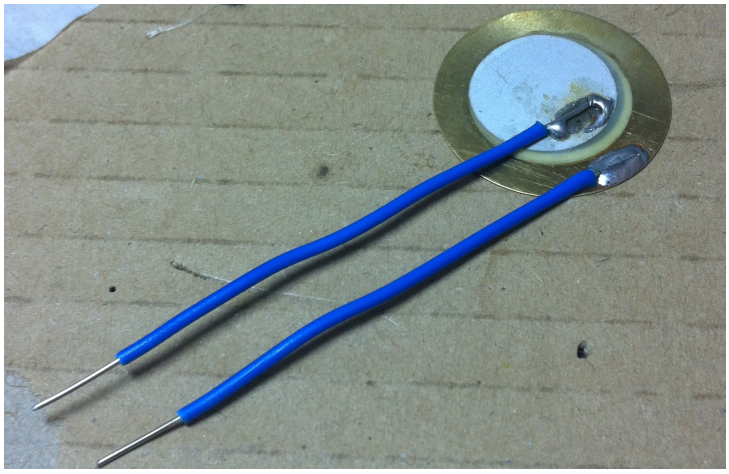
Escolha dois pontos na lateral da célula como mostrado abaixo. Prenda a célula em uma superfície firme (ex: usando fita crepe). Lixe a superfície metálica, passe um pouco de pasta de solda e em seguida derreta um pouco de solda sobre ela.





Derreta um pouco de solda também na ponta do fio. Depois, encoste o fio no ponto onde deve ser soldado, e use o ferro de solda para derreter a solda que está no fio e na célula. Assim que derreter, afaste o ferro e espere secar.

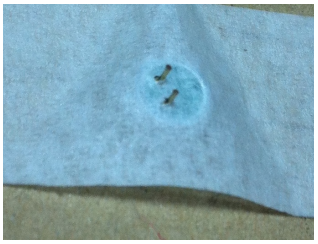
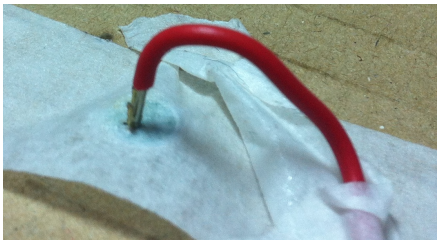
Faça o mesmo na parte cerâmica, com mais cuidado pois a fina camada que permite a aderência da solda pode desaparecer facilmente com o calor.



Você não precisa *soldar* fios na célula piezo elétrica. Se preferir, pode usar presilhas ou clipes que prendam fios firmemente em cada placa (eles devem estar bem isolados), usar fita adesiva de prata ou cobre, ou usar um soquete especialmente criado para esse tipo de célula. Você pode também improvisar o contato, durante os experimentos, com um pegador de roupa pequeno.

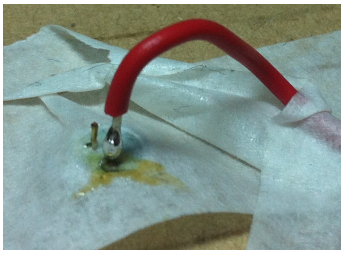
6.1.4. Soldagem de terminais mais longos no microfone de eletreto

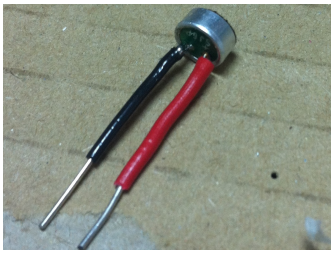
O microfone de eletreto distribuído no kit tem terminais muito curtos que podem não encaixar muito bem no protoboard. Para alongá-los, prenda o microfone firmemente em uma superfície (pode ser usando um suporte de placa (terceira mão) ou com uma fita crepe, perfurando a fita com os dois terminais, deixando-os livres para que possam ser soldados.

Passe um pouco de pasta de solda em cada terminal, e depois derreta um pouco de solda em cada um. Faça o mesmo na ponta de cada um dos fios que será soldado.

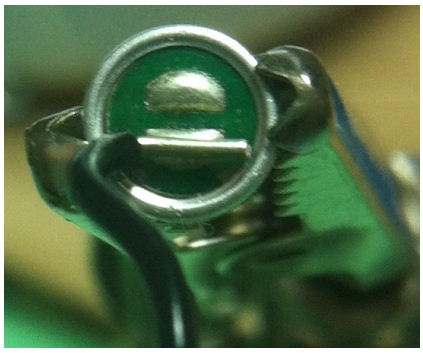
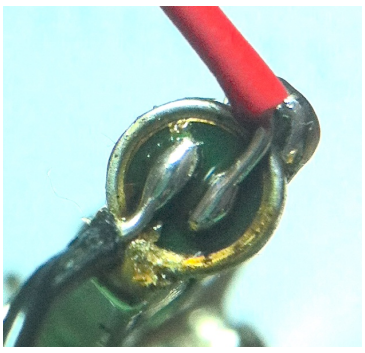
Segure o fio paralelamente ao terminal a ser soldado. Se necessário, use um alicate ou pinça. Como o fio é rígido, você pode tentar prender o fio na mesa e curvá-lo de forma que ele fique na posição esperada. Aproxime o ferro de solda e derreta a junção. Se necessário, acrescente um pouco mais de solda.





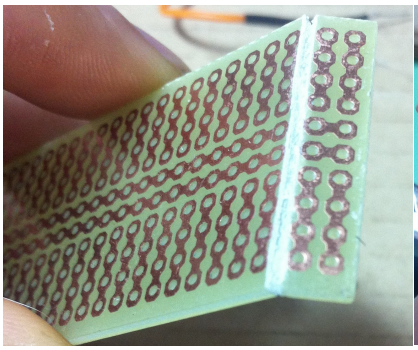
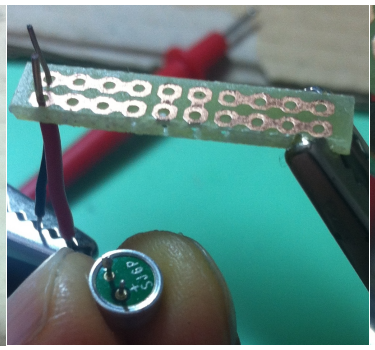
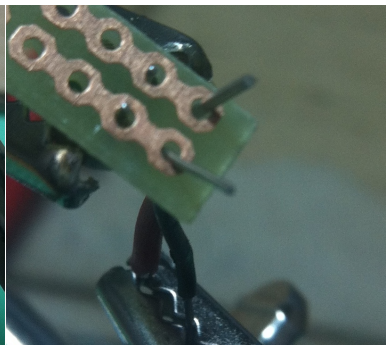
Se o microfone **não tiver terminais**, você terá que soldá-los diretamente nas ilhas localizadas na parte traseira. Nem sempre a polaridade é indicada, mas se você olhar bem verá que um dos lados está conectado à embalagem metálica do microfone. Este é o negativo.

Para soldar, usamos o suporte "terceira mão" que possui garras jacaré para sustentar os microfone, que é muito pequeno, e também os fios. Escolha cores dos fios que ajudem a identificar a polaridade (preto para negativo, e vermelho para positivo.) Prenda um fio preto com a ponta desencapada de forma a encostar no terminal negativo. Depois esquente com o ferro de solda e um pouco de solda até a solda derreter. Quando secar faça o mesmo com o terminal positivo.

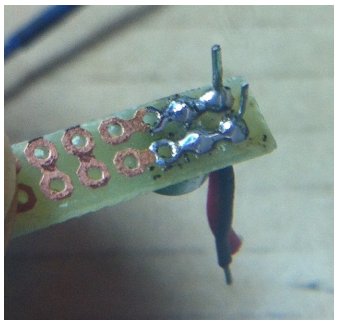
Outra maneira de acrescentar terminais longos no microfone é encaixá-lo em uma **placa de circuito impresso**, e soldar fios mais longos (ou conectores do tipo "header") para possibilitar o uso no protoboard.

Corte um pedaço sua placa de circuito impresso universal contendo duas fileiras. O ideal é usar uma mini-serra, mas você pode vincar várias vezes, dos dois lados, com um estilete, e depois usar um alicate para cuidadosamente partir em duas metades.

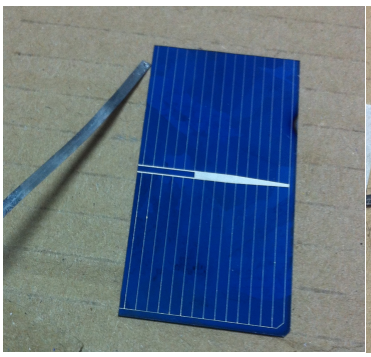
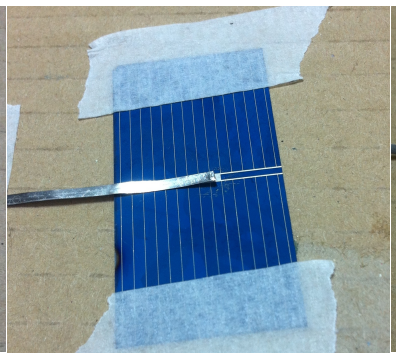
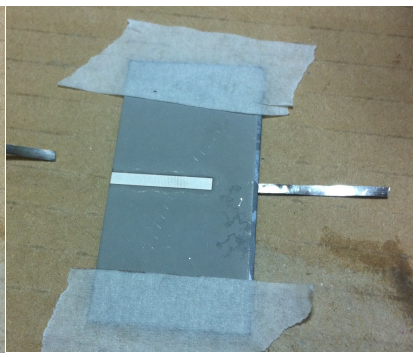
Encaixe o microfone em dois furos que estejam ligados a ilhas separadas, anotando qual deles é o positivo e qual o negativo (já que essa informação ficará oculta debaixo da placa). Encaixe fios preto e vermelho nas posições correspondentes. Prenda essa estrutura em uma superfície firme (usando fita crepe) ou use um suporte de placa ou terceira mão, se tiver.

Passe um pouco de pasta de solda nas conexões, depois derreta solda sobre as ilhas. Não se preocupe se ela for excessiva. Você pode sempre remover o excesso usando o ferro ou a malha dessoldadora.




6.1.5. Soldagem de terminais na célula fotovoltaica

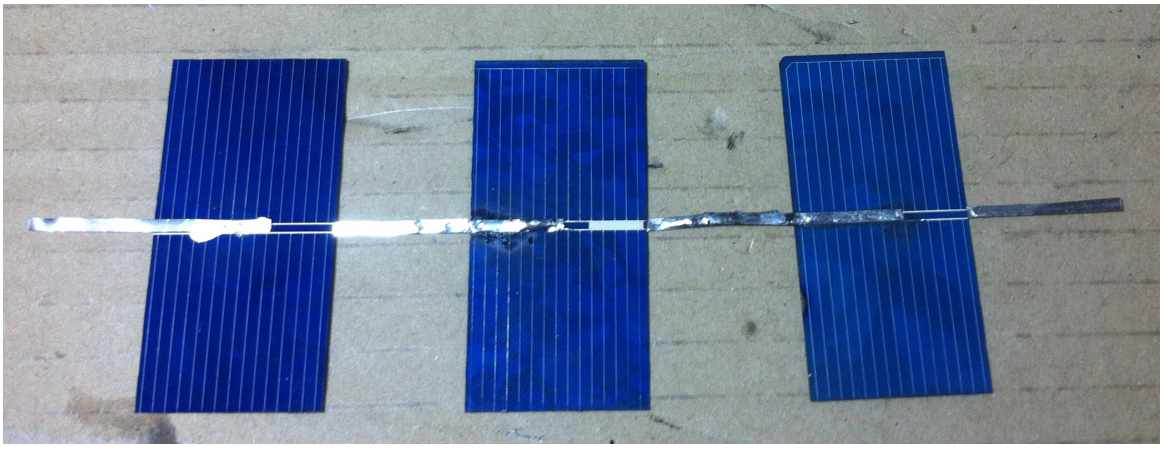
A célula fotovoltaica distribuída no kit não possui terminais soldados. Se você deseja fazer uma célula maior, com 1,5V, 3V ou mais, precisará de células adicionais. Aproveite e adquira também **fita de estanho** (veja foto) que é mais adequada para soldar essas placas que são muito frágeis.

Prenda a célula numa superfície deixando a ilha central exposta. Passe pasta de solda na fita de estanho, e posicione-a sobre a ilha na fotocélula, e encoste o ferro de solda por 10 segundos. Se a fita não colar facilmente, derreta um pouco (muito pouco) de solda sobre ela e tente novamente.

Vire a placa ao contrário e faça o mesmo do outro lado.

Para conectar várias placas em série, solde os terminais de baixo aos terminais colados na parte de cima da placa. A foto abaixo mostra uma configuração com três células, que gera até 1,5V.



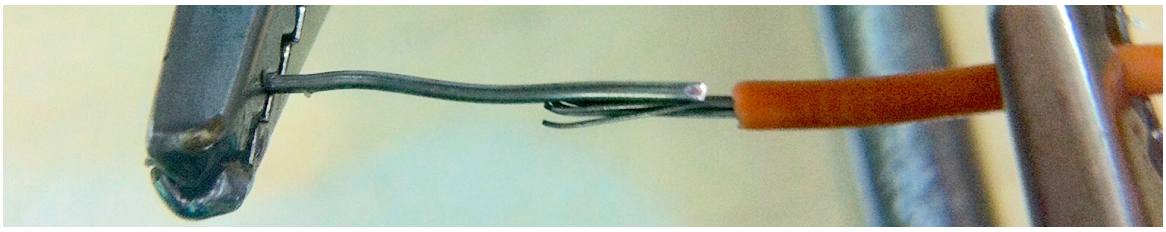
Usando 6 células, o conjunto é capaz de gerar até 3V (e ligar um motor, acionar uma cigarra ou acender um LED).

As células são muito frágeis, e a solda que une os contatos não é muito forte. Assim que você decidir como organizar as células, cole-as em uma superfície mais rígida (ex: cartão, plástico) e cubra com uma camada de resina epóxi líquida, que servirá para protegê-las e também garantir isolamento.

6.1.6. Soldagem de terminais rígidos nos terminais do motor

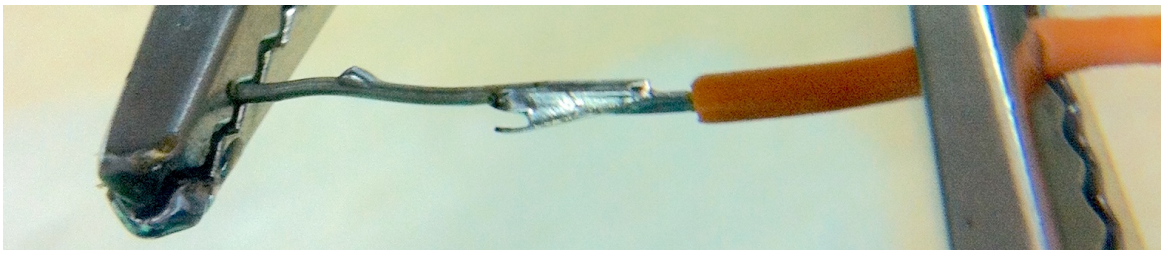
O motor distribuído no kit vem como **fios flexíveis** que não são muito fáceis de inserir no protoboard. Para torná-lo mais fácil de usar, podemos soldar **fios rígidos** nas pontas.

Primeiro é preciso prender os fios firmemente em uma superfície. Se você tiver um suporte do tipo "terceira mão" prenda o fio do motor em um jacaré, e o fio rígido em outro, e posicione-os da forma indicada abaixo.

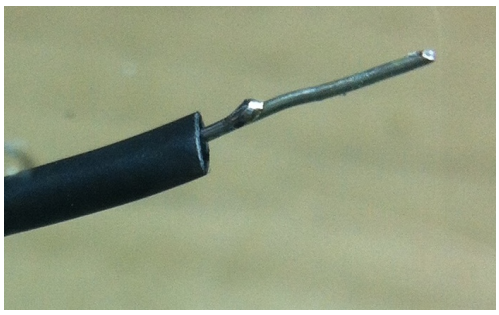


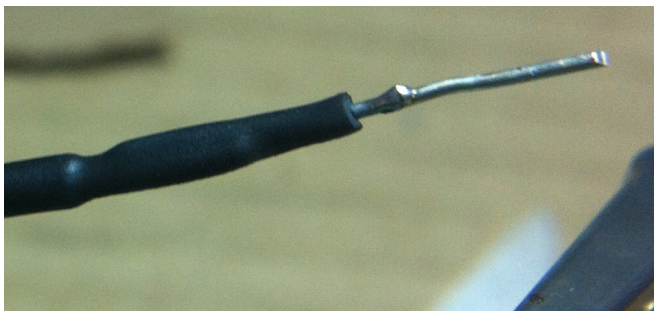
Se você não tiver uma terceira mão, use um cabo de garras jacaré que foram distribuídas no kit. Deslize a capa protetora de cada uma delas, para expor o jacaré inteiro, e prenda-os, com os fios a serem soldados, numa superfície com fita crepe, de forma que os jacarés fiquem a uma distância de alguns centímetros da mesa.

Passe um pouco de pasta de solda e depois derreta solda na junção. Se for excessiva, espere um pouco e derreta novamente com um ferro de solda limpo para tirar o excesso. A solda deverá ficar lisa e brilhante.



Quando terminar você deve isolar a conexão. A melhor forma de fazer isto é usando 2cm de espaguete termo-retratil, que é um tubo que encolhe com o calor. Use um tubo de 2mm e deslize sobre a junção, depois aqueça rapidamente com um isqueiro para que o tubo encolha e isole a ligação.





Se você não tiver espaguete termo-retrátil, use fita isolante, cola quente, cola epóxi (*Araldite*) ou em último caso, fita adesiva (mas não deixe a junção sem isolamento).

Você pode usar o motor nos experimentos com protoboard sem precisar soldar. Prenda garras jacaré firmemente em cada terminal (lembrando de baixar a capa isolante dos jacarés para evitar curtos-circuitos), e na outra ponta prenda jumpers. Agora você pode inserir os jumpers no protoboard.

6.1.7. Soldagem de um circuito na placa de fenolite universal

Você pode usar a placa de fenolite universal para fazer uma versão permanente de um circuito que você montou no protoboard. A configuração da placa é muito parecida com o protoboard e não é difícil transportar circuitos para a placa. O tutorial abaixo (Sparkfun) mostra como soldar componentes em uma placa de circuito impresso:

https://learn.sparkfun.com/tutorials/how-to-solder-through-hole-soldering

6.2. Eletrônica para vestir (wearable electronics)

A soldagem é ideal para circuitos rígidos, mas para circuitos têxteis que serão construídos em bolsas, sapatos, roupas e acessórios flexíveis, usando condutores como linha de costura condutiva, é preciso buscar técnicas alternativas. Abaixo estão alguns links para tutoriais que demonstram essas técnicas, e algumas aplicações criativas.

- **Como costurar** com linha condutiva (Sparkfun): https://learn.sparkfun.com/tutorials/lilypad-basics-e-sewing
- **Técnicas de isolamento** para circuitos de eletrônica para vestir (Sparkfun): https://learn.sparkfun.com/tutorials/insulation-techniques-for-e-textiles
- **Porta-pilhas** de feltro: http://ohvillo.blogspot.com.br/p/e-textiles-portapilas.html e http://cosiriferampolles.blogspot.com.br/2014/10/tutorial-portapilas-de-fieltro.html
- **Sensor de pressão**: http://www.instructables.com/id/Flexible-Fabric-Pressure-Sensor/
- **Jaqueta para ciclistas** com LEDs e Arduino: http://www.instructables.com/id/turn-signal-biking-jacket/ e http://www.instructables.com/id/Signaling-Cyclist-Jacket/
- **Soutien Theremin** com Arduino: https://www.youtube.com/watch?v=LmZJy8lvj5A
- **Sapatilhas do Mágico de Oz:** https://ohvillo.wordpress.com/portfolio/zapatillas-el-mago-de-oz/
- **Alto-falantes** de tecido: http://www.kobakant.at/DIY/?p=2936
- **Seção de wearables do site oficial do Arduino:** https://blog.arduino.cc/category/wearable-computing/
- **Seção de wearables da AdaFruit:** https://learn.adafruit.com/category/wearables
- **Como preparar um LilyPad:** https://ohvillo.wordpress.com/portfolio/primeros-pasos-con-el-lilypad-que-es-y-como-prepararlo/

6.3. Eletrônica com outros materiais

Os links abaixo contém informações, produtos e tutoriais úteis para a construção de projetos usando outros materiais além de circuitos tradicionais rígidos e tecido.

6.3.1. Material para circuitos de papel

Canetas de tinta condutiva:

- **Circuit Scribe**. Caneta de tinta condutiva. (https://www.circuitscribe.com/).
- **AgIC Circuit Maker**. Caneta de tinta condutiva (https://agic.cc/en/). https://www.youtube.com/watch?v=4TnkmatWAiM https://www.youtube.com/watch?v=FAC3kqzWm4g

Adesivos condutivos:

- **Cola epoxi condutiva**: http://www.mgchemicals.com/products/adhesives/electrically-conductive-adhesives/
- **Fita de cobre condutiva**: http://produto.mercadolivre.com.br/MLB-788470458-fita-adesiva-de-cobre-condutiva-5mm-_JM
- **Fita de tecido de prata condutiva**: http://produto.mercadolivre.com.br/MLB-800437426-fita-adesiva-de-tecido-com-prata-condutiva-10m-x-10mm-_JM
- **Stickers condutivos**: https://www.crowdsupply.com/chibitronics/circuit-stickers

6.3.2. Massa condutiva

Receitas, materiais e circuitos:

- **Receita**: http://blog.novaeletronica.com.br/como-fazer-massa-condutiva/
- **Cremor de Tártaro** (ingrediente usado em massa condutiva) http://www.americanas.com.br/produto/9781558/cremor-de-tartaro-com-50g-mix
- **Circuitos** com massa condutiva e massa isolante http://courseweb.stthomas.edu/apthomas/SquishyCircuits/buildingCircuits.htm

7. Componentes: referência rápida

Para maiores informações, procure o *datasheet* ou manual na Internet (alguns produtos abaixo contém o link), usando o código do componente na pesquisa.

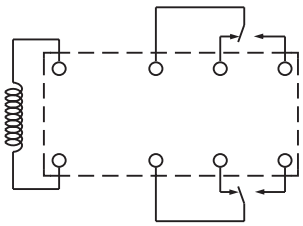
7.1. Componentes eletromagnéticos

7.1.1. Relé

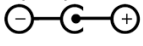
Sanyou DSY2Y-S-205L

- *Datasheet:* http://www.sanyourelay.ca/public/products/pdf/DSY2Y.pdf
- Voltagem nominal: 5V
- Voltagem mínima de acionamento: 3,75V
- Voltagem de liberação: 4,75V
- Voltagem máxima: 10V (não aplique mais de 10V)
- Corrente de operação: 40mA
- Resistência: 125R
- Carga nominal: 1A, 120V.
- Corrente máxima de chaveamento: 2A.

Esquema de pinagem e conexões internas:



7.1.2. Fonte de alimentação chaveada

- Tensão fornecida: **9V** ou **12V** (confira a indicação na parte de baixo da fonte)
- Corrente máxima fornecida: 1 ampere (**1 A**)
- Saída: plugue P4 (positivo no centro: )

7.1.3. Motor RF-300CA-09550

- Faixa de operação: 1 a 6V (não aplique mais de 6V)
- Valor nominal: 3V; velocidade 2700 rpm (sem carga), corrente 12mA
- Máxima eficiência: 2100 rpm com 2,8 g*cm de torque, 60mW, 42mA
- Motor parado: 13 g*cm de torque, 150mA
- *Datasheet:* http://www.jameco.com/Jameco/Products/ProdDS/238458.pdf

Embora não tão eficiente, este motor também pode ser usado como **gerador**. Se você colocar um LED entre os dois terminais, e der um giro rápido no eixo do motor com os dedos, deverá ver o LED acender brevemente. Um gerador mais eficiente (para acender LEDs em uma pipa, por exemplo) pode ser construído usando um motor com torque maior (e menos RPM), e uma tensão nominal maior (ex: 12 ou 24V). Para acender Leds e operar circuitos em energia contínua também é necessário construir um **circuito retificador** na saída do gerador, com diodos e capacitores, pois a energia gerada é alternada.

7.1.4. Buzzer (cigarra) ativo 5V

- Tensão nominal: 5V
- Faixa de operação: 3 a 8 V (não aplique mais de 8V)
- Saída de som: 80 dB
- Corrente máxima: 30mA
- Frequência média: 2,7kHz (+/-500Hz)

7.1.5. Mini microfone de eletreto

- Sensibilidade: -38db +/- 2dB
- Impedância: 2,2k ohms
- Frequência: 20Hz a 16kHz
- Operação: 1 a 10V
- Valor típico de operação: 3V
- Corrente máxima: 500mA

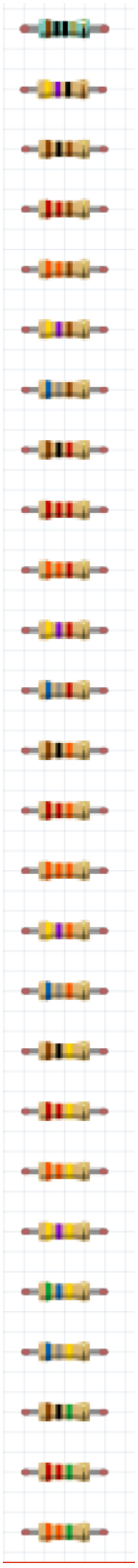
7.1.6. Mini alto-falante

- Impedância: 8 ohms
- Potência nominal: 0,5W

Para usar, é necessário soldar fios ou jumpers nos terminais. Não ligue diretamente em corrente contínua, pois a bobina poderá esquentar e queimar. Não há polaridade.

7.2. Resistores e capacitores

Todos os resistores são de **¼ W**, exceto o de 10 ohms que suporta até **1W**. Todos os capacitores suportam pelo menos **16V**.

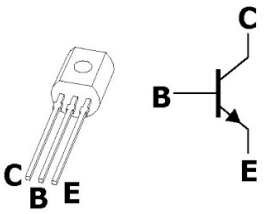
Resistores		Capacitores	
	**10 Ω** marrom-preto-preto-**ouro** (100 * **0,1**) 47 Ω amarelo-violeta-preto-ouro (470 * 0,1) **100 Ω** marrom-preto-marrom (10 * 10) 220 Ω vermelho-vermelho-marrom (22 * 10) 330 Ω laranja-laranja-marrom (33 * 10) 470 Ω amarelo-violeta-marrom (47 * 10) 680 Ω azul-cinza-marrom (68 * 10) **1k Ω** (1000 Ω) marrom-preto-vermelho (10 * 100) 2,2k Ω vermelho-vermelho-vermelho (22 * 100) 3,3k Ω laranja-laranja-vermelho (33 * 100) 4,7k Ω amarelo-violeta-vermelho (47 * 100) 6,8k Ω azul-cinza-vermelho (68 * 100) **10k Ω** (10 000 Ω) marrom-preto-laranja (10 * 1000) 22k Ω vermelho-vermelho-laranja (22 * 1000) 33k Ω laranja-laranja-laranja (33 * 1000) 47k Ω amarelo-violeta-laranja (47 * 1000) 68k Ω azul-cinza-laranja (68 * 1000) **100k Ω** (100 000 Ω) marrom-preto-amarelo (10 * 10000) 220k Ω vermelho-vermelho-amarelo (22 * 10000) 330k Ω laranja-laranja-amarelo (33 * 10000) 470k Ω amarelo-violeta-amarelo (47 * 10000) 560k Ω verde-azul-amarelo (56 * 10000) 680k Ω azul-cinza-amarelo (68 * 10000) **1M Ω** (1 000 000 Ω) marrom-preto-verde (10 * 100000) 2,2M Ω vermelho-vermelho-verde (22 * 100000) 3,3M Ω laranja-laranja-verde (33 * 100000)		**100pF** **1nF** (1000pF, 1kpF) 3,3nF (3,3kpF) 4,7nF (4,7kpF) **10nF** (10kpF, 0,01 µF) 22nF (22kpF, 0,022 µF) 47nF (47kpF, 0,047 µF) **100nF** (0,1 µF) 330nF (0,33 µF) 470nF (0,47 µF) **1 µF** 2,2 µF 3,3 µF 4,7 µF **10 µF** 22 µF 33 µF 47 µF **100 µF** 220 µF 470 µF **1000 µF** 2200 µF

7.3. Semicondutores

7.3.1. Transistores bipolares de junção

NPN de propósito geral: BC548/548/549 ou 2N3904/2N2222

- Corrente de coletor típica: 100mA
- Ganho típico: 100-300 (quantidade de vezes que a corrente na base é amplificada)
- Tensão máxima entre base e emissor: 5V
- Tensão máxima entre coletor e emissor: 30V
- *Datasheet*, série BC: http://cygnus.et.put.poznan.pl/~kklima/aue/BC549_550.pdf
- *Datasheet*, série 2N: https://www.sparkfun.com/datasheets/Components/2N3904.pdf

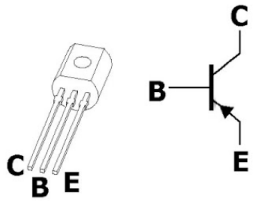


PNP de propósito geral: BC557/558/559 ou 2N3906

Mesmas especificações gerais dos correspondentes NPN.

Datasheets:

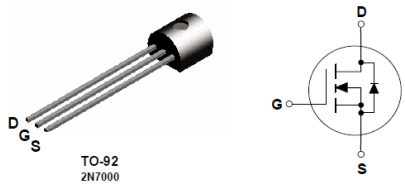
- BC: https://cdn.instructables.com/ORIG/FYG/F1O8/II0K92HF/FYGF1O8II0K92HF.pdf
- 2N: https://www.sparkfun.com/datasheets/Components/2N3906.pdf



7.3.2. Transistores de efeito de campo (MOSFETs)

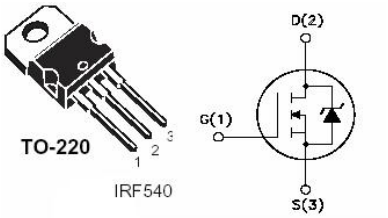
MOSFET de propósito geral: 2N7000

Datasheet: https://www.onsemi.com/pub/Collateral/2N7000-D.PDF



MOSFET de potência: IRF540

Datasheet: http://www.vishay.com/docs/91021/91021.pdf



7.3.3. Diodos

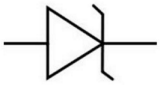
Diodo de propósito geral: 1N4148

Estes diodos são tipicamente usados em circuitos retificadores (para converter sinais de corrente/tensão alternada), para proteção contra pulsos de corrente reversa (em circuitos com motores, relés e transformadores) e em qualquer situação onde for necessário que a corrente flua apenas em um sentido (nos astáveis 555, para controlar a largura dos pulsos).

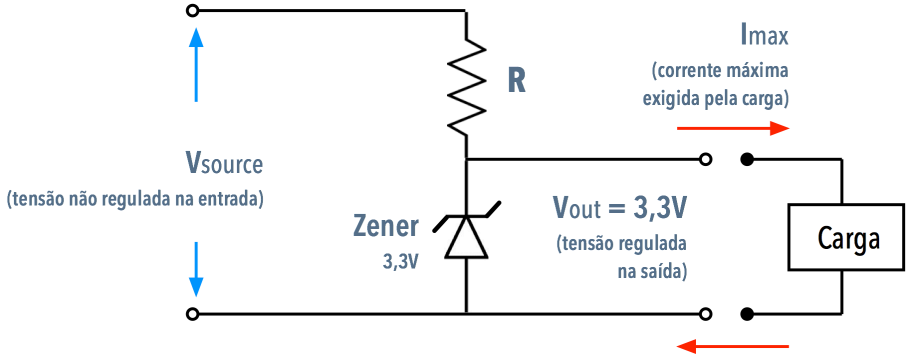
Datasheet: http://www.vishay.com/docs/81857/1n4148.pdf

Diodo regulador de tensão: 1N4728A – Diodo Zener

Datasheeet: http://www.vishay.com/docs/85816/1n4728a.pdf



Este diodo mantém uma tensão constante sobre ele mesmo quando a tensão de entrada varia. É usado em reguladores de tensão (para garantir que a tensão em uma carga não ultrapasse um determinado valor). A tensão mínima de entrada deve ser de 1 a 2V maior que a tensão de saída. É necessário calcular o valores da **resistência** e **potência** do resistor que fará um divisor de tensão com o diodo, e também a potência suportada pelo diodo zener.



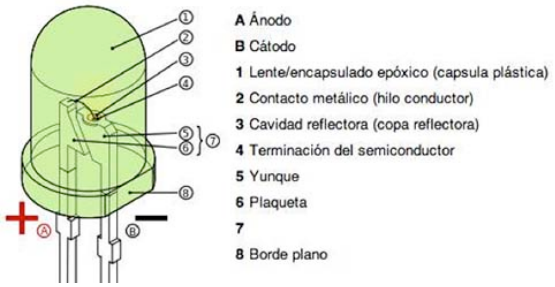
Calculadora online: http://ncalculators.com/electronics/zener-diode-calculator.htm

É muito importante que potências do diodo e resistor usados no circuito sejam iguais ou superiores aos valores calculados para evitar que esquentem demais.

O diodo zener1N4728 incluído no kit é calibrado para regular 3,3V e tem potencia máxima de **1W**. A maior parte dos resistores incluídos no kit são de **¼ W** (0,25W).

7.3.4. LEDs

LEDs são diodos que emitem luz. Há vários diferentes tipos de LED no kit.

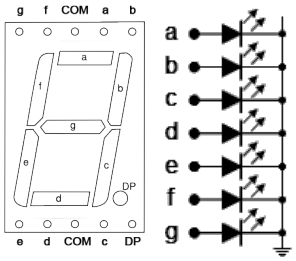


(Wikimedia)

Tutorial da Sparkfun: LEDs: https://learn.sparkfun.com/tutorials/light-emitting-diodes-leds.

Display de LED com 7 segmentos HS5161AS

- Catodo comum (o catodo de cada LED é interligado e acessível nos terminais centrais)
- Tensão direta típica: 1,8V (esta é a queda de tensão em cada LED)
- Tensão reversa máxima: 5V
- Corrente máxima por segmento: 20mA
- Corrente de operação recomendada: 12mA (calcule o resistor usando esta corrente)
- *Datasheet*: http://www.dipmicro.com/?datasheet=TOS-5161AS.pdf



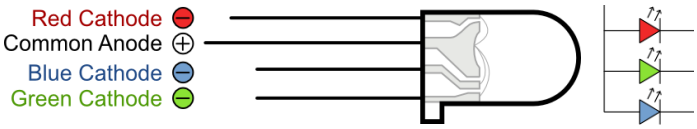
LED 5mm de alto-brilho

- Tensão direta:
  - Branco, Azul, Verde, Rosa: 3,2V;
  - Vermelho, Amarelo: 2V;
  - Violeta: 3,4V
- Corrente máxima de operação: 20mA
- Tensão/corrente reversa: 5V/10uA
- Máxima corrente pulsada (0,1ms): 100mA

LED 5mm vermelho difuso

- Tensão direta: 2,2V
- Corrente máxima de operação: 20mA
- Tensão/corrente reversa: 5V/10uA
- Máxima corrente pulsada (0,1ms): 100mA

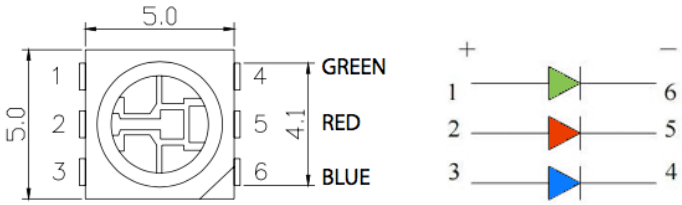
LED 5mm RGB de anodo comum



(fonte: arduino-info.wikispaces.com)

- Tensão/corrente reversa: 5V/10uA
- Tensão direta (20mA): R 2V, G 3,2V, B 3,2V
- Corrente direta de operação: 20mA/20mA/20mA (60mA)

LED SMD RGB 5050



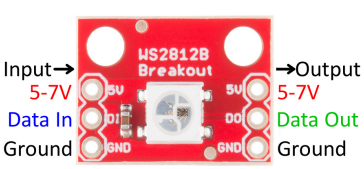
- Anodo (+): terminais 1, 2, 3; Catodo (-): terminais 4, 5, 6
- Tensão/corrente reversa: 5V/10uA
- Tensão direta (20mA): R 2V, G 3,2V, B 3,2V
- Corrente direta de operação: 20mA/20mA/20mA (60mA)

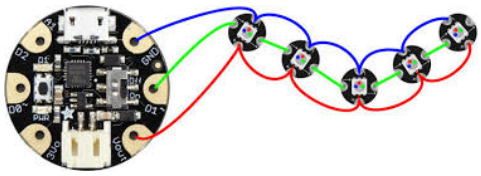
LED 5050 WS2812 (NeoPixel)

Este é um LED RGB 5050 contendo um chip endereçável. Ele não acende com alimentação direta, mas requer um pulso de largura definida que controla qual a cor a ser exibida. Tipicamente é usado em circuitos Arduino, conectando o pino **Din** a um pino de saída digital do Arduino. O pino **Dout** é usado quando vários LEDs WS2812 são conectados em série.

- *Datasheet*: https://cdn-shop.adafruit.com/datasheets/WS2812.pdf
- *Tutoriais*: https://learn.sparkfun.com/tutorials/ws2812-breakout-hookup-guide/all.pdf
  https://www.tweaking4all.com/hardware/arduino/arduino-ws2812-led/
  https://learn.adafruit.com/adafruit-neopixel-uberguide/power

Conecte 5V no positivo, GND no negativo, DI em qualquer pino Arduino (entrada). Se for usar em série com outro, conecte o DO/Dout ao DI/Din do LED seguinte.



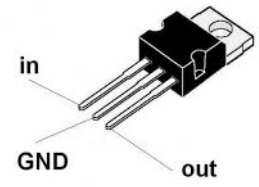


- Tensão nominal: 5V; Tensão máxima absoluta: 5,5V (não ultrapasse esta tensão)
- Usar resistor de 470 ohms entre o Arduíno e primeiro LED. Usar capacitor de 1000uF entre terminais de alimentação. Cada LED consome 60mA no brilho máximo. Não alimente (Pino 5V) mais que 7 LEDs via Arduino (use uma fonte externa).

7.3.5. Circuitos integrados e outros semicondutores

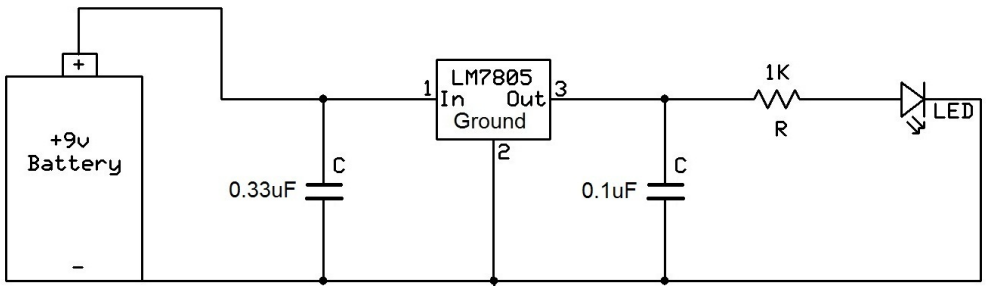
L7805CV – Regulador de tensão

Este componente é usado para limitar a tensão em circuitos que requerem no máximo 5V. Tem funcionamento similar ao diodo zener. Ele permite que circuitos desse tipo sejam alimentados por baterias ou fontes com mais de 5V (o Arduino possui reguladores similares, para 3,3 e 5V). Este regulador garante a tensão na saída de 5V desde que a entrada tenha pelo menos 7V.



- Entrada: 7 a 15V
- Saída 5V
- Corrente: 500mA
- *Datasheet*: http://www.mouser.com/ds/2/389/l78-974043.pdf

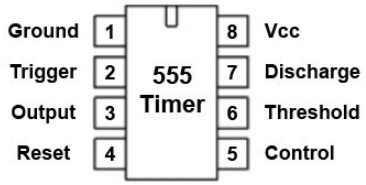
Circuito com exemplo de uso (reduzindo de 9V para 5V):



LM 555 CN – Temporizador multiuso de propósito geral

O 555 é um circuito integrado simples, barato e versátil que serve para construir diversos circuitos fundamentais da eletrônica digital, como alternadores de estado (flip-flop bi-estável), temporizadores (cronômetro monoestável) e geradores de pulsos (clock) em onda quadrada (multivibrador astável).

Datasheet: https://www.diodes.com/assets/Datasheets/NE555_SA555_NA555.pdf

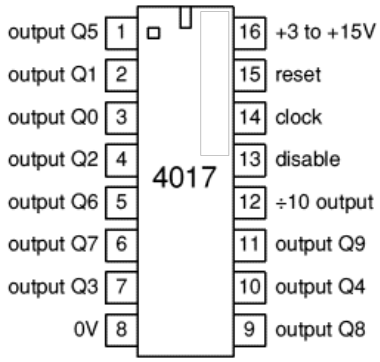


Veja exemplos de uso nos experimentos da apostila.

CD 4017 BD – Contador de década

Produz valor lógico alto nos pinos Q0 a Q9 sequencialmente a cada pulso recebido no pino 14. Reinicia o contador com nível alto no pino 15. Desliga o contador com nível alto no pino 13.

Datasheet: http://www.ti.com/lit/ds/symlink/cd4017b.pdf

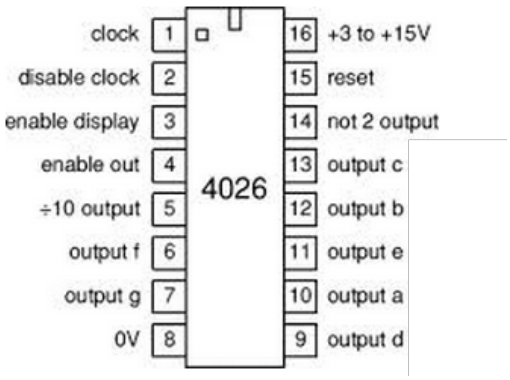


Veja exemplo de uso nos experimentos da apostila.

CD 4026 BE – Contador de década com decodificador para display de 7 segmentos.

Produz uma sequência de valores entre 0 e 9 codificados em pinos de saída especificamente configurados para acender displays de 7 segmentos.

Datasheet: http://www.ti.com/lit/ds/symlink/cd4026b.pdf

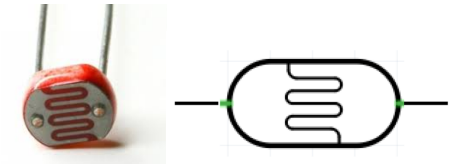


Veja exemplo de uso nos experimentos da apostila.

7.4. Sensores

7.4.1. LDRs genéricos de 5 e 7mm (valores típicos)

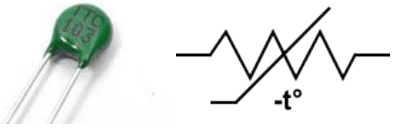
Sensores resistivos difusos cuja resistência diminui com o aumento da intensidade luminosa.



- Resistência no escuro: tipicamente 500k a 1M ohms
- Resistência no claro: tipicamente 50 a 100 ohms

7.4.2. Termistor genérico NTC 10k típico

Sensor resistivo cuja resistência diminui com o aumento da temperatura, e aumenta com a redução. Sua resistência é de 10k ohms em temperatura ambiente de 25 graus Celsius.



Temperatura: Resistência

- 0 °C: 36k Ω
- 15 °C: 16k Ω
- 20 °C: 13k Ω
- 25 °C: 10k Ω
- 30 °C: 8k Ω
- 35 °C: 6k Ω
- 50 °C: 3k Ω
- 100 °C: 550 Ω

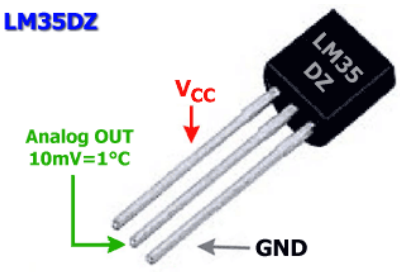
7.4.3. LM35DZ – Termômetro de precisão

O LM35 é um circuito integrado que funciona como sensor de temperatura de precisão, produzindo uma variação bem definida de tensão entre seus terminais de acordo com a temperatura.

Datasheet: http://www.ti.com/lit/ds/symlink/lm35.pdf

Cada 10mV (entre a perna OUT e GND) corresponde a 1 grau Celsius. Vcc pode ser 5 a 15V. Este modelo não mede temperaturas negativas. A margem de erro é de 0,5 graus Celsius na faixa entre 2 e 100 graus Celsius.

Pode-se construir um termômetro de LEDs usando um circuito integrado comparador (como o LM3914, que não foi incluído no kit), ou, de forma muito simples ligando o pino central a uma entrada analógica do Arduino, no qual pode-se obter a tensão e calcular a temperatura.

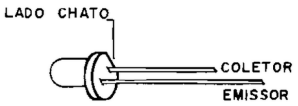


7.4.4. Fototransistor TIL 78

O fototransistor incluído no kit tem uma embalagem idêntica a um LED (mas foi distribuído no kit entre os transistores.) Se você tentar acender um LED transparente com pernas longas e ele não funcionar, não jogue fora. Pode ser um fototransistor!

TIL78

Foto-transistor NPN de Silício - Texas Inst.
(Compatível com o TIL32 e TIL902 - emissores)



Características:
$V_{CE(max)}$.....50 V
P_{tot}.....50 mW
Corrente no escuro (max)...100 nA
Corrente no claro (tip).....1 µA
$V_{CE(SAT)}$.....0,4 V (tip)

(http://eletronicassim.blogspot.com.br/)

7.4.5. Chave/sensor magnético reed

O *reed* é uma chave normalmente aberta que fecha com a aproximação de um imã.

- Tensão máxima de operação: 200V
- Corrente máxima de operação: 0,5 A

Evite dobrar os terminais, pois a ampola é muito frágil. Se necessário, dobre os terminais cuidadosamente com alicates de ponta fina, e não dobre muito perto do vidro, ou solde fios mais longos diretamente nos terminais. Dependendo de onde for usado, o reed pode ser protegido soldando os terminais em uma placa, ou envolvendo a ampola com resina epóxi.

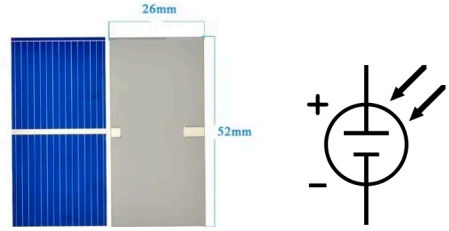
7.4.6. Célula piezoelétrica

O sensor piezoelétrico é sensível à **deformação**, gerando pulsos de corrente alternada quando há compressão ou expansão do material cristalino. Isto pode ser provocado por pressão, aceleração, impactos, tensão, força. O efeito piezoelétrico foi descoberto por Pierre Curie em 1880, mas aplicações práticas só começaram a surgir na segunda metade do século 20. Sensores piezoelétricos são hoje usados em diversas aplicações, inclusive como forma alternativa de geração de energia.

O sensor incluído no kit gera pulsos de alguns volts para cada deformação (ex: impacto ou pressão no centro do sensor). Um impacto é suficiente para piscar um LED mas a potência não é suficiente para queimá-lo.

7.4.7. Célula fotovoltaica de 0,5V (silício policristalino)

É uma célula base para painéis solares. Esta célula é muito frágil. É como vidro superfino. Manuseie com cuidado.



- Tensão média em luz solar direta: 0,5V
- Potência média: 0,20W
- Corrente máxima fornecida: 400mA
- Eficiência: 15-20%

Esta célula gera apenas 0,5V no máximo. É necessário soldar 10 células **em série** para gerar 5V e alimentar um Arduino. Para acender um LED são necessárias 4 a 6 células em luz solar direta. Duas ou três células já permitem acionar a cigarra de 5V ou o motor de 3V incluídos no kit. A solda deve ser feita com terminais de estanho. Há um mini-tutorial nesta apostila (e vários tutoriais na Internet explicando como fazer). Depois de soldar, proteja com uma camada de resina epóxi líquido.

Você também pode adquirir painéis prontos com 5V ou mais, além de células flexíveis. Para aplicações práticas, normalmente conecta-se o painel a uma placa mini-carregador de baterias, conectado a baterias recarregáveis, para que o circuito alimentado funcione quando não há luz. Esse tipo de configuração pode ser usada para alimentar um Arduino.

7.5. Imãs

7.5.1. Imã de neodímio N24 10x4mm

Este é um imã de terras raras. Os imãs de neodímio Dentre os imãs de neodímio, N24 (24 mega-gauss) é um dos mais fracos. Existem imãs N35, N42 e N52 que são bem mais fortes. Este imã é forte o suficiente para acionar sensores com vários centímetros de distância. É ideal para experimentar com solenoides e alto-falantes em *wearables* (circuitos vestíveis).

Não aproxime este imã de dispositivos magnéticos, celulares, discos rígidos, etc. Evite impactos, pois a proteção metálica externa pode rachar, e o material interno é frágil.

7.5.2. Imã de ferrite 10x4mm

Imã comum "de geladeira". Pode ser usado para acionar chaves magnéticas, mas é bem mais fraco que o de neodímio e precisa estar bem próximo para funcionar. Evite impactos, pois o material do imã quebra com facilidade.

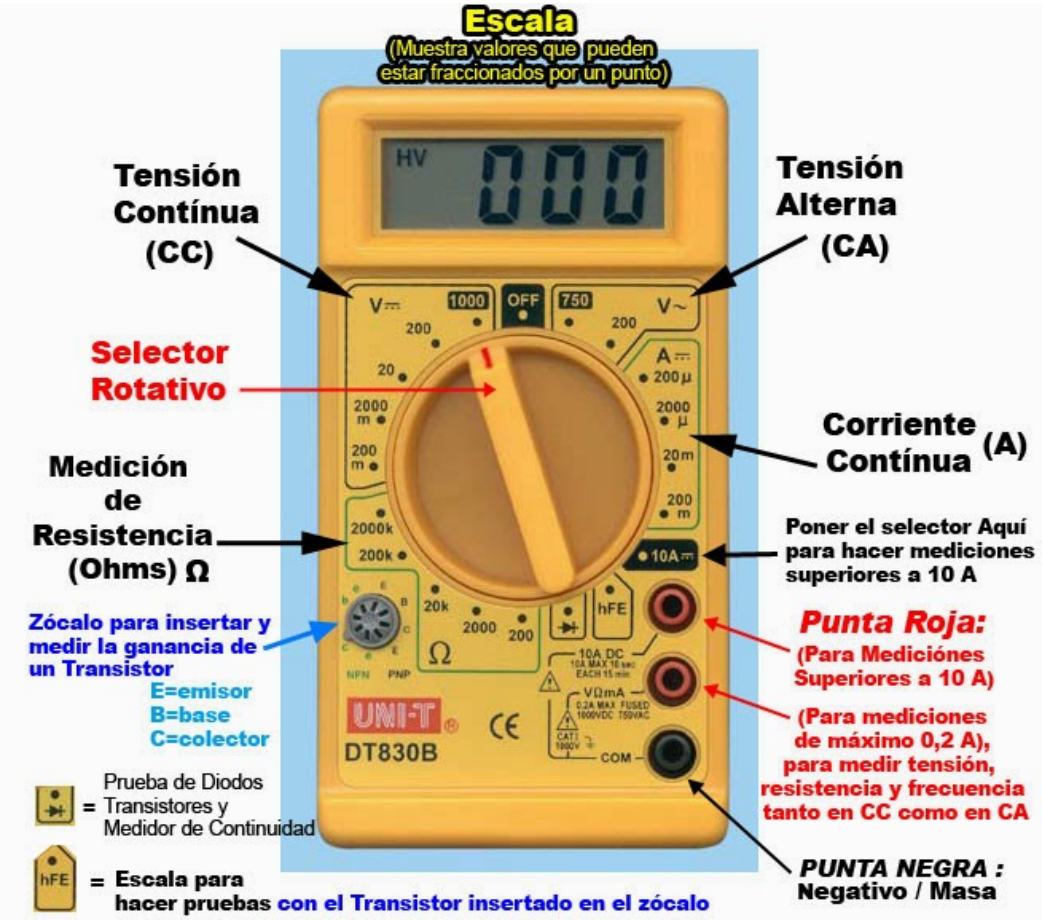
7.5.3. Eletroimã

Um parafuso de 7mm foi incluído no kit para experimentos com eletromagnetismo. Para fazer um eletroímã enrole umas 200 espiras de fio (de preferencia fio esmaltado fino 28 AWG) no parafuso, e aplique tensão (pode ser 9V) para magnetizar temporariamente o parafuso, fazendo-o funcionar como um imã.

7.6. Ferramentas e acessórios

7.6.1. Multímetro DT830B

Este modelo de multímetro é distribuído por vários fabricantes diferentes (com qualidade que também varia bastante). Ele é muito barato e simples, mas é suficiente para usar no curso para medir tensão e resistência. Ele não tem boa precisão para valores muito baixos de tensão e resistência, e alguns modelos infelizmente não têm fusível no medidor de corrente (apesar do manual e embalagem dizer que têm).



Fonte: http://www.explicofacil.com/2014/05/como-utilizar-un-multimetro-o-tester-es.html

Alguns cuidados básicos são:

- **Verifique o que será medido antes**, e posicione o seletor na área correspondente. Para medir tensões contínuas, inicie posicionando o seletor em posição correspondente a **tensão maior que a alimentação do circuito**, e gire o seletor baixando o valor para melhor a precisão, se necessário.
- Você pode usar o multímetro para medir **tensão alternada da rede elétrica**, mas é preciso posicionar o seletor na seção correspondente a tensão alternada (750 ou 200V) e **não tocar nas pontas metálicas das pontas de prova**, já que há risco de choque.
- A medição de **resistência** deve ser feita **apenas com circuitos desligados**. Idealmente, apenas para componentes individuais. Se o circuito contiver capacitores, descarregue-os antes (faça seus terminais tocarem).
- A medição de tensão é feita **sem abrir o circuito**. Apenas uma pequena parte de corrente entra no multímetro. Coloque o multímetro **em paralelo** com o componente a ser medido.
- **A medição de corrente é a que oferece o maior risco**, pois requer que toda a corrente do trecho a ser medido passe por dentro do multímetro. Até um pulso breve de alta corrente pode ser excessivo. Coloque o multímetro **em série** com o trecho a ser medido. No seletor de baixas correntes, **o máximo que o multímetro suporta é 200mA.** Mais do que isto pode

queimar o fusível do medidor de corrente. Para evitar a queima, primeiro calcule o máximo de corrente que pode passar no trecho a ser medido e **sempre comece medindo na posição 10 A**. Isto requer o encaixe da ponta de prova vermelha no primeiro soquete, e mover o seletor para a posição 10A. Se o valor estiver muito baixo, e menos de 0,2 A (preferencialmente menos de 0,1 para maior segurança), mova a ponta de prova para o segundo soquete e gire o seletor para a **posição 200m**, para ter melhor resolução. Gire o seletor para valores menores se necessário.

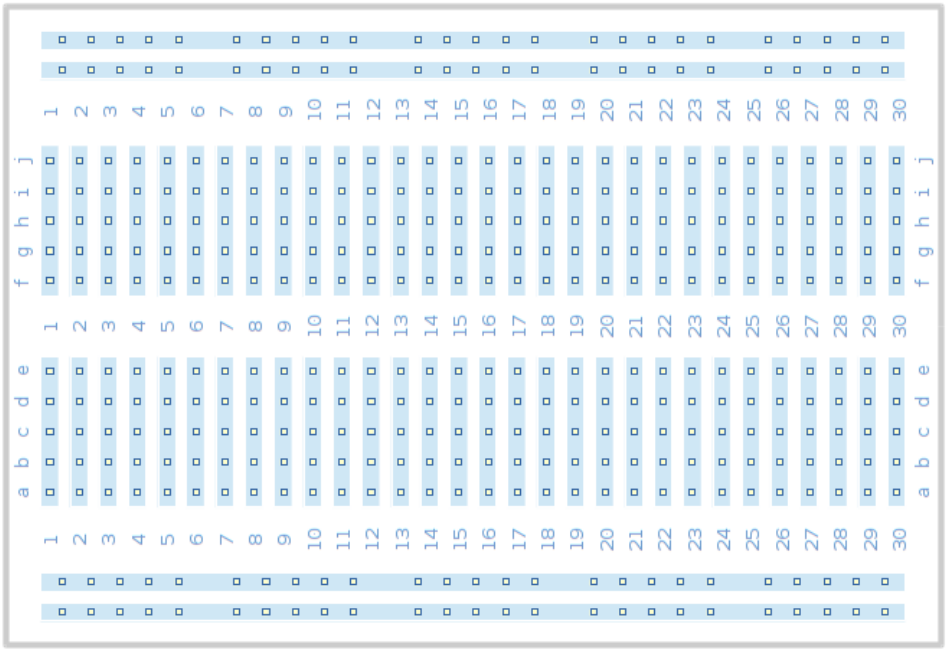
- **Evite usar as posições de baixa corrente para medir corrente em circuitos com relés, motores e transformadores.** Mesmo que consumam baixas correntes, esses componentes provocam breves pulsos muito altos de corrente (acima de 200mA) que podem queimar o fusível mesmo antes que qualquer medição seja feita.

O multímetro DT830B deve ter um **fusível** que irá se romper se uma corrente maior que 0,2 A for aplicada no multímetro. Se for um de melhor qualidade, você pode abrir o multímetro e trocar o fusível se ele queimar, por outro de 0,2 A. Mas **alguns fabricantes ou soldam o fusível no lugar, ou mesmo substituem ele por um fio (ou seja, não há fusível)**. Neste caso, se o amperímetro queimar, você não conseguirá mais usá-lo (mas é possível que as funções de voltímetro e ohmímetro continuem a funcionar).

Este multímetro é vendido por valores que variam de 12 a 50 reais. Se for comprar outro, verifique se ele permite a troca do fusível e da bateria. Os muito baratos costumam ser quase descartáveis. Existem também outros modelos (mais caros) que têm melhor precisão na medição, medem resistências maiores, apitam para medir continuidade, além de medirem frequência e capacitância.

7.6.2. Protoboard de 830 pontos

O protoboard (também chamado de breadboard ou base de prototipagem) incluído no kit corresponde a duas vezes o esquema abaixo (60 linhas) e tem as trilhas laterais interrompidas no meio (nem todos têm essa interrupção, e alguns têm mais de uma).



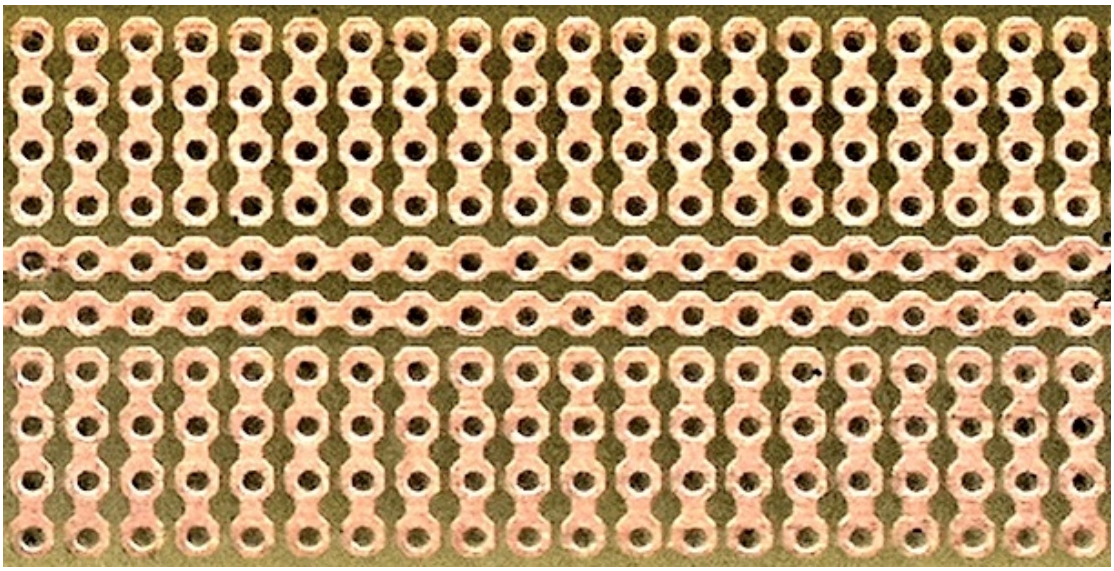
Para interligar um componente nas colunas centrais (a-j), coloque seus terminais **em linhas diferentes** (interligar um componente na mesma linha numerada é juntar todos os seus terminais no mesmo ponto; ligar uma bateria assim causa um curto-circuito).

Use as laterais para conexões que têm muitas ligações. Normalmente duas laterais são usadas para as entradas positivo e negativo da bateria. Alguns protoboards vêm com essas colunas marcadas com "+" e "-". Uma boa prática é usar apenas uma fila de cada lado, e escolher um lado para o positivo e outro para o negativo, para evitar o risco de inverter polaridades.

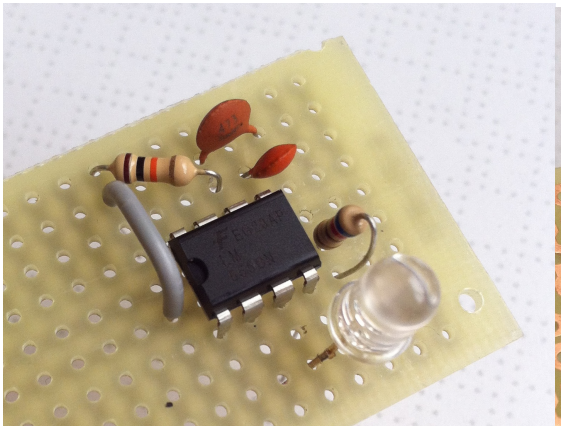
Veja mais neste tutorial (Sparkfun): https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard.

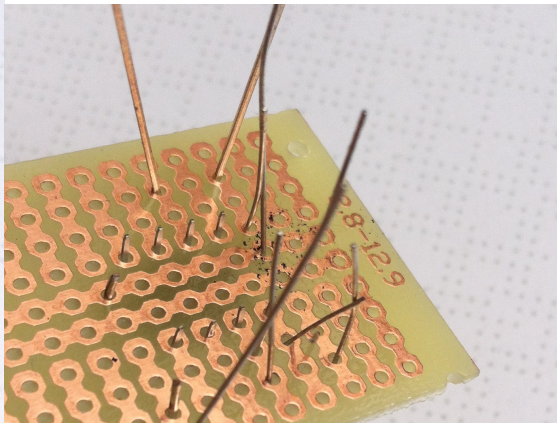
7.6.3. Placa de circuito impresso universal

Não existe um padrão para este tipo de placa. O desenho abaixo corresponde à placa incluída no kit. Use as fileiras do meio para os terminais positivo e negativo (correspondentes às laterais do protoboard) e as outras para encaixar componentes e circuitos integrados (que cabem perfeitamente no centro.



Insira os componentes pelo lado que não tem cobre, e solde os terminais do lado do cobre.





A solda adere com facilidade ao cobre. Verifique as conexões antes, pois é mais trabalhoso remover o componente depois de soldado. Após a soldagem, corte os terminais.

Você também pode usar uma placa perfurada (sem cobre) ou mesmo perfurar superfícies de plástico, cartão, tecido, couro e madeira para montar circuitos, soldando, amarrando ou fixando os terminais do outro lado. É muito importante que as conexões sejam **firmes** (para evitar mal contato) e **bem isoladas** (para evitar contatos indevidos e curto-circuitos que podem alterar o funcionamento ou até danificar o circuito).

8. Mini-referência de Arduino

Esta seção contém um resumo das especificações e programação da placa **Arduino Nano**, que faz parte do kit. Veja mais informações e exemplos no capítulo sobre Projetos com Arduino.

8.1. Programação

As seções abaixo contém apenas tópicos de programação que foram abordados na apostila. Para uma referência mais completa e abrangente, consulte o site oficial: http://arduino.cc.

8.1.1. Sintaxe básica

Um programa Arduino é escrito na linguagem Processing (derivada de C++) e chamado de sketch. Um sketch é armazenado no computador como uma pasta que contém os arquivos que compõem o programa. Um sketch possui no mínimo um arquivo, gravado com a extensão .ino, mas pode conter outros, por exemplo, arquivos .h.

A estrutura básica de um sketch consiste da declaração das funções loop() e setup() que são chamadas automaticamente pelo Arduino. A função setup() é chamada uma vez e a função loop() é chamada dentro de uma repetição infinita. Para escrever um programa para Arduino, instruções e chamadas de outras funções devem ser declaradas dentro das definições de loop() e setup().

O sketch mínimo está listado abaixo.

```
void setup() {

}

void loop() {

}
```

É importante fazer upload do sketch mínimo antes de construir o circuito de um Arduino, para testar a comunicação, e para garantir a segurança dos pinos (eles são todos inicializados como entradas de alta-impedância).

Instruções terminam sempre em ponto-e-vírgula. Definição de funções, estruturas de repetição e estruturas condicionais têm parênteses para declarar parâmetros, e um bloco delimitado por chaves para declarar outras instruções.

Variáveis e constantes

Variáveis são declaradas dentro ou fora de funções. Se declaradas fora das funções são variáveis globais e podem ser usadas em qualquer lugar. Se declaradas dentro das funções, como nos parâmetros de uma definição de função, só podem ser usadas dentro da função.

A declaração de variáveis consiste do seu tipo e de um nome, usado como identificador:

```
int pino;
```

A variável pode ter um valor inicial, que é atribuído a ela pelo operador "=" de atribuição:

```
int  pino = 5;
```

Se a declaração de uma variável for precedida pela palavra reservada "const", ela será tratada como uma constante e não poderá ser mais modificada.

```
const int pino = 5;
```

Variáveis pode ser usadas para gravar vários tipos de dados. Nesta apostila usamos variáveis para guardar números de pinos (que são inteiros), valores decimais (ponto-futuante) e outros valores inteiros. Para isto declaramos as variáveis como sendo dos tipos int (inteiro) e float (decimais).

Atribuição

A operação de atribuição copia o valor do lado direito do sinal de "=" para a variável do lado esquerdo.

```
float tensao = 12.5;
```

Essa operação também acontece quando uma função é chamada com parâmetros. Os valores dos parâmetros na chamada da função são copiados para as variáveis declaradas na definição da função. Por exemplo, para usar a definição de função abaixo:

```
void mostrarTensao(float tensao) {
    Serial.println(tensao);
}
```

Fazemos uma chamada da forma:

```
imprimirTensao(12.5);
```

que é equivalente à atribuição mostrada anteriormente (o valor 12.5 será copiado para a variável *tensao*).

Durante uma atribuição, o lado direito da expressão é calculado primeiro, e quaisquer variáveis são substituídas pelos valores que contém. Por isso é possível fazer expressões como:

```
int x = 5;
x = x + 1;
```

Onde o **x** do lado direito é substituído pelo **valor anterior de x** (5), depois é realizada a soma de 5 + 1, e por fim o **novo valor** (6) é atribuído à variável **x**.

Operações aritméticas

Operações aritméticas podem ser feitas envolvendo números e variáveis (que contém números). Todas funcionam com tipos float. Divisões entre inteiros requerem que um dos números seja float. Isto pode ser feito acrescentando um ponto-decimal e um zero:

```
float resultado  = 12.0 / 10;
```

As operações básicas são soma (+), subtração (-), multiplicação (*) e divisão (/). Nesta apostila também usamos a operação de incremento: x++, que é o mesmo que fazer x = x + 1.

Existem várias outras operações aritméticas que o Arduino suporta, além de funções matemáticas úteis. Consulte a documentação do Arduino no site arduino.cc para mais detalhes.

Operações de lógica relacional e booleana

As principais operações lógicas relacionais são igual (==), diferente (!=), maior-que (>), menor-que (<), maior ou igual (>=) e menor ou igual (<=). Elas geralmente são usadas em parâmetros ou blocos que esperam expressões condicionais, como blocos if() e a segunda parte da declaração de uma repetição for():

```
if(digitalRead(6) != HIGH) { ... }

for(int i = 0; i < analogRead(0); i++) { ... }
```

Valores lógicos podem ser conectados formando expressões maiores usando proposições **OU** (OR) e **E** (AND), que são representados pelos símbolos **||** e **&&**, respectivamente. Por exemplo:

```
if(x >= 5 && y == 9) { ... }
```

Executa o bloco se **x** for maior que 5 **e** se y for igual a 9.

```
if(digitalRead(2) == HIGH || analogRead(2) > 512) { ... }
```

Executa o bloco se o valor lido no pino digital D2 for 5V, **ou** se o valor lido no pino analógico A2 for 2,5V (metade de 5V).

Estrutura condicional: if

O bloco if() executa seu conteúdo apenas se a expressão entre parênteses for verdadeira. É importante que essa expressão seja uma expressão de lógica relacional (igualdade, diferença, maior ou menor que) e/ou booleana (expressões OR "||" e AND "&&"). Exemplos de uso:

```
if(digitalRead(6) != HIGH) {
    analogWrite(9, 128);
}
```

Um bloco if pode ser seguido por um bloco else, que irá conter instruções que serão executadas se a expressão do if não for verdadeira.

```
if(digitalRead(6) != HIGH) {
    analogWrite(9, 128);
} else {
    analogWrite(9, 0);
}
```

Entre os blocos if e else pode também haver zero ou mais expressões else if, que testam situações intermediárias. Por exemplo:

```
if(digitalRead(6) != HIGH) {
    analogWrite(9, 128);
} else if (analogRead(0) > 512 && analogRead(1) < 255) {
    digitalWrite(6, LOW);
} else {
    analogWrite(9, 0);
}
```

Estrutura de repetição: for

A estrutura de repetição for tem uma declaração entre parênteses seguido por um bloco contendo instruções:

```
for(int i = 0; i < 6; i++) {
    analogWrite( 9, i * 51);  // i varia de 0 a 5 (produzirá 0 a 255)
}
```

A declaração contém três partes:

- Uma **inicialização de variável** (a variável que será testada para decidir se a repetição continua). Esta inicialização acontece apenas uma vez.
- Uma **expressão condicional para testar a variável**. Este teste ocorre antes de cada repetição.
- Uma expressão para **alterar o valor da variável** (incrementar ou decrementar). Esta operação acontece no final de cada execução do bloco entre chaves.

O código entre as chaves no bloco for será executado zero ou mais vezes, dependendo do resultado da expressão condicional.

Definição de funções

Funções são definidas declarando seu **tipo**, nome usado como **identificador**, sua lista de **parâmetros** (declarando o tipo de cada um) e um **bloco entre chaves** contendo as instruções que irá executar quando for chamada. Por exemplo:

```
void piscar(int pino, int tempo) {
    digitalWrite(pino, HIGH);
    delay(tempo / 2);
    digitalWrite(pino, LOW);
    delay(tempo / 2);
}
```

Todas as funções que vimos nesta apostila eram do tipo void – que não retornam valor, mas é possível definir funções que retornam valor, como por exemplo a função analogRead() que devolve um valor int como resposta. A função abaixo soma dois números e retorna o resultado como int:

```
int soma(int x, int y) {
    return x + y;
}
```

Chamada de funções

A sintaxe para chamar uma função consiste em usar seu nome como um comando, e passar valores ou variáveis compatíveis com o tipo e quantidade de seus parâmetros. Para chamar a função piscar() acima, podemos usar:

```
piscar(5, 1000);
```

A chamada irá atribuir o valor 5 à variável pino, e o valor 1000 à variável tempo, e a função executará comandos que farão o pino piscar uma vez, com 500ms entre cada pulso.

Para chamar uma função que retorna valor, como a função soma() que definimos acima, provavelmente iremos querer guardar o valor retornado em uma variável:

```
int resultado = soma(5,9);
```

A chamada irá atribuir o valor 5 para x, e o valor 9 para y dentro da definição de função, que vai somar os números e retornar. O valor de retorno será guardado na variável resultado.

Listas indexadas

Listas (ou arrays, ou vetores) guardam uma coleção de itens de um determinado tipo em uma variável, que podem ser referenciados através de seu **índice** (posição dentro da lista). O índice **começa em zero**. Existem várias formas de definir uma lista. As listas que usamos nesta apostila foram definidas declarando explicitamente seus componentes entre chaves.

```
int pinos = {3, 5, 6, 9, 10, 11};
```

O número de componentes representa o tamanho da lista. O índice varia de 0 a tamanho-da-lista – 1. Para recuperar um elemento, use o índice entre colchetes. O comando abaixo grava o valor máximo de tensão no pino 9 (quarto pino da lista acima, índice 3):

```
analogWrite( pinos[3], 255);
```

Os índices passados podem ser usados também para trocar o valor de um elemento da lista, através de uma atribuição. O tipo de dados deve ser compatível:

```
pinos[0] = 13;  // muda a lista para {13, 5, 6, 9, 10, 11}
```

Listas são frequentemente usadas em estruturas de repetição, onde o índice é passado como uma variável:

```
for(int i = 0; i < 6; i++) {
    analogWrite( pinos[i], 255 );
}
```

A repetição for repete o bloco seis vezes, com um valor diferente de i em cada repetição. O valor de i varia de 0 a 5, assim o bloco produz o valor máximo de tensão PWM em cada um dos seis pinos listados na lista pinos[].

Diretiva #include

É usado para incluir arquivos de cabeçalho (.h) e bibliotecas em um sketch.

Os **arquivos de cabeçalho** devem estar na mesma pasta e são incluídos pelo nome, entre aspas (e sem ponto e vírgula no final):

```
#include "funcoes.h"
```

Bibliotecas precisam estar previamente **instaladas** no ambiente do Arduino. Use a opção de menu Sketch / Include Library / Manage Libraries para baixar e instalar novas bibliotecas. Uma vez instaladas, elas poderão ser **incluídas** chamando o header da biblioteca entre < e >:

```
#include <FastLED.h>
```

Geralmente a instalação de bibliotecas adiciona vários exemplos de uso que podem ser acessados e usados a partir do menu File / Examples da IDE do Arduino.

Diretiva #define

Nesta apostila usamos **#define** para definir constantes (como alternativa a usar **const int**). Por exemplo, declarar:

```
# define PINO_LED 8
```

Tem o mesmo efeito prático que:

```
const int PINO_LED = 8;
```

Em programas Arduino, declarar constantes globais usando const int ou #define é uma questão de estilo. É importante saber reconhecer os dois casos e entender como funcionam.

8.1.2. Variáveis e constantes do Arduino

O Arduino define diversas constantes e variáveis com valores prévios. Eles representam opções e níveis lógicos.

Estados lógicos

As constantes inteiras HIGH e LOW contém respectivamente os valores 1 e 0, e representam níveis lógicos dos pinos digitais do Arduino. No Arduino Nano, esses níveis lógicos representam respectivamente os valores de tensão 5 volts, e 0 volts.

HIGH e LOW podem ser usados em testes:

```
if (digitalRead(6) == HIGH) { ... }
```

E em funções que alteram o nível lógico dos pinos:

```
digitalWrite(9, LOW);
```

Finalidade de um pino (pinMode)

A função **pinMode()** permite alterar a finalidade de um pino, configurando-o como entrada, saída ou ligando-o a um resistor de pull-up ou um LED. Na apostila usamos três opções de pinMode():

- `OUTPUT` – configura o pino como saída digital.
- `INPUT` – configura explicitamente o pino como entrada (opcional – ele já é naturalmente entrada)
- `INPUT_PULLUP` – configura o pino como entrada ligada a um resistor interno de pull-up, para que o estado inicial do pino seja HIGH.

8.1.3. Funções do Arduino

As funções abaixo incluem apenas as que foram abordadas na apostila. Para uma referência mais completa, consulte a documentação no site oficial do Arduino.

analogRead(pino-analógico)

Esta função retorna um valor inteiro de **0 a 1023** correspondente ao nível de tensão em um dos 8 **pinos analógicos**. Normalmente o pino deve ser ligado a um divisor de tensão (entre os pinos 5V e GND) que pode ser o próprio sensor (se ele tiver 3 terminais) ou no meio da ligação em série de um resistor com o sensor (se ele tiver 2 terminais).

```
int leitura = analogRead(2);
```

analogWrite(pino-digital-pwm, valor)

A função recebe dois parâmetros. O primeiro deve ser um **pino digital que suporte PWM**. O segundo é um número inteiro de **0 a 255**. A função irá gerar no pino correspondente um **valor médio simulado** de tensão proporcional ao valor passado de 0 a 255.

```
analogWrite(10, 127);  // produz pulso PWM com ciclo de trabalho de 50%
```

A função analogWrite não produz saída digital verdadeira. É PWM, ou seja, é uma onda quadrada que alterna entre os valores 0 e 5V, em frequência entre 500 e 1KHz (500 a 1000 vezes por segundo), variando o tempo que o pulso está no estado HIGH (ciclo de trabalho). O tempo será proporcional ao valor de 0 a 255 passado pela função. Isto produz um valor médio que simula variação de tensão e permite variar brilho de LEDs e velocidades de motor.

Algumas aplicações requerem sinais analógicos verdadeiros. Neste caso é preciso construir um filtro RC na saída PWM (calcular um capacitor e resistor) para retificar a onda e gerar um valor constante.

digitalRead(pino-digital)

A função retorna o valor lido no pino digital correspondente, que será retornada como HIGH ou LOW. No Arduino Nano pode-se também ler dos pinos analógicos A0 a A7 (que podem ser identificados desta forma, ou através dos números 14 a 19).

```
int estado = digitalRead(8);
```

Se o valor for um pouco abaixo de 5V ele ainda será interpretado como HIGH até o limite de 3V. Havendo menos de 3V no pino, o Arduino irá devolver o valor LOW como resposta.

O valor lido por digitalRead() sempre será HIGH ou LOW, mesmo quando a entrada tiver valor indefinido. Para evitar valores indefinidos, o ideal é configurar o pinMode como INPUT_PULLUP, que garante um valor inicial HIGH. A função digitalRead() também pode ser usada para ler o nível lógico de pinos configurados como OUTPUT.

digitalWrite(pino-digital, HIGH ou LOW)

Esta função requer que o pino esteja habilitado como OUTPUT, através da instrução pinMode(). Ele produzirá na saída uma tensão de 0 ou 5V, conforme o valor passado como parâmetro, respectivamente LOW ou HIGH.

```
digitalWrite(4, LOW);
```

Produzir HIGH não significa necessariamente que a saída irá ter corrente. Isto depende da diferença de potencial entre a saída e a referência onde ela estiver conectada. Se a referência for 5V, a diferença de potencial será zero e HIGH irá desligar o componente na saída (que será ativado com LOW). Veja o exemplo usando o LED RGB anodo-comum, que tem esse comportamento.

delay(tempo-em-milissegundos)

A função **delay()** interrompe o programa por no mínimo o tempo especificado em milissegundos.

```
delay(2000);  // interrompe por 2 segundos
```

Usar delay() com intervalos curtos (tipicamente de 10 segundos) é importante para lidar com sinais enviados por chaves, que durante o processo de ligar e desligar podem gerar vários pulsos que serão recebidos pelo Arduino.

tone(pino, frequência, duração)

Gera, no pino indicado, uma onda quadrada na frequência especificada, por um tempo determinado. Isto pode ser usado para gerar som (ou mesmo fazer um pisca-pisca se forem usadas frequências muito baixas).

O comando abaixo gera no pino 9 uma frequência de 523 Hz por 1 segundo:

```
tone(9, 523, 1000);  // 523 Hz corresponde ao dó central do piano
```

noTone(pino)

Muda o estado do pino para LOW:

```
noTone(9);
```

pinMode(pino, função)

Esta instrução é **obrigatória** para configurar pinos como saída digital ou como entrada usando resistores pull-up. Nos outros casos é **opcional** (a saída analógica automaticamente muda a função do pino quando chamada, e as funções de entrada sem pull-up são o estado natural dos pinos).

A função pinMode() deve ser chamada dentro de setup():

```
void setup() {
    pinMode(3, OUTPUT);  // configura o pino 3 como saída digital
}
```

8.2. Placa Arduino Nano

Esta seção contém um resumo e **não se refere ao Arduino Nano original**, mas ao clone fabricado na China que é usado na oficina (as especificações são um pouco diferentes).

8.2.1. Especificações técnicas

- *Microcontrolador*: ATMega328
- *Arquitetura*: AVR
- *Tensão de operação*: 5V
- *Consumo*: 19 mA
- *Clock*: 16 MHz
- *Memória flash*: 32kB (2kB são usados pelo bootloader)
- *Pinos analógicos*: 8 (A0 – A7)
- *Pinos digitais de entrada e saída*: 22 (1-19 + 5V + 3V3 + GND)
- *Pinos de saída PWM*: 6 (3, 5, 6, 9, 10, 11)
- *Saídas reguladas*: 3,3V (3V3) e 5V (5V)
- *Tensão de alimentação via pino VIN*: 6 a 12V
- *Tensão de alimentação via USB*: 5V
- *Tensão de alimentação via pino 5V* (evite, e não use Vin nem USB): 3,7 a 5,5V
- *Corrente máxima por pino de entrada/saída*: 40mA (máximo de 200mA no total)
- *Corrente máxima nos pinos 5V*: 500mA (depende da fonte de alimentação)
- *Corrente máxima na saída 3V3*: 25mA
- *Taxa de comunicação serial (CH340)*: 2400 a 115200 bps
- *Dimensões*: 18 x 45 mm
- *Peso*: 7g

Fontes e datasheets:

- https://www.arduino.cc/en/Main/arduinoBoardNano
- http://actrl.cz/blog/index.php/2016/arduino-nano-ch340-schematics-and-details/
- https://cdn.sparkfun.com/datasheets/Dev/Arduino/Other/CH340DS1.PDF

Esquema:

- http://actrl.cz/blog/wp-content/uploads/nano_ch340_schematics.pdf

8.2.2. Pinagem

Veja um diagrama detalhado com a pinagem do Arduino Nano no capítulo desta apostila sobre Projetos com Arduino.

9. Links e referências

9.1. Referências bibliográficas

9.1.1. Livros

[1] Charles Platt. **Make: Electronics. Second Edition**. Maker Media, 2015. *Este livro foi usado como referência para vários assuntos abordados na oficina e na apostila, e serviu de inspiração para alguns exemplos. É uma introdução à eletrônica para leigos que explora em detalhes os conceitos mais importantes, através de vários experimentos.*

[2] **Revista Bê-a-bá da Eletrônica**. **Números 1 a 20.** Editor: Bártolo Fittipaldi. São Paulo, 1982 a 1984. *Pouco menos de 30 edições foram publicadas na década de 80 e influenciaram uma geração de entusiastas de eletrônica na época. Ela contém vários circuitos simples e tutoriais com uma abordagem didática voltada para iniciantes. Alguns exemplos do curso foram baseados em circuitos publicados nesta revista.*

[3] Paul Scherz and Simon Monk. **Practical Electronics for Inventors. Fourth Edition.** McGraw-Hill, 2016. *Um livro enorme (mais de 1000 páginas) com uma abordagem prática e aprofundada de eletrônica e referências detalhadas de componentes, ferramentas, técnicas, etc.*

[4] Harry Kybett. **Electronics – A Self-teaching guide. 2nd. Edition.** John Wiley & Sons, 1986. *Um pequeno guia prático focado nos aspectos teóricos da eletrônica (lei de Ohm, leis de Kirchhoff, capacitores, resistores, divisores de tensão, transistores)*.

[5] Massimo Banzi & Michael Shiloh. **Make: Getting Started with Arduino. 3rd Edition.** *Uma breve introdução à plataforma Arduino pelos criadores da plataforma.*

[6] Simon Monk. **Programming Arduino: getting started with sketches. Second Edition.** McGraw-Hill, 2016. *Este livro proporciona uma introdução abrangente a Arduino através de vários exemplos e projetos. O autor também disponibiliza um tutorial de Arduino em 18 lições no site da AdaFruit:* https://learn.adafruit.com/series/learn-arduino *que inspirou alguns experimentos de Arduino desta apostila.*

[7] Bill Urmenyi. **Electronics for Artists**. Self-published. CD-ROM, 2001. *Vários capítulos do livro são acessíveis via Google Books. O autor vende o livro em formato digital através do seu site* http://www.urmenyi.co.uk/.

[8] Simon Quellen Field. **Electronics for Artists: Adding Light, Motion and Sound to your Artwork**. Chicago Review Press, 2015. *O autor disponibilizou o conteúdo do livro em um site interativo em* http://artists.scitoys.com/

[9] Ken Rinaldo. **Interactive Electronics for Artists and Inventors**. 2015. *O livro está disponível para download ou visualização online no site do artista* http://www.kenrinaldo.com/publications/.

[10] Emily Lovell. **Getting Hands-on with Soft-Circuits**. *Um guia para construir circuitos de eletrônica para vestir. Disponível online em* http://alumni.media.mit.edu/~emme/guide.pdf

9.1.2. Apostilas e material de cursos

[11] Mark Allen. **Electronics for Artists**. Machine Project, Los Angeles, 2007. *Apostila da oficina descrita em* http://machineproject.com/2007/workshops/electronics-for-artists/. *Slides da oficina e outros materiais ainda estão online em* http://machineproject.com/files/pdf/.

[12] Iain Sharp. **Electronics for Artists. 4th Revision**. 2010. *Apostilas de cursos (e outros materiais) disponível em* http://lushprojects.com/electronicsforartists/ *e* http://lushprojects.com/absolutebeginners/

9.1.3. Manuais

[13] **Guia oficial do Arduino.** https://www.arduino.cc/en/Guide/HomePage. *Contém referência completa da linguagem, especificação das placas, tutoriais e exemplos.*

9.1.4. Revistas online, tutoriais, vídeos e blogs

[14] **Instructables**. http://www.instructables.com/. *Um site que nasceu do laboratório multimídia do MIT e se tornou um dos mais populares portais colaborativos da comunidade "maker" DIY (Do-It-Yourself), com mais de 100 mil projetos, muitos usando eletrônica e Arduino.*

[15] **Makezine**. http://makezine.com/projects/. *Revista online com vários artigos e tutoriais com projetos detalhados passo-a-passo usando eletrônica e Arduino.*

[16] **DIY Hacking.** https://diyhacking.com/. *Outro portal similar aos anteriores, porém mais focado em projetos eletrônicos (que são classificados em niveis de complexidade).*

[17] **Electronics Tutorials**. http://www.electronics-tutorials.ws/ *Diversos tutoriais detalhados e ilustrados sobre conceitos fundamentais da eletrônica, escritos para iniciantes. Veja especialmente os tutoriais sobre resistores, capacitores, diodos e transistores, que estão relacionados a temas abordados nesta apostila.*

[18] **Random Nerd Tutorials.** http://randomnerdtutorials.com/. *Uma coleção de tutoriais de eletrônica básica, Arduino, ESP8266 e Raspberry Pi, com foco em automação residencial. Os tutoriais simples e didáticos sobre o circuito integrado 555, funcionamento de transistores, chaves e potenciômetros foram usados como referência para esta apostila.*

[19] **Blog do Arduino**. https://blog.arduino.cc/ *Blog oficial do Arduino, em inglês. Contém tutoriais de projetos com Arduino.*

[20] **Tutoriais da AdaFruit**. https://learn.adafruit.com. *Tem seções sobre wearables, Arduino e outras, com tutoriais detalhados.*

[21] **Tutoriais da Sparkfun.** https://learn.sparkfun.com. *Similares aos da AdaFruit, também com seções sobre wearables, Arduino, etc.*

[22] **Collin's Lab**. https://www.youtube.com/playlist?list=PLjF7R1fz_OOU08_hRcayfVZSmTpBCGJbL *Uma série de vídeos sobre fundamentos da eletrônica, técnicas e conceitos. É parte do canal da AdaFruit.*

[23] **Instituto Newton C. Braga**. http://www.newtoncbraga.com.br/ *O autor é um grande divulgador da eletrônica como hobby, publicou vários livros e escreveu inúmeros artigos de eletrônica popular. No site há vários tutoriais sobre componentes, transistores, circuitos integrados, etc.*

[24] **Blog Fazedores**. http://blog.fazedores.com/. *Blog em português com artigos e tutoriais sobre Arduino, eletrônica e outros assuntos relacionados.*

[25] **Blog FilipeFlop**. *Um dos sites mais populares em português. Contém vários tutoriais usando Arduino e sensores.* http://blog.filipeflop.com/

[26] **Arduino&Cia**. http://www.arduinoecia.com.br/. *Outro blog em português com projetos com Arduino.*

9.2. Software

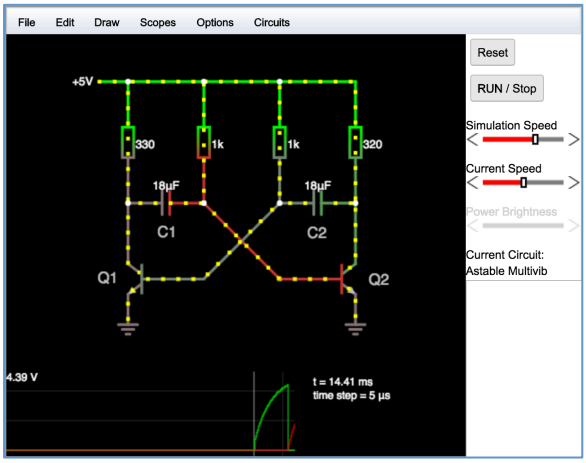
9.2.1. Editores e simuladores de circuitos

Estes aplicativos permitem desenhar circuitos online, e executar simulações sobre eles, calculando automaticamente correntes e tensões. Eles não são muito simples de usar, e requerem algum conhecimento de eletrônica, mas também podem ser usados apenas para desenhar esquemas. Alguns são gratuitos, outros são assinaturas pagas (mas podem ser usados de graça para circuitos simples):

1. **PartSim**.
   http://www.partsim.com
2. **EasyEDA**.
   https://easyeda.com
3. **Schematics Editor**.
   http://www.schematics.com/editor

4. **CircuitLab**.
   https://www.circuitlab.com/

5. **Simulador animado de circuitos** (Falstad)
   http://www.falstad.com/circuit/e-index.html



O autor disponibiliza várias animações interativas. Elas são ótimas para entender como funcionam diversos tipos de circuitos e componentes. Você pode interagir clicando na tela, mudando os valores dos componentes e variando correntes e tensões mexendo em controles que alteram a animação.

Selecionamos algumas animações abaixo, que estão relacionados a temas abordados nesta apostila:

Fundamentos de eletrônica

- Demonstração da Lei de Ohm: http://www.falstad.com/circuit/e-ohms.html
- Resistores em paralelo e divisor de corrente: http://www.falstad.com/circuit/e-resistors.html
- Resistores em série e divisor de tensão: http://www.falstad.com/circuit/e-voltdivide.html
- Carga e descarga de um capacitor: http://www.falstad.com/circuit/e-cap.html

Transistores

- Funcionamento de um transistor como amplificador: http://www.falstad.com/circuit/e-npn.html
- Funcionamento de um transistor como chave: http://www.falstad.com/circuit/e-transswitch.html
- Multivibrador astável com transistores: http://www.falstad.com/circuit/e-multivib-a.html
- Circuito biestável com transistores: http://www.falstad.com/circuit/e-multivib-bi.html
- Circuito monoestável com transistores: http://www.falstad.com/circuit/e-multivib-mono.html

Circuito integrado 555

- 555 monoestável: http://www.falstad.com/circuit/e-555monostable.html
- 555 astável (gerador de ondas quadradas): http://www.falstad.com/circuit/e-555square.html
- PWM com 555: http://www.falstad.com/circuit/e-555pulsemod.html
- Funcionamento interno do 555: http://www.falstad.com/circuit/e-555int.html

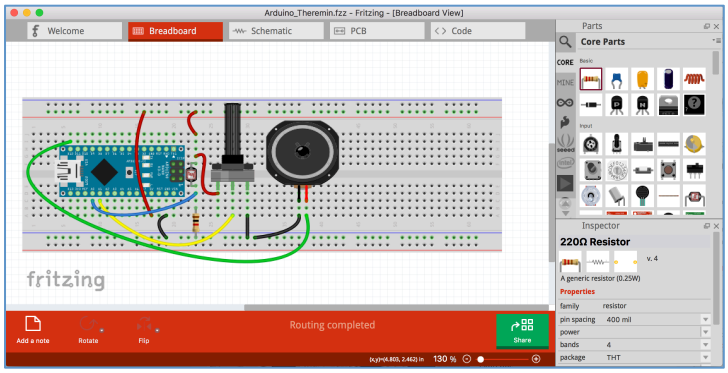
6. **EveryCircuit**
   http://everycircuit.com/

É uma app para iPad, mas também roda no Chrome, que permite criar simulações de circuitos animados. Há uma galeria com circuitos submetidos pelos membros da comunidade (não tão bem feitos quanto os do Falstad).

7. **Fritzing**
   http://fritzing.org/

Editor gráfico para desenhar circuitos, esquemas e placas.

Este aplicativo gratuito foi usado para desenhar todas as ilustrações de protoboard da apostila. Existe para Mac, Linux e Windows, e também gera um esquema associado ao protoboard (mas editar esquemas no Fritzing não é muito fácil) e placa para imprimir circuito impresso.



9.2.2. Calculadoras online

Você pode sempre calcular os circuitos você mesmo, mas é mais rápido e prático simplesmente digitar as informações que você tem em um formulário e deixar que um computador faça os cálculos para você. Os simuladores já fazem isto, mas as páginas a seguir podem ser mais fáceis de usar. As informações estão em inglês, e o uso das calculadoras requer um conhecimento mínimo sobre o que se deseja calcular.

1. **Calculadora de Lei de Ohm.**
   http://www.ohmslawcalculator.com/ohms-law-calculator

   Digite dois valores entre corrente, tensão, resistência e potência, e obtenha outros dois.

2. **Calculadora de divisor de tensão**
   http://www.ohmslawcalculator.com/voltage-divider-calculator

3. **Calculadora de resistor limitador de corrente** para **LEDs**
   http://www.ohmslawcalculator.com/led-resistor-calculator

4. **Conversor de milicandelas para** lumens (calculadora de luminosidade de LEDs)
   http://www.ohmslawcalculator.com/mcd-to-lumens

5. **Calculadora para circuito de multivibrador astável com** transistores
   http://www.homemade-circuits.com/p/transistor-astable-multibivrator-amv.html

   Esta calculadora obtém valores de resistores e capacitores, fornecendo-se a duração de cada pulso (ligado e desligado): T1 e T2, tensão de alimentação do circuito, tensões das junções do transistor (são típicos, você pode deixar os valores que são sugeridos ou usar zero), fator β (ganho) do transistor (use 100 a 250 para BC549) e corrente no coletor (onde ficará o LED).

6. **Calculadora de capacitores e resistores** para **multivibrador astável 555**
   http://houseofjeff.com/555-timer-oscillator-frequency-calculator/

   Recebe uma frequência ou intervalo de tempo e oferece várias opções de resistores (R1, R2) e capacitores (C) para configurar o 555. Indica também quantos % do pulso corresponde ao ciclo de trabalho (tempo em que o pulso está no estado ligado) para cada sugestão.

7. **Calculadoras de frequência para multivibrador astável** 555
   https://www.allaboutcircuits.com/tools/555-timer-astable-circuit/
   http://www.ohmslawcalculator.com/555-astable-calculator
   http://www.royalrife.com/555_calculator.html

   Estas calculadoras fazem o oposto da anterior. Elas recebem os valores de R1, R2 e C e calcula frequências e intervalos. É ideal quando você já tem valores fixos para dois dos componentes, e precisa calcular o terceiro.

8. **Caluladora para circuito 555 monoestável**
   http://www.ohmslawcalculator.com/555-monostable-calculator

   Informe dois valores (ex: capacitor e resistor, ou capacitor e largura do pulso, ou largura do pulso e resistor) que o programa calcula o valor restante.

9. **Calculadora de duração de bateria**
   https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-battery-life

   Na verdade, a conta é mais complicada. É preciso levar em conta o tipo de bateria e outros aspectos. Mas este cálculo permite uma boa estimativa.

10. **Calculadora de tempo de carga e descarga de capacitor**
    https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-time-constant

9.3. Fornecedores de material

Esta é uma lista incompleta de fornecedores de material usado nesta oficina e nos experimentos da apostila (baseada principalmente nos fornecedores de material para o kit).

No exterior (Arduinos, wearables, linha condutiva, acessórios, etc.)

- **AdaFruit** (New York): https://www.adafruit.com/
- **SparkFun Electronics** (Niwot, Colorado): https://www.sparkfun.com/
- **Loja do Arduino** (Europa): https://store.arduino.cc/

Em São Paulo (Sta. Ifigênia)

- **Dabi** (Componentes Eletrônicos). http://www.dabicomercio.com.br/. R. dos Timbiras, 299.
- **ELT** (Componentes Eletrônicos). R. dos Gusmões, 399.
- **Mult Comercial** (Componentes e Arduino). http://loja.multcomercial.com.br/ R. dos Timbiras, 257.
- **Mamute Eletrônica** (Robótica e Arduino). http://www.mamuteeletronica.com.br. R. Vitória, 125

Lojas online no Brasil

- **Ryndack** (Componentes Eletrônicos). https://www.ryndackcomponentes.com.br/
- **Soldafria** (Componentes Eletrônicos). http://www.soldafria.com.br/
- **FilipeFlop** (Robótica e Arduino). http://www.filipeflop.com/
- **RoboCore** (Robótica e Arduino). https://www.robocore.net/
- **Instituto Digital** (Robótica e Arduino) http://www.institutodigital.com.br/

10. Índice de experimentos

11. Material usado nos experimentos

A lista a seguir relaciona todos os componentes que fazem parte do kit distribuído para cada aluno (depende da duração e formato da oficina). Há também uma descrição dos principais componentes nesta apostila.

- 1 caixa organizadora de plastico
- 1 bolsa de tecido
- 1 apostila impressa (pasta com folhas avulsas)

- 50g de pó de ferro

- 1 limão ou batata
- 1 prego de zinco
- 1 pedaço de cobre
- 1 parafuso 7mm com porca
- 1 mangueira 5cm (para eletroímã)

- 0,5 metro de fita dupla-face tecido condutivo de prata 1cm de diâmetro
- 0,5 metro de fita adesiva tecido condutivo de prata 1cm de diâmetro
- 1 metro de fita adesiva de folha de cobre com 0,5cm de diâmetro
- 1 metro de linha de costura condutiva de aço inoxidável
- 5 metros de fio esmaltado AWG 26
- 2 imãs de neodímio N24/N35 com 10x4mm

- Óculos de proteção (para soldagem) de plástico
- 1 alicate de ponta-fina
- 1 multímetro digital DT-830B (mede voltagem, corrente e resistência)
- 1 base de contatos para montagens (protoboard-breadboard) com 830 furos

- Jumpers macho (fios) para conexões em protoboard e Arduino
- 1,2m de fio rígido AWG 22 para conexões em protoboard (30cm de preto, branco, vermelho, azul, verde, amarelo)
- 1 placa de fenolite circuito impresso universal
- 5 cabos de 20cm com garras jacaré
- 1 clip para bateria de 9V
- 1 plugue macho P4 com bornes (para usar em bateria 9V)
- 1 plugue fêmea P4 com bornes (para circuitos alimentados com fonte de 9V)

- 1 suporte para uma pilha AAA
- 2 pilhas AAA de zinco-carbono com 1,5V
- 1 bateria CR2032 de 3V
- 1 pegador de roupa de madeira (suporte para pilha CR2032)
- 1 fonte de 9V / 1A com saída P4
- 1 fotocélula (fonte de energia solar) de silício poli-cristalino 0,5V, 0,28W
- 1 célula piezoelétrica de 2,5cm
- 2 chaves tipo alavanca com duas posições
- 1 chave de pressão com duas posições
- 5 mini chaves tácteis (1 posição)
- 1 relé de 5V com 2 pólos e 2 posições
- 2 fusíveis pequenos de 0,2A (para multímetro)
- 1 mini motor de 3V
- 1 mini alto-falante de 8Ω
- 1 cigarra (buzzer) de 5V
- 1 microfone de eletreto

- 1 sensor magnético reed
- 1 sensor passivo de temperatura (termistor) NTC 10k
- 2 sensores passivos de luminosidade LDR 5mm e 7mm

- Resistores de ¼ de watt
  - 1 de cada: 10Ω, 220kΩ, 680kΩ, 2,2MΩ, 3,3MΩ
  - 2 de cada: 47Ω, 330Ω, 680Ω, 2,2kΩ, 3,3kΩ, 4,7kΩ, 6,8kΩ, 22kΩ, 33kΩ, 47kΩ, 68kΩ, 330kΩ, 470k Ω, 560kΩ, 1MΩ (1 000 000 Ω)
  - 5 de cada: 100kΩ (100 000 Ω)
  - 10 de cada: 100Ω, 1kΩ (1000 Ω), 10kΩ (10 000 Ω),
  - 14 de cada: 220Ω
  - 16 de cada: 470Ω
- Capacitores de cerâmica e poliéster
  - 1 de cada: 100pF, 3,3nF (3,3kpF)
  - 2 de cada: 1nF (1000pF, 1kpF), 4,7nF (4,7kpF) , 22nF (22kpF, 0,022 µF), 47nF (47kpF, 0,047 µF), 100nF (0,1 µF), 330nF (0,33 µF), 470nF (0,47 µF)
  - 8 de cada: 10nF (10kpF, 0,01 µF)
- Capacitores eletrolíticos
  - 1 de cada: 4,7 µF, 33 µF, 470 µF, 1000 µF, 2200 µF
  - 2 de cada: 2,2 µF, 47 µF, 220 µF
  - 3 de cada: 3,3 µF
  - 4 de cada: 1 µF, 10 µF, 22 µF, 100 µF

- 5 potenciômetros (resistores variáveis) com 5k, 10k, 20k, 50k e 100kΩ

- 12 LEDs de alto-brilho (transparentes) de 5mm – 2 de cada cor: vermelho, rosa, amarelo, verde, azul, violeta
- 4 LEDs de alto-brilho (transparentes) de 5mm de luz branca
- 10 LEDs difusos vermelhos de 5mm
- 1 LED amarelo de 3mm
- 1 LED RGB de alto-brilho 5mm e anodo-comum (4 terminais)
- 1 LED RGB 5050 (SMD 6 terminais)
- 1 LED infravermelho
- 1 LED RGB endereçável WS8212 com dissipador
- 1 display LED de 7 segmentos catodo comum

- 4 diodos de silício de propósito geral (1N4148 ou equivalente)
- 1 diodo Zener de 3,3V
- 1 sensor de temperatura de precisão LM35DZ (embalagem TO-92)
- 1 foto-transistor TIL 78 (embalagem LED 5mm)
- 4 transistores bijunção NPN de propósito geral (BC548, 2N3904, ou equivalente)
- 1 transistor bijunção PNP de propósito geral (BC558, 2N3906, ou equivalente)
- 1 transistor MOSFET de baixa potência (2N7000 ou equivalente)
- 1 transistor MOSFET de potência nível lógico (IRL540 ou equivalente)

- 1 circuito integrado 7805 (regulador de tensão 5V)
- 3 circuitos integrados 555 (temporizador)
- 1 circuito integrado 4017 (contador de década)
- 1 circuito integrado 4026 (decodificador para display de 7 segmentos)

- 1 placa microcontroladora compatível Arduíno Uno, Nano ou similar
- 1 cabo USB para Arduino

12. Sobre o autor

Helder da Rocha é um peixe abissal.

(Foto: Maurício Franco de Carvalho.)