PoCA | Portal de Cursos Abertos

Arduino para professores de música

Glauber Santiago

2020

ufscar

DEPARTAMENTO DE ARTES E COMUNICAÇÃO

Arduino para professores de música

Glauber Santiago

Sumário

Apresentação[1]

Já pensou em poder **construir** instrumentos e jogos musicais eletrônicos? Em arquitetar um equipamento no qual, ao se apertar um botão, algo acontece? Em criar um teclado musical gigante, para a diversão e interação dos estudantes com a música? Se estas questões lhe são interessantes, você chegou ao lugar certo.

Este livro fornece ao **leigo** em eletrônica e computação conhecimentos para elaborar aparatos eletrônicos que emitam sons e luzes com objetivos para o ensino de música. A principal tecnologia abordada é a do Arduino. Para atingir o objetivo, todos os elementos necessários serão apresentados, desde os conceitos de eletricidade e eletrônica, até elementos de computação e programação, sempre buscando ser o mais prático e útil para o público-alvo de educadores. A cada momento, serão apresentados exemplos práticos e simulações para que o leitor possa fazer experimentos e fixar o conteúdo. Os experimentos explorados são os seguintes: Tocador de duas notas, Teclado musical simples, Metrônomo visual e sonoro, Randômicos, *IFs*, Tocador de pulso com escala cromática, Tocador de pulso com frequências e Gerador melódico.

O **público-alvo** desta proposta são os professores de música e músicos em geral, que desejem aprender sobre estas tecnologias, desde o início; bem como quaisquer professores e pessoas que almejem navegar por estas águas. Aliás, o livro é apenas o início de um percurso que não tem fim, mas que prontamente possibilita a realização de várias aplicações uteis a uma prática docente diferenciada.

É **importante**, para o educador em geral, conhecer as tecnologias apresentadas neste livro para que possa empoderar-se destes recursos que, muitas vezes, estão circunscritos ao

[1] Assista esta apresentação em vídeo no link: <https://youtu.be/Z_xCFBtQBRo>

pessoal de ciências exatas em aulas de robótica e em *oficinas maker*. A ideia é que professores de quaisquer áreas possam envolver-se com projetos relacionados a esta vertente educacional.

Para entender o conteúdo de tecnologia deste livro, não existe **pré-requisito**, ao leitor. Sobre o conteúdo musical, pode ser que o leigo em música não entenda algum aspecto, mas isso não impede que ele consiga aproveitar a lógica geral trabalhada.

Espero que resulte em ótimos frutos.

Bom estudo!

São Carlos, 2020

Prof. Glauber Santiago

1. Introdução ao hardware

Arduino é uma tecnologia desenvolvida para a prototipagem rápida para a física computacional. Ou seja, uma forma de criação veloz de aplicações que envolvem computação em interação com ambientes físicos, reais.

Seriam exemplos para o uso de aplicações destas para o ensino de música: instrumentos musicais artesanais, jogos sonoros, simuladores musicais, etc[2]. Para uso educacional em geral poderíamos imaginar quaisquer aplicações que envolvam sons, luzes, botões e sensores diversos.

A tecnologia do Arduino envolve três aspectos a serem considerados: um de **hardware** (placas com componentes eletrônicos), um de **software** (plataforma para elaboração de programas, bibliotecas, a linguagem de programação em si, etc.) e um **legal** (possibilidade de utilizar a tecnologia sem muita necessidade de preocupações com direitos autorias[3]). A Figura 1.1 ilustra estes aspectos.

[2] Para verificar um exemplo de aplicação sonora acesse o seguinte link: https://youtu.be/xBUQeSavSpw

[3] Ao leigo na produção tecnológica este aspecto parece irrelevante, mas isto é porque ele não sabe que os produtos tecnológicos são submetidos, também, a regras de direitos autorais e patentes para sua produção. Portanto, com o Arduino tendo os direitos liberados, qualquer um pode utilizá-lo para produzir seus equipamentos, sem problema legal.

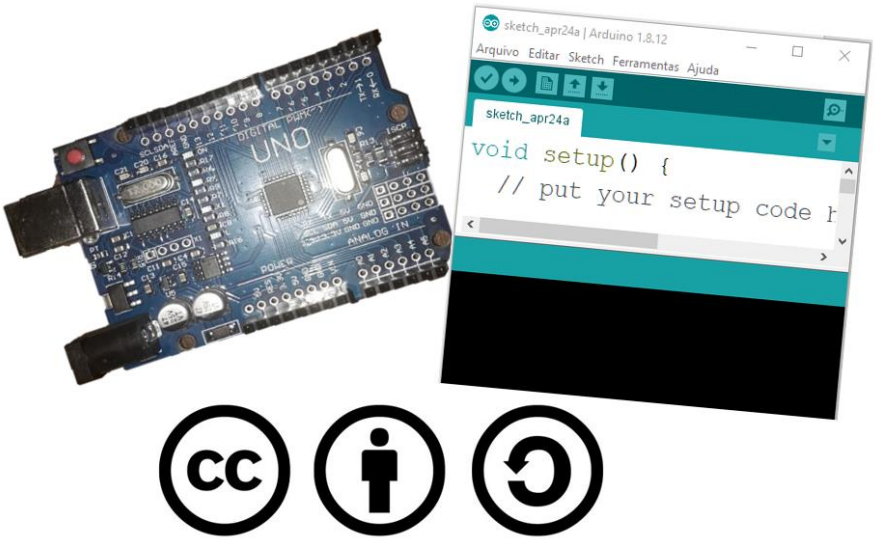


Figura 1.1 – Os 3 aspectos do Arduino: hardware, software e legal

Para o leitor entender qual a lógica geral de utilização destes aspectos, ele deve pensar assim, por exemplo: “Eu desejo criar um teclado musical simples para utilizar de forma lúdica em minhas aulas. Para tanto, irei adquirir uma placa de Arduino e outras pequenas coisas para montar o aparelho (hardware); e depois, irei fazer uma programação para dizer como o teclado funcionará (software). Para fazer esta programação, utilizarei um computador e depois irei carregar a programação no hardware para ele funcionar sozinho (sem o computador). Além disso, eu poderei ficar tranquilo, sem medo de problemas de direitos autorais por utilizar esta tecnologia. Posso até batizar o teclado com o meu nome”.

Ficou um pouco mais claro? Então, prossigamos com a explanação destes aspectos. Mas antes, vale à pena lembrar que mais à frente serão apresentados alguns apontamentos acerca de eletricidade e eletrônica. Ou seja, se restar algumas dúvidas sobre este tema, elas serão sanadas em uma sessão posterior.

1.1 Placas

Em relação ao hardware, existem diversos modelos de placas, conforme mostra a Figura 1.2. Na figura, da esquerda para a direita, temos os seguintes modelos: Lilypod, Nano, Uno e Mega.

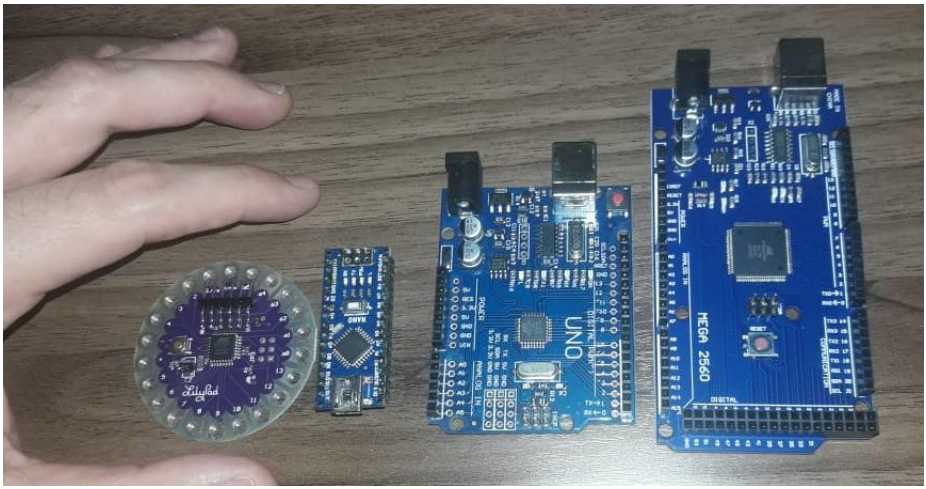


Figura 1.2 – Arduinos Lilypod, Nano, Uno e Mega

Neste livro vamos enfocar em uma das placas que existem com esta tecnologia, a placa *Arduino Uno*. Isso porque é uma das mais baratas em termos financeiros e possui um tamanho e conexões confortáveis para se trabalhar. Além do mais, tudo o que se aprende para uma placa serve para a outra. Assim, o *Arduino Uno* é a opção didática mais difundida. A Figura 1.3 apresenta uma placa compatível[4] com o *Arduino Uno*. Ao centro, temos o microprocessador (um quadradinho preto) que é o cérebro da placa. Os demais componentes são para alimentação, comunicação e outras coisas secundárias.

[4] Ou seja, não é fabricada na Itália, com a marca oficial *Arduino*, mas é totalmente compatível. Além disso, em termos de aparência, é praticamente idêntica.

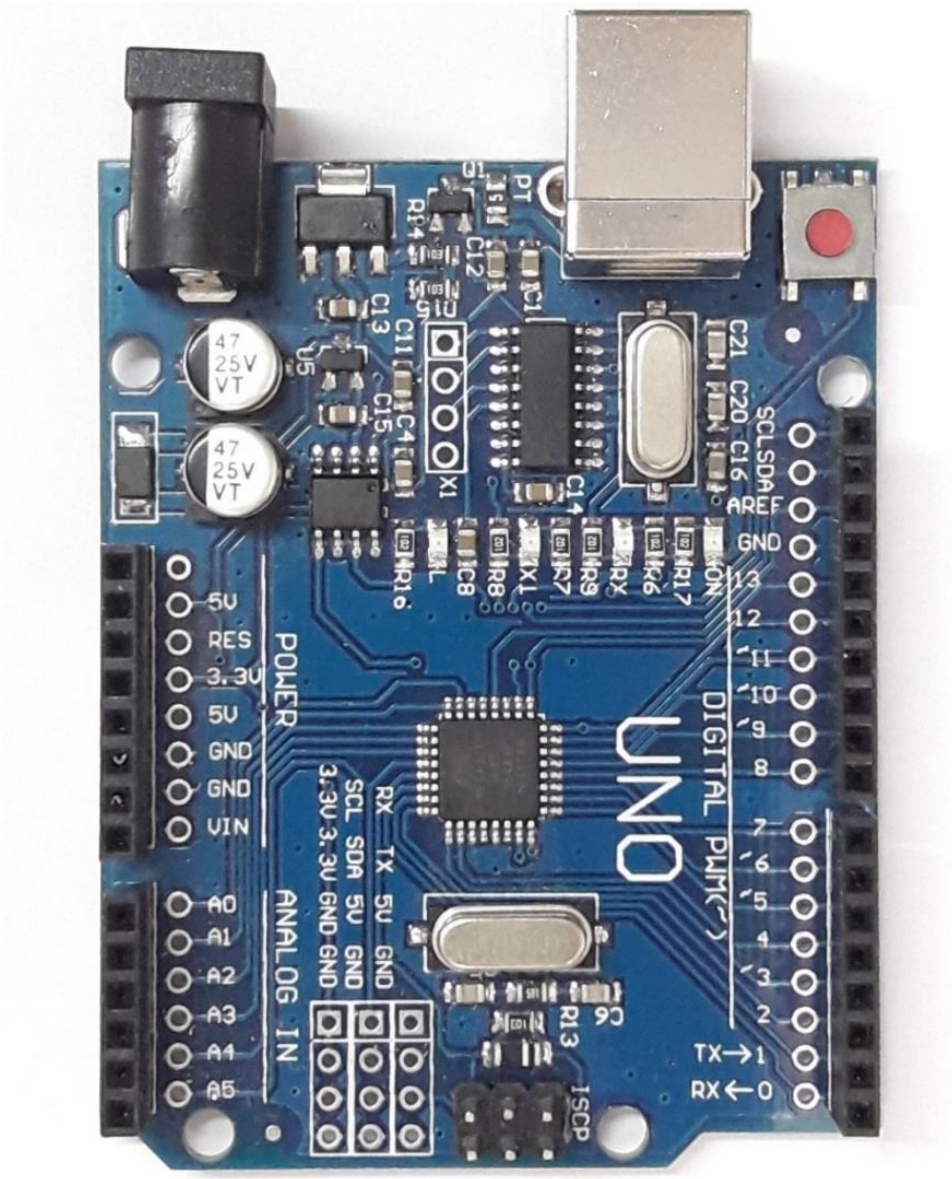


Figura 1.3 – Arduino Uno (compatível)

Em uma linguagem mais simples, pode-se dizer que uma placa de Arduino é um computador, só que, no lugar de mouse, teclado e monitor são ligados cabos de eletricidade. Observe-a

com cuidado para retirar quaisquer tabus tecnológicos que, eventualmente, você carregue. Não é nenhum bicho de sete cabeças. Você não tem que saber tudo, mas, aprenda a apreciar. Isso lhe dará confiança para continuar o percurso.

Na Figura 1.3, vemos conexões para ligar cabos. Estas conexões são duplas, ou seja, você pode ligar um cabo soldando um fio no orifício da placa (de cor prateada) ou pode encaixar um conector no terminal (de cor preta). Aqui, neste livro, não iremos utilizar soldagens. Deste modo, atente-se apenas aos terminais de cor preta na placa. Verifique o nome das conexões: Na esquerda vemos, de cima para baixo, as anotações: *5V, RES, 3.5V, GND, GND, Vin, A0, A1, A2, A3, A4* e *A5*. Na direita, vemos *SCL, DAS, AREF, GND* e depois de *13* até *0*[5]. Não se assuste com estas siglas e nem com a necessidade de memorizar estas informações. Para a proposta deste nosso estudo, só serão explorados elementos básicos e necessários. Busque apenas ir acompanhando o raciocínio da explanação.

As conexões (pinos) de 0 a 13 são chamadas de portas digitais[6]. Isto porque elas só funcionam emitindo ou recebendo um sinal fixo de 5 Volts[7]. Ou seja, é digital, pois na linguagem computacional utilizada 0 volts representa o número 0 **e** 5 Volts, o número 1. As conexões de *A0* a *A5* são portas analógicas[8]. Estas são portas apenas para entrada e são do tipo analógica. Assim, recebem sinais de 0 **a** 5 volts.

Então! Como funciona este computador que não tem teclado nem monitor? Vejamos um exemplo como o montado na Figura 1.4. Na montagem temos uma caixa de som com um terminal ligado no *GND* (do inglês *ground*, ou seja "terra" que é o **zero** volts)

---

[5] Em cor verde no texto indicamos as conexões que serão utilizadas neste livro.

[6] Considere que o fundamento da lógica digital é o sistema binário, com a utilização de, apenas, zeros e uns. Para entender melhor sobre o termo *digital,* aguarde a parte final do capítulo.

[7] Mais à frente no livro, haverá uma melhor explicação para estes termos e conceitos da eletrônica. Por enquanto, basta seguir a lógica da explanação.

[8] Para entender melhor sobre o termo *analógico,* aguarde a parte final do capítulo.

e o outro terminal ligado na porta 10 do Arduino. Além disso, temos um LED amarelo, com o terminal positivo ligado em um resistor (de cor azul)[9] que está ligado na porta 4 do Arduino. Já o terminal negativo do LED está ligado diretamente no outro *GND*.

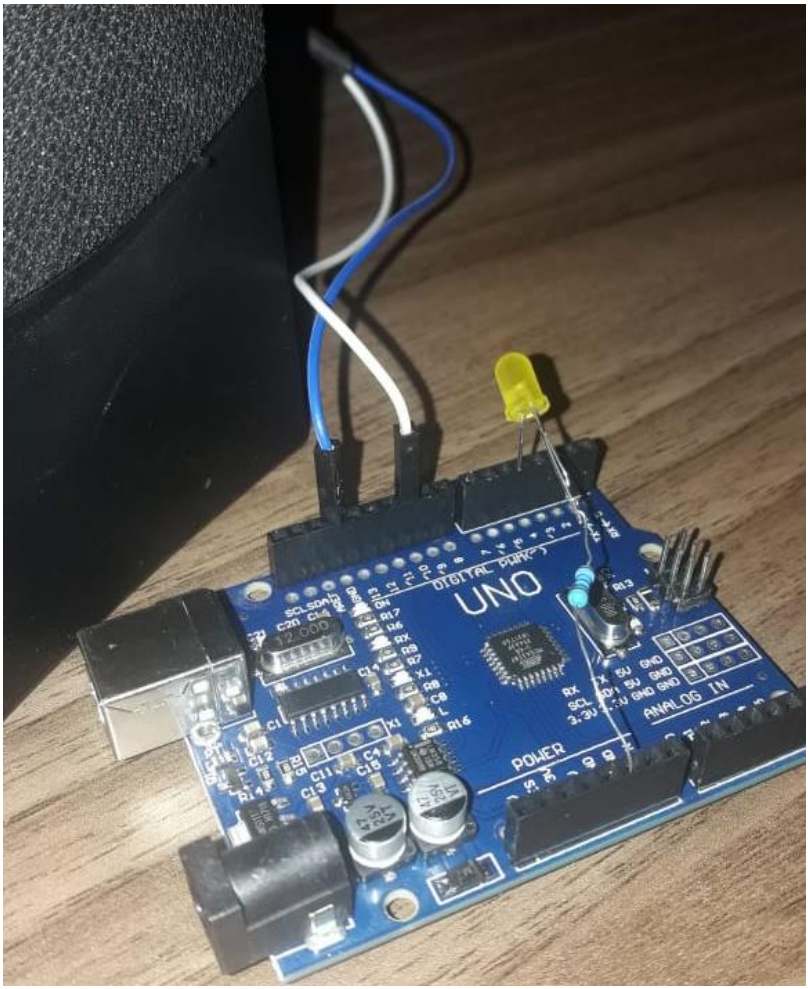


Figura 1.4 – Montagem com caixa de som e LED

[9] O resistor serve para fazer com que o LED não queime, porquanto os 5V da porta digital é muita tensão para ele. No máximo ele suporta uns 2,5 Volts.

Nesta montagem podemos criar um programa que faça o Arduino mandar ligar a porta 4, fazendo com que o LED acenda. Além disso, poderíamos mandar o Arduino enviar um sinal sonoro pela porta 10 emitindo uma nota musical[10].

O que fazer com isso depende de sua criatividade. Poderia ser uma aplicação que emitisse notas musicais aleatórias a cada 5 segundos, e quando a nota fosse sustenido ou bemol o LED acendesse 3 segundos após a emissão do som; servindo como uma aplicação para treinamento de percepção musical.

Para entender melhor a explicação é fundamental que você faça práticas, com equipamentos reais ou com simulações virtuais. O site **tinkercad.com** é um ótimo local para a realização de simulações, como veremos no decorrer de nosso estudo.

Iniciemos com algo bem simples! Entre no *Modelo – Multímetro entre GND e 5V* no Tinkercad, neste Link:

<https://www.tinkercad.com/things/bjB7yG5cEhJ>

Depois, no modelo, clique abaixo, em *Simular*, e em seguida, acima, em *Iniciar simulação*. Você verá um circuito funcionando como na Figura 1.5, a seguir:

[10] Neste momento um leitor atento pode pensar: “Mas se na porta digital só sai 0 ou 5V, como é que pode sair um sinal de áudio, que é algo analógico?” A resposta é que isso é possível porque existe uma maneira de o Arduino simular uma variação de sinal analógico. Mas este é um assunto mais complexo, desnecessário para este curso introdutório.

Figura 1.5 – Modelo com medição de tensão no Arduino

Veja ainda o *Modelo – Arduino, LED e Resitor*, no Tinkercad. Trata-se da simulação do circuito apresentado anteriormente na Figura 1.4 – Montagem com caixa de som e LED. O link é o seguinte:

<https://www.tinkercad.com/things/eTFyMNhRxcs>

A Figura 1.6 Ilustra este Modelo.

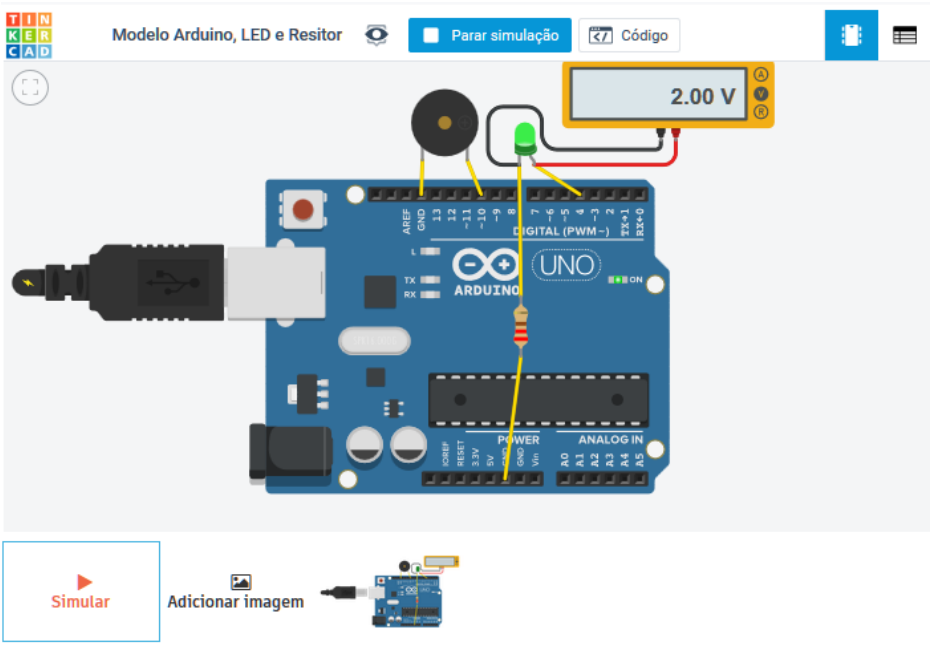


Figura 1.6 – Modelo em simulação

Para reforçar seu aprendizado, assista ao vídeo *Introdução ao hardware: Placas*, disponível no seguinte link:

<https://youtu.be/gTAsGhjsKGk>

1.2 Protoboard

Na Figura 1.4 vimos uma montagem na qual o LED estava conectado diretamente nos terminais (pinos) do Arduino de uma maneira bastante precária, enrolado com a mão. Para evitar isso, na prototipagem experimental o mais usual é fazer as ligações por meio de uma *protoboard*[11] (também conhecida como *breadboard*

[11] O termo protoboard, do inglês é a junção do prefixo grego *proto,* que dá a ideia de ser algo inicial, com *board* que significa placa. Placa é uma alusão à uma placa de circuito no qual os componentes eletrônicos são soldados. No caso da protoboard não é necessário soldar nada. Basta inserir os terminais nos orifícios desejados.

ou placa de ensaio), como a placa de cor branca na Figura 1.7, e cabos com conectores *dupont*, como o verde e o preto, na figura.

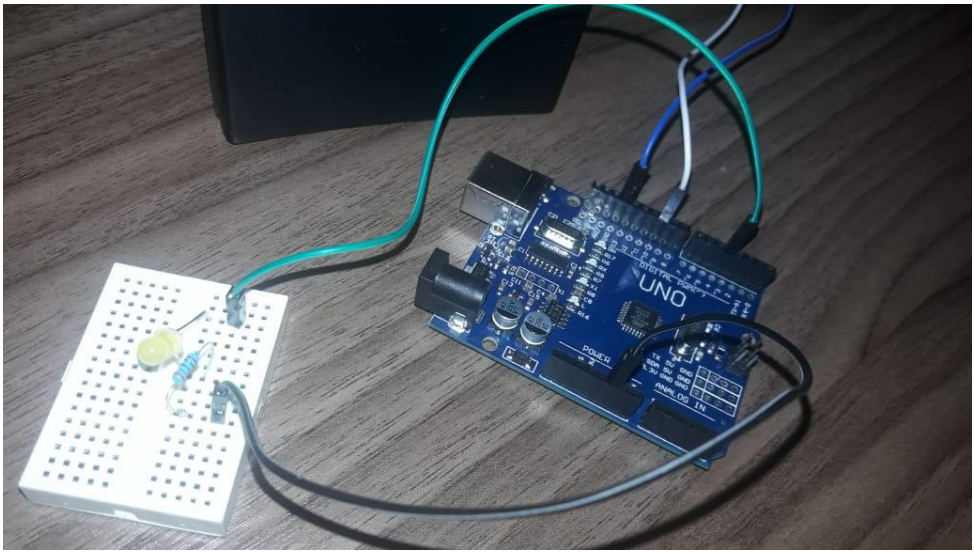


Figura 1.7 – Montagem com protoboard

Na protoboard, os orifícios em linha, próximos, estão interconectados, internamente. A Figura 1.8 apresenta um detalhe do circuito com uma seta dupla vermelha indicando a direção dos orificios eletricamente conectados em uma das metades e em roxo na outra metade da protoboard. Ou seja, vemos que o pino macho[12] do cabo dupont verde está em contato elétrico com um dos terminais do LED, pois está na mesma linha. O outro terminal do LED está conectado a um dos terminais do resistor e, por fim, o outro terminal do resistor está na linha do pino macho do cabo preto.

[12] Os pinos são chamados de macho quando possuem uma ponta que é inserida e, de fêmea quando consistem em um orifício que é um local para a inserção.

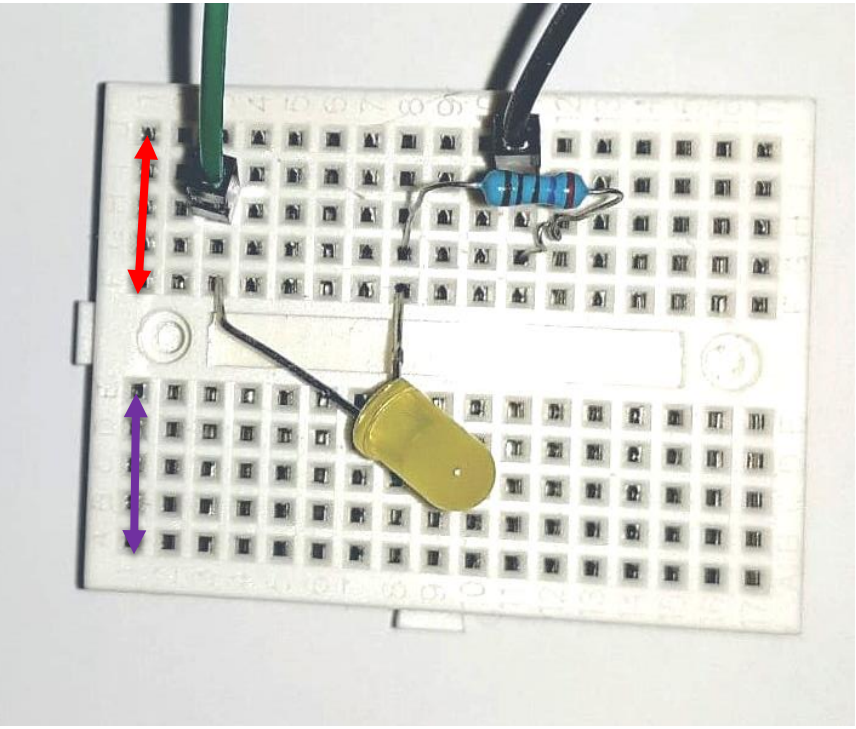


Figura 1.8 – Detalhe da protoboard

Veja o *Modelo – Arduino, LED, Resitor e Protoboard* para ver a simulação com Arduino, LED, Resistor e Protoboard, no seguinte link:

<https://www.tinkercad.com/things/6J7r6UFNCS9>

1.2.1 Prática para entendimento 1

1-Crie uma conta no site https://www.tinkercad.com.

2-No site, clique em *Circuits* e depois em *Criar um novo circuito.*

3-Na parte direita superior da tela, em *Componentes,* escolha a opção *Todos*. Depois, logo abaixo, vá procurando a imagem de uma *placa de ensaio mini* (é uma *protoboard*). Depois de encontrá-la, clique com o mouse e solte-a na parte central da tela. Vide a Figura 1.9 para entender melhor.

Figura 1.9 – Colocando uma Placa de ensaio mini no Tinkercad.

4-Depois, procure por um LED e por uma Pilha de 1.5V e acrescente-os à parte central da tela. Vide a Figura 1.10 para entender melhor.

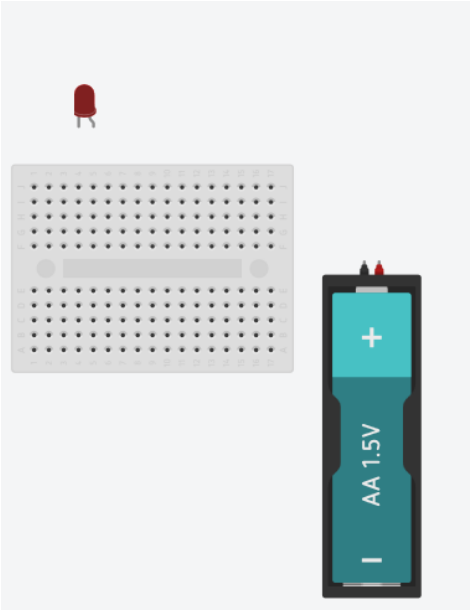


Figura 1.10 – LED, Protoboard e Pilha de 1.5V no Tinkercad

5-Coloque o LED encaixado em dois pinos da Protoboard. É para ficar como na Figura 1.11.

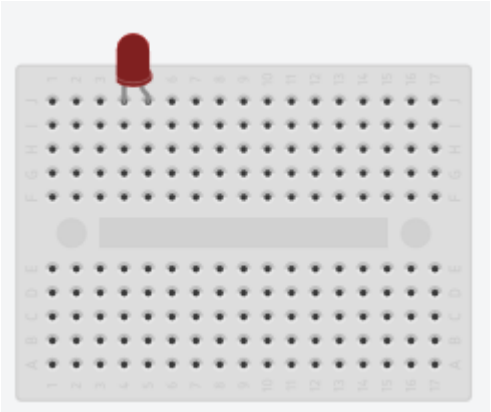


Figura 1.11 – LED na protoboard

6-Em seguida, clique no terminal preto da pilha e puxe um cabo até chegar em um dos pontos da trilha na qual a perna mais curta do LED ficou encaixada. É para ficar como na Figura 1.12.

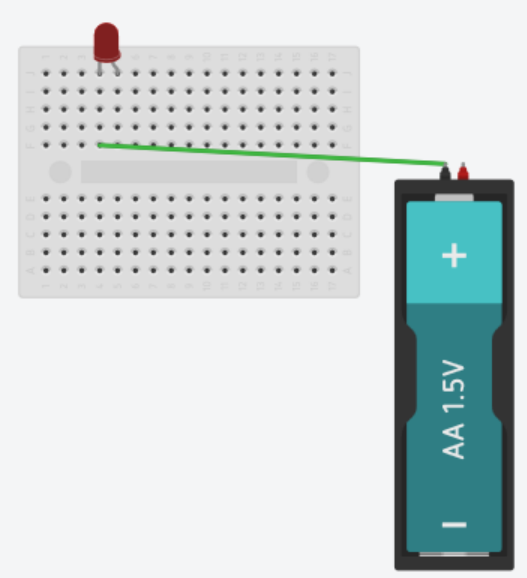


Figura 1.12 – Cabo ligado no negativo da pilha e na perna mais curta do LED

7-Agora, ligue o positivo da pilha (terminal vermelho) na trilha na qual o terminal mais longo do LED está ligado. Ficará como na Figura 1.13.

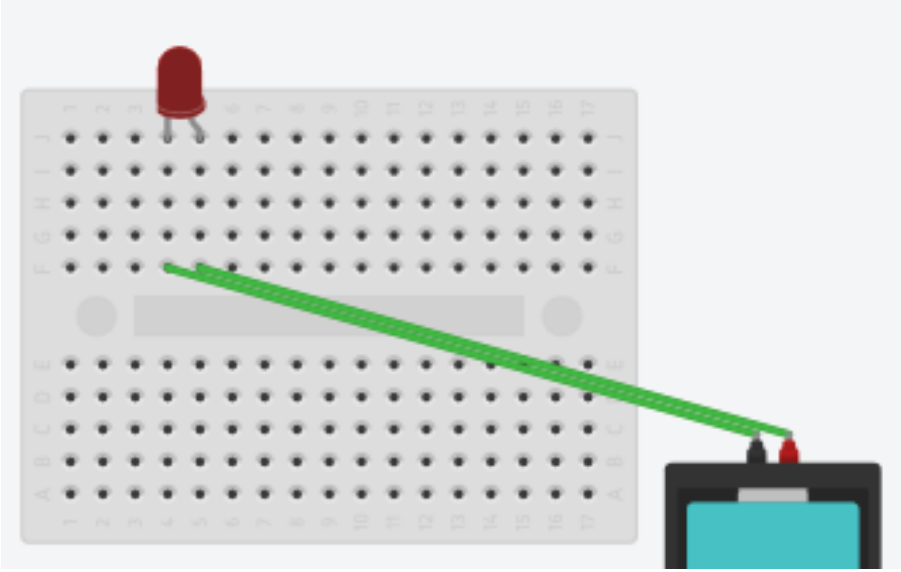


Figura 1.13 – Circuito de pilha no LED montado

8-Agora, você já pode testar seu circuito clicando em *Iniciar simulação*. Se fez tudo certo, o LED irá acender.

9-Aproveite e deixe seu circuito mais bonito e organizado, reposicionando os componentes; e encurvando e colorindo os cabos. Para encurvar um cabo, basta dar um clique duplo no meio dele. Para recolorir um cabo, basta clicar nele e apertar um número no teclado (1, 2, 3, até a tecla 0). A Figura 1.14 ilustra o mesmo circuito, agora mais organizado de se ver.

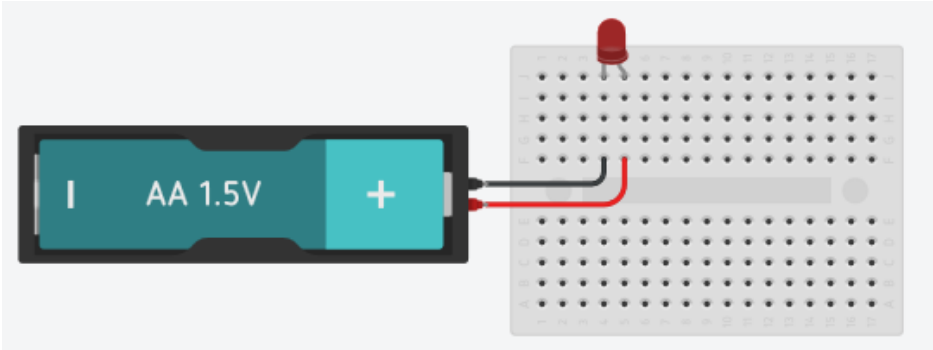


Figura 1.14 – Circuito mais bonito

10-Agora, experimente mudar a localização dos cabos da protoboard para ver o que ocorre na simulação. Ou seja, teste fazendo ligações erradas e corretas. A Figura 1.15 mostra um exemplo de ligação correta e a Figura 1.16, de ligações erradas.

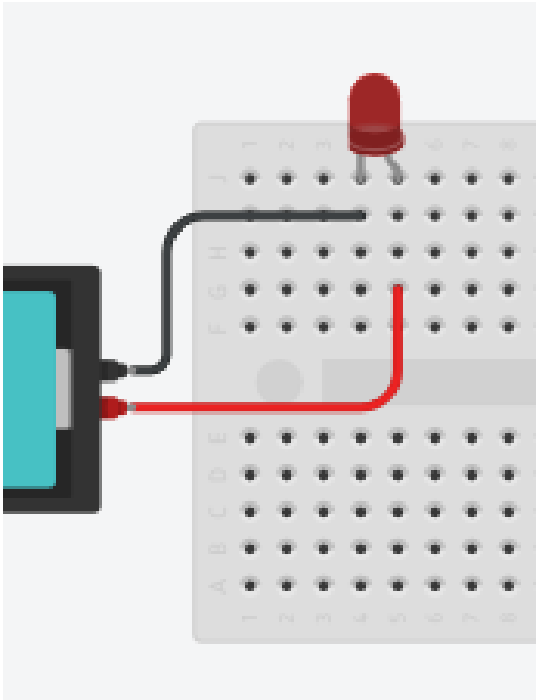


Figura 1.15 – Outra forma de ligar, **correta**. O LED acende

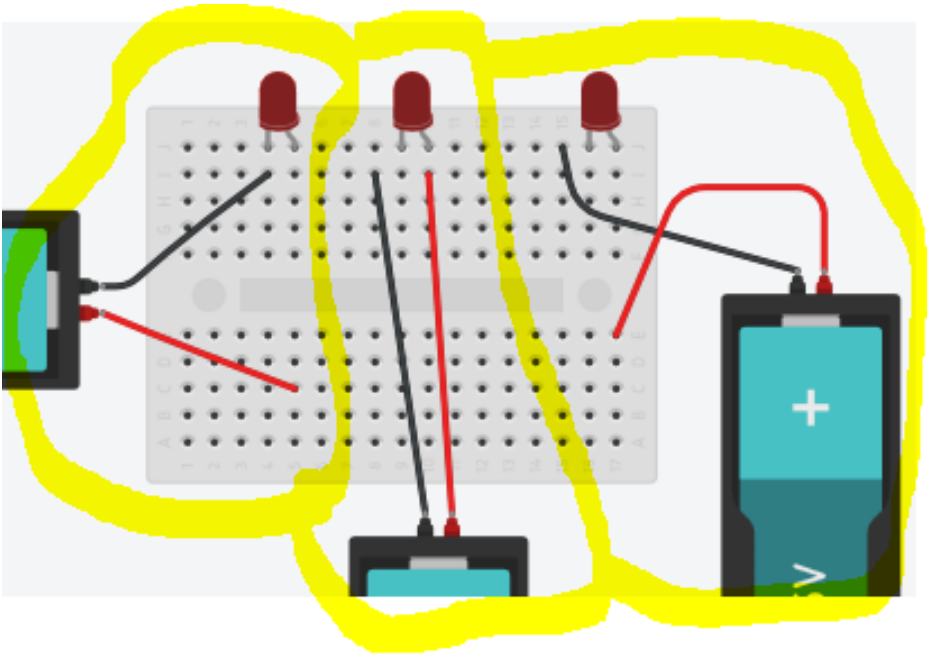


Figura 1.16 – Ligações **erradas**. Os LEDs não acendem

Para reforçar seu aprendizado, assista ao vídeo *Introdução ao hardware: Protoboard*, disponível no seguinte link:

<https://youtu.be/X891q3jWUL8>

1.3 Outros componentes e equipamentos

Até o momento já lhe foram apresentados alguns itens utilizados na eletrônica como placas de Arduino, Protoboard, LED, Resistor, Cabo com conector dupont e Caixa de som. A seguir, temos um maior detalhamento apenas para você ir se familiarizando com os elementos. Não se preocupe em memorizar! Neste capítulo estamos apenas apresentando alguns termos, conceitos e equipamentos para ir quebrando o gelo. Nos demais capítulos tudo será apresentado de forma bastante prática e apenas com os conceitos essenciais para a realização dos experimentos.

A Figura 1.17 apresenta diversos modelos de **protoboard**.

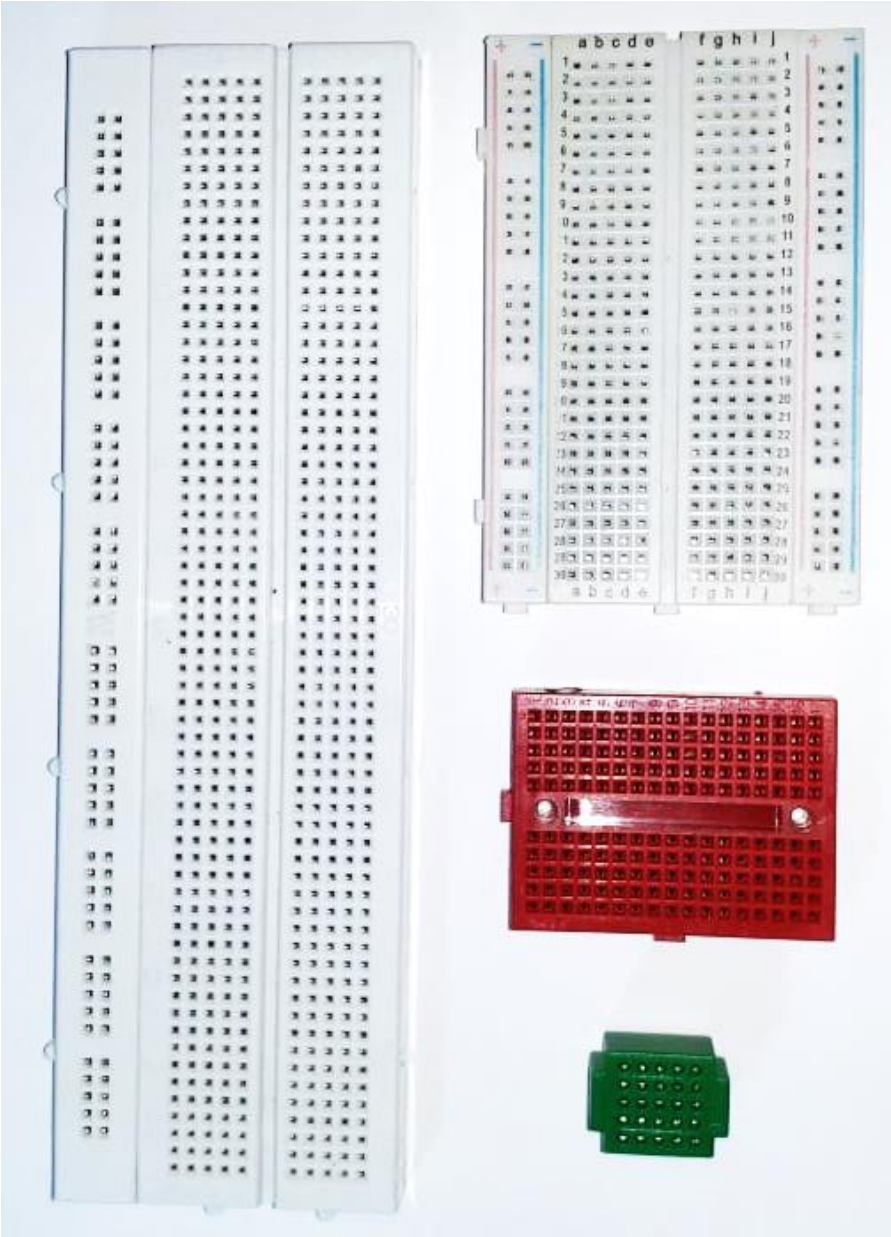


Figura 1.17 – Vários modelos de protoboard (placas de ensaio)

Agora podemos observar, na Figura 1.18, os três tipos de cabos com **conectores dupont**. Na figura, o mais acima é chamado de macho-macho, o do meio é chamado fêmea-fêmea e o de baixo, macho-fêmea. A nomeação macho ou fêmea advem da analogia com sistemas de reprodução biológicos.

Figura 1.18 – Cabos com conectores Dupont machos ou fêmeas

Na Figura 1.19 temos três **LEDs**. Eles são como “pequenas lâmpadas” que emitem luz. Na figura, o mais para a esquerda é um LED de alto brilho (este utilizado para esta foto emite a cor vermelha), os demais LEDs da figura emitem as cores correspondentes. É importantíssimo salientar que o LED possui uma polaridade específica de funcionamento. Assim, se você ligar o positivo no negativo, o LED não ligará ou pode até queimar. Para saber qual é o terminal positivo, basta comparar o tamanho entre eles. O terminal mais comprido é o positivo e o mais curto é o negativo. Confira na figura!

Figura 1.19 – LEDs diversos

Outra atenção que se deve que ter com os LEDs é que eles não podem ser ligados diretamente nas portas do Arduino. Um dos terminais do LED tem que ser ligado a um resistor para evitar que o LED queime.

Por falar em **resistor**, a Figura 1.20 apresenta um deles. Ele serve para limitar a corrente elétrica fazendo com que a tensão diminua nas ligações. O quanto o resistor resiste é dado em ohms (Ω). Por exemplo, um cabo (ou fio[13]) normal seria um "resistor" de 0Ω, uma borracha de apagar seria um "resistor" de infinitos Ω. Um resistor, para não deixar um LED queimar ao ligar em 5V poderia ter cerca de 200 Ω.

Figura 1.20 – Resistor com as cores verde, azul, vermelho e dourado

Para exemplificar como uma ligação errada pode queimar um LED, elaboramos o *Modelo – Como queimar LEDs*, vide no Link: https://www.tinkercad.com/things/hymFzt1iMNZ. Ou, seja, são exemplos do que NÃO fazer.

No resistor, o valor é indicado por um código de cores. Por exemplo, neste resistor da Figura 1.20, temos 4 faixas de cores: verde, azul, vermelha e dourada. Esta marcação indica um resistor de 5600Ω (ou 5,6KΩ ou 5K6). Para saber o significado das cores utilize sites na internet, como o seguinte: <http://www.novaeletronica.com.br/ferramentas_online/cores-de-resistor-online.php>.

Uma dica: Ao comprar resistores, o ideal é o novato em eletrônica simplesmente sempre guardar separado os resistores de mesmo valor em um saquinho com a resistência anotada. Assim não terá dificuldades em saber o seu valor.

[13] Tecnicamente a diferença entre um fio e um cabo é que o cabo tem vários filamentos juntos e o fio tem apenas um filamento.

1.3.1 Prática para entendimento 2

1-Abra o seguinte modelo no Tinkercad, no link:

<https://www.tinkercad.com/things/6DLspI5V8oH>

Ele vai aparecer como na Figura 1.21.

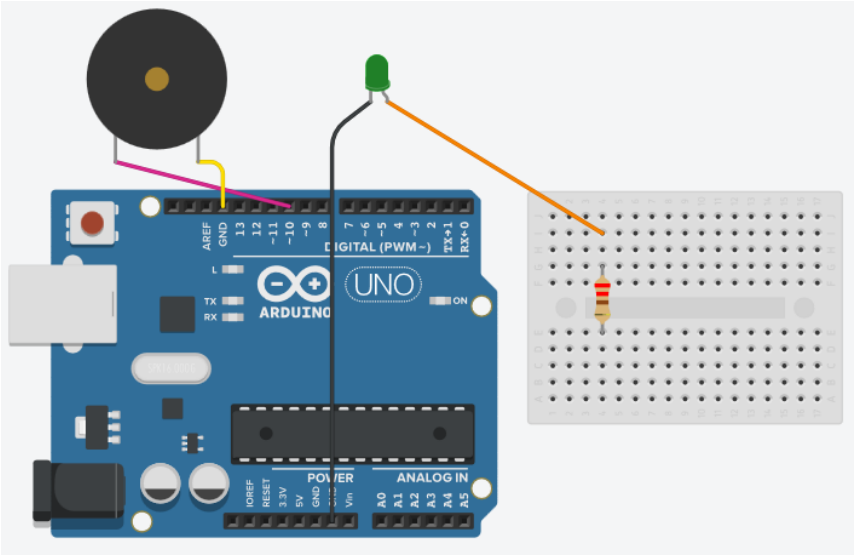


Figura 1.21 – Toca lá-ré-mi e acende LED no lá, incompleto

Para este modelo, foi elaborada uma programação no Arduino na qual ele vai tocar as notas lá, ré e mi, aleatoriamente, e sempre que tocar o lá ele vai acender o LED, pela porta 7 do Arduino. Ou seja, o programador resolveu usar a porta 7, ou seja, mandar 5V pela porta 7. O problema no circuito é que falta ligar o cabo entre a porta 7 do Arduino e o terminal mais comprido do LED. Só que, se isso for feito direto, o LED vai queimar. Por isso, foi colocado um resistor, já com a resistência certa para este caso (220ohms). Mas, o resistor não está completamente ligado.

2-Entre o modelo e clique em *Iniciar simulação* para perceber o que ocorre. Observe que as notas musicais são executadas, mas o LED não acende, nunca. Pois ele **não** está ligado na porta 7. Então, pare a simulação e faça a ligação correta e veja se funciona o LED. É um desafio! Boa sorte!

A Figura 1.22 apresenta, da esquerda para a direita, um **piezo elétrico**, um **autofalante** e um **microfone**. Eles são transdutores eletroacústicos, ou seja, componentes que transformam energia elétrica (áudio) em energia mecânica (som). No Arduino é recomendável se ligar um resistor entre uma porta do Arduino e um terminal do autofalante, isso para evitar que o Arduino queime.

Figura 1.22 – Alguns transdutores

Para a **alimentação do Arduino** temos várias possibilidades. Na Figura 1.23 vemos algumas delas: um cabo USB, um soquete com uma bateria de 9V e uma fonte de alimentação de 6 a 12V DC.

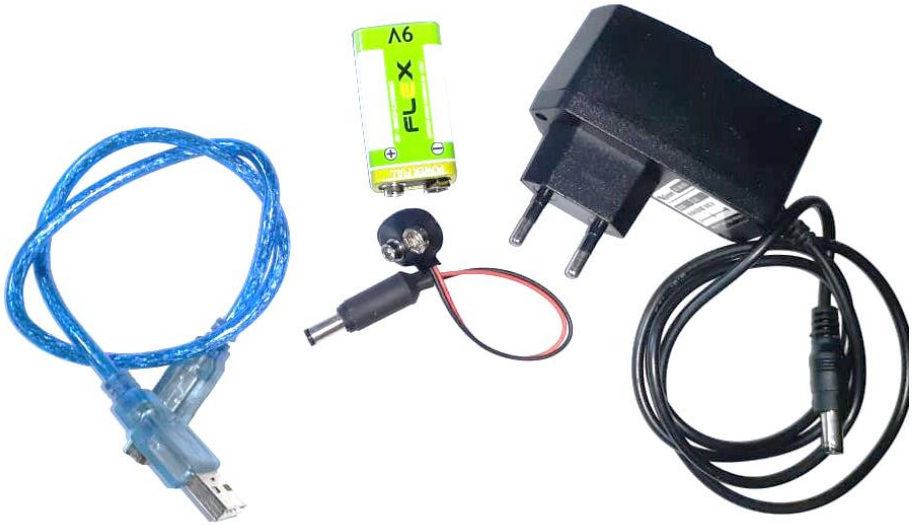


Figura 1.23 – Alimentação para o Arduino

No jargão da tecnologia utiliza-se os termos **sensores** e **atuadores**. Os sensores recebem dados ou ações do meio físico e os atuadores emitem dados ou ações ao meio físico. Em uma comparação com o corpo humano, nossos sensores seriam olhos, ouvidos papilas gustativas etc., e nossos atuadores seriam braços, pernas, pregas vocais etc. Assim, um LED e um autofalante são exemplos de atuadores, enquanto um microfone é um exemplo de sensor. Já o piezo elétrico pode ser tanto um sensor quanto um atuador, dependendo de como é ligado. A Figura 1.24 apresenta alguns botões de apertar, ou seja, são sensores, pois transformam a ação mecânica em uma sensibilidade elétrica.

Figura 1.24 – Botões (*push-buttons*) diversos

A Figura 1.25 mostra dois **potenciômetros**, que são resistores variáveis. Ou seja, são utilizados como sensores. Você pode fazer um programa no Arduino para modificar a frequência de uma nota musical conforme gira ou desliza o potenciômetro, por exemplo.

Figura 1.25 – Potenciômetro deslizante e potenciômetro giratório

Um sensor manual também bastante interessante e apresentado na Figura 1.26, que é um **Joystick**. Na verdade, este joystick é um conjunto de dois potenciômetros (para os controles laterais) e um botão (para quando se aperta para baixo o controle).

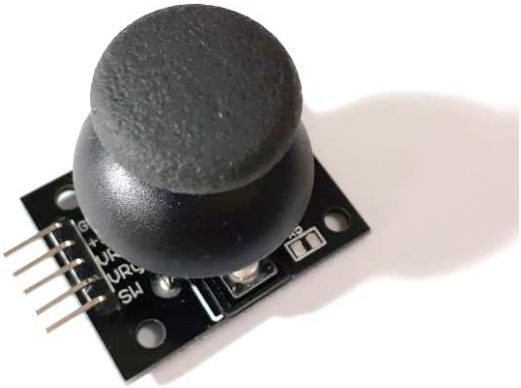


Figura 1.26 – Joystick

Como exemplos de atuadores temos o **mini motor elétrico** da Figura 1.27 e o **servo motor** na Figura 1.28.

Figura 1.27 – mini motor elétrico

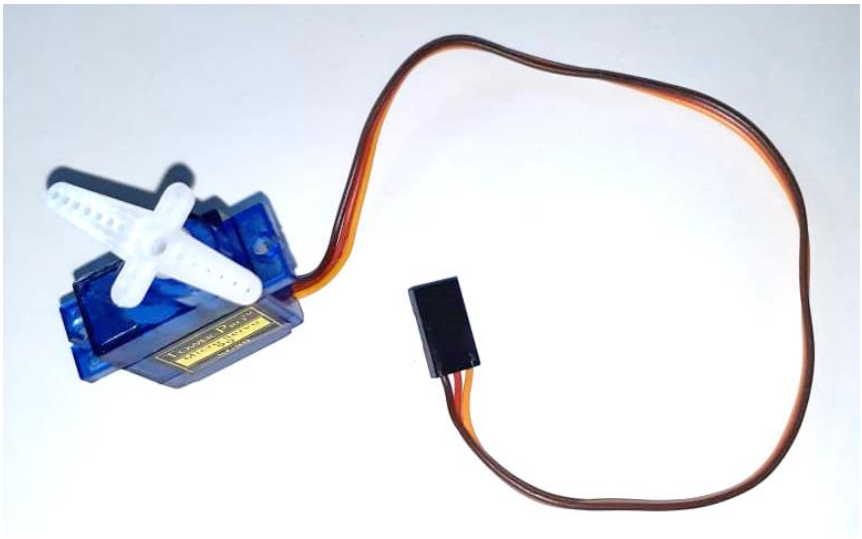


Figura 1.28 – Servo motor

Por fim, na Figura 1.29 podemos ver **diversos** sensores, atuadores e módulos para o Arduino. Incluem displays, encoder rotativo, sensor de pulso, módulo para cartão SD, sensor de posição, relê, controle remoto, sensor de linha, sensor de toque, módulo Wifi e módulo bluetooth. Não se preocupe pois neste livro não iremos aprender a utilizar estes sensores. O objetivo aqui é apenas mostrar todo o potencial de aplicações com o Arduino para lhe incentivar a aprendizagens futuras.

Figura 1.29 – Diversos sensores, atuadores e módulos para o Arduino

Para reforçar seu aprendizado, assista ao vídeo *Introdução ao Hardware: Outros componentes e equipamentos,* no seguinte link:

<https://youtu.be/gTAsGhjsKGk>

1.4 Noções super elementares sobre eletrônica

Para evitar dissabores com a queima de equipamentos, iremos indicar aqui alguns rudimentos super elementares sobre eletricidade e eletrônica. Leia com atenção!

1.4.1 Condutores e isolantes

Os elementos da natureza são formados por átomos. Alguns dos elementos possuem elétrons bastante livres que podem sair de um átomo e ir para outros átomos. Estes elementos são chamados de condutores elétricos. Em outros elementos a passagem de elétrons é mais difícil, eles são chamados de isolantes elétricos. São exemplos de condutores os metais em geral e algumas soluções salinas. São exemplos de isolantes: borracha, cerâmica, plástico, vidro e o ar.[14] Para entender melhor, assista o seguinte vídeo intitulado *Condutores e Isolantes,* com uma simulação:

<https://youtu.be/1SeRJXolTqg>

1.4.2 Corrente elétrica

De forma simplificada, podemos dizer que, para que ocorra uma **corrente elétrica** em um componente, são necessários **dois "fios"** a serem ligados a uma fonte de energia elétrica. Um, de onde vêm os elétrons e outro, para onde vão. Você pode notar isso nos equipamentos em geral que possuem tomadas com dois pinos. Vide a simulação no *Modelo – Dois fios para corrente elétrica*, no link:

[14] Observe que, na verdade, não existe isolante perfeito, porquanto, dependendo das condições, sempre alguma condução de eletricidade pode existir. Por exemplo, o ar é "isolante" mas não o suficiente para evitar um raio.

<https://www.tinkercad.com/things/dVTVN3OYnsL>

A Figura 1.30 apresenta o aspecto geral da simulação indicada.

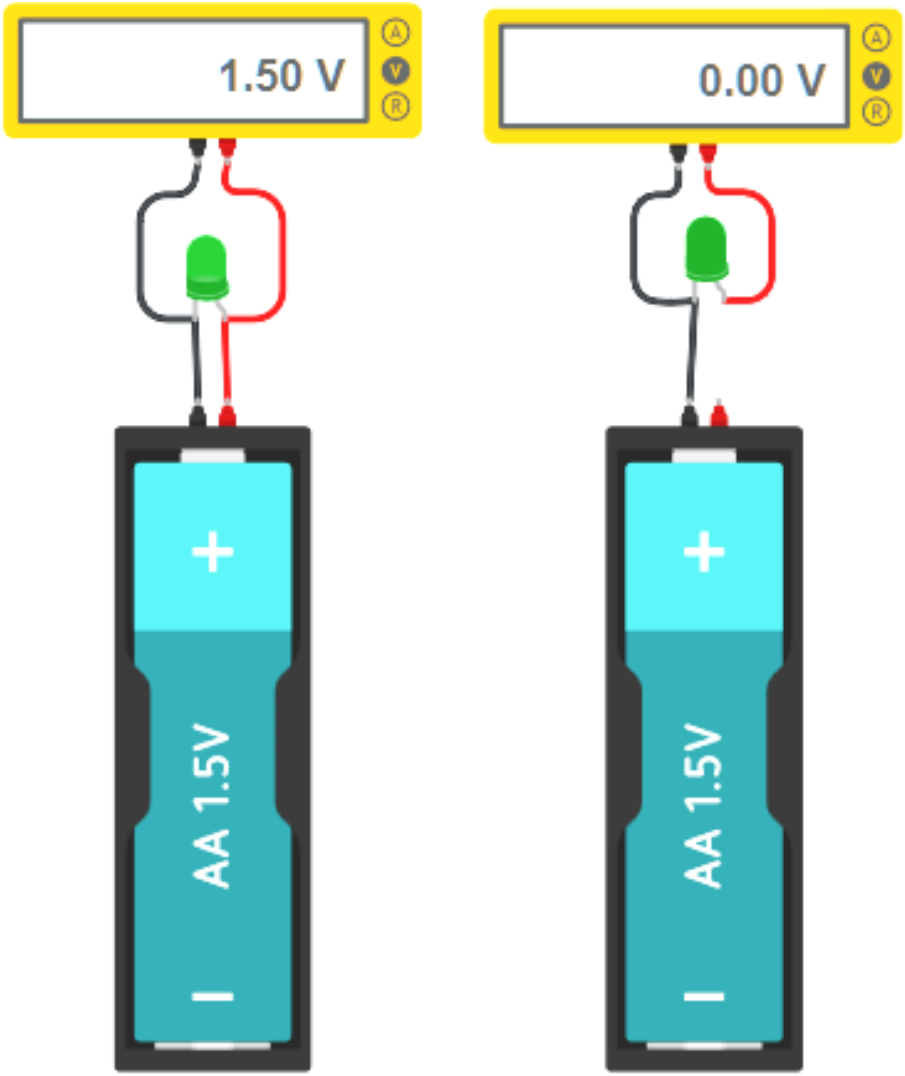


Figura 1.30 – Circuito fechado e circuito aberto

A questão é que a corrente elétrica é um movimento de elétrons, sempre de onde tem mais elétrons para onde tem menos elétrons; já que na natureza as coisas buscam o equilíbrio. Por exemplo, se você estiver descalço e pegar em um dos fios desencapados de uma tomada de sua casa irá levar um choque[15]. Isso, porque no fio que você pegou existia uma quantidade de elétrons diferente da quantidade de elétrons da terra. Enquanto você não pegava no fio desencapado nada ocorria, depois que você pegou, ocorreu a corrente elétrica. Trata-se, então, de uma máquina de choque elétrico. NÃO recomendamos que faça esta experiencia, pois seria dolorosa, senão, fatal. Para entender melhor, assista ao seguinte vídeo intitulado *Circuito Elétrico,* com uma simulação:

<https://youtu.be/f4l-URP0wal>

Existem duas possibilidades de configuração da eletricidade nos fios. Uma é chamada de **corrente alternada** (AC, do inglês *alternating current*), que é a que tem na parede de sua casa etc. Na corrente alternada, cada pino da tomada fica variando a quantidade de elétrons 60 vezes por segundo, como é o padrão aqui no Brasil. A outra possibilidade de corrente é chamada de **corrente contínua** (DC, do inglês *direct current*) que é a que tem em uma pilha ou bateria, na saída de seu carregador de celular ou no Arduino.

Em **eletrônica normalmente se usa corrente contínua** (DC). Ou seja, existe um terminal negativo (-), com mais elétrons e um positivo (+), com menos elétrons. Assim,

- uma pilha tem terminais + e -,
- uma saída de carregador de celular tem + e -,
- uma bateria de carro tem + e -,
- um cabo USB de computador tem + e -.

[15] A bem da verdade, na tomada de 127 volts padrão de uma residência, apenas um dos dois fios dá choque. O outro é o **terra**. Assim, você tem 50% de chance de ser eletrocutado, neste caso.

Assim que algum componente elétrico for ligado entre estes terminais, iniciará uma corrente elétrica. Só lembre de nunca ligar o positivo no negativo diretamente pois isso gerará um chamado **curto circuito**, que pode gerar incêndios e queima dos equipamentos.

O **carregador do celular** transforma a corrente alternada (da tomada) em contínua, pois o celular utiliza componentes eletrônicos. Veja a Figura 1.31 com um carregador de celular indicando que na entrada é *AC*, ou seja corrente alternada, de *100* a *240* volts, e de *50* a *60* vezes por segundo. Veja ainda que na saída ele indica *DC, 5* volts.

Figura 1.31 – Carregador de celular. Fonte: https://novaknup.com.br/produtos/carregador-de-celular-3-1a-kp-ic005a/

Em relação ao risco de choque elétrico, a DC é melhor pois nela a voltagem tende a ser bem menor. Ou seja, se você pegar em um fio na saída DC de seu carregador de celular, mesmo

descalço, não sentirá o choque que levará, pois, 5 volts não é percebido pelo seu corpo, em geral. Só não coloque na língua pois aí, sim, perceberá o choque.

Nos circuitos de corrente contínua (DC), o **negativo é chamado de GND** (do inglês *ground*), ou terra. A ideia é que no positivo existe uma menor quantidade de elétrons (que possuem carga negativa). Assim, para compensar, os elétrons vão do negativo para o positivo. Esta diferença resulta no que chamamos de tensão, ou popularmente, voltagem (medida em volts, V).

Embora, como vimos, os elétrons caminhem do negativo para o positivo, **na prática dizemos que o sinal elétrico vai do positivo para o negativo**[16]. Ou seja do + para o -. O nome disso é sentido de corrente *convencional*. É mais prático pensar assim! Para entender melhor, assista ao seguinte vídeo intitulado *Corrente Contínua X Alternada,* com uma simulação:

<https://youtu.be/P0F1YsexIxM>

Alguns componentes eletrônicos possuem uma polaridade. Ou seja, devem ser ligados corretamente para funcionarem e não queimarem. Ou seja, deve-se ligar corretamente o positivo no positivo e negativo no GND, conforme as indicações em um circuito. Este é o caso do LED. Existem outros componentes que não possuem polaridade, como um resistor ou um autofalante. Nestes casos, tanto faz o terminal ligado.

1.4.3 Alimentação e tensões elétricas no Arduino

O Arduino utiliza 5V para poder funcionar. A porta **USB** do computador fornece 5V. Um carregador de celular também fornece 5V. Porém, existem outras fontes que possuem voltagem diferente de 5V e você deve estar sempre atento a isso para evitar que o Arduino queime.

[16] O motivo disto é que no passado a teoria da eletricidade ainda não estava madura o suficiente. Então pensavam que a corrente ia, realmente, do positivo para o negativo.

Existem várias formas de se alimentar o Arduino. Uma delas é utilizando o próprio **cabo USB** que fornece os 5V utilizados pelo Arduino. Esta já é a voltagem operacional do Arduino. Outra forma é com uma fonte de alimentação conectada na **entrada de alimentação do Arduino**. Neste caso, pode-se ligar uma fonte DC, com tensão entre 5V e 12V, uma vez que, nesta entrada, existe um circuito que faz a conversão para 5V.

No *Arduino Uno* temos 3 pinos com a indicação **GND**. Ou seja, é o **terra**, o **negativo**, o **-** . Os 3 pinos são exatamente iguais, ou seja, tanto faz em qual você ligar, é a mesma coisa.

No *Arduino Uno* temos um **pino** chamado **5V**. Ele é o sinal positivo de 5V. Muito **cuidado** em onde você ligá-lo. Por exemplo, se você conectar um cabo entre o pino 5V e o GND você irá QUEIMAR o Arduino. Então terá que jogar fora o Arduino e arcar com o prejuízo.

Os **pinos de 0 a 13** são pinos de entrada ou de saída. Ou seja, sua função varia conforme o programa. Da mesma forma que no pino 5V, nunca ligue diretamente um dos pinos de 0 a 13 no GND pois poderá queimar o Arduino já que estes pinos também podem emitir 5V. A diferença é que o pino 5V emite esta tensão o tempo todo, enquanto os outros apenas quando o programa mandar. Confira na Figura 1.32 essas indicações no Arduino Uno:

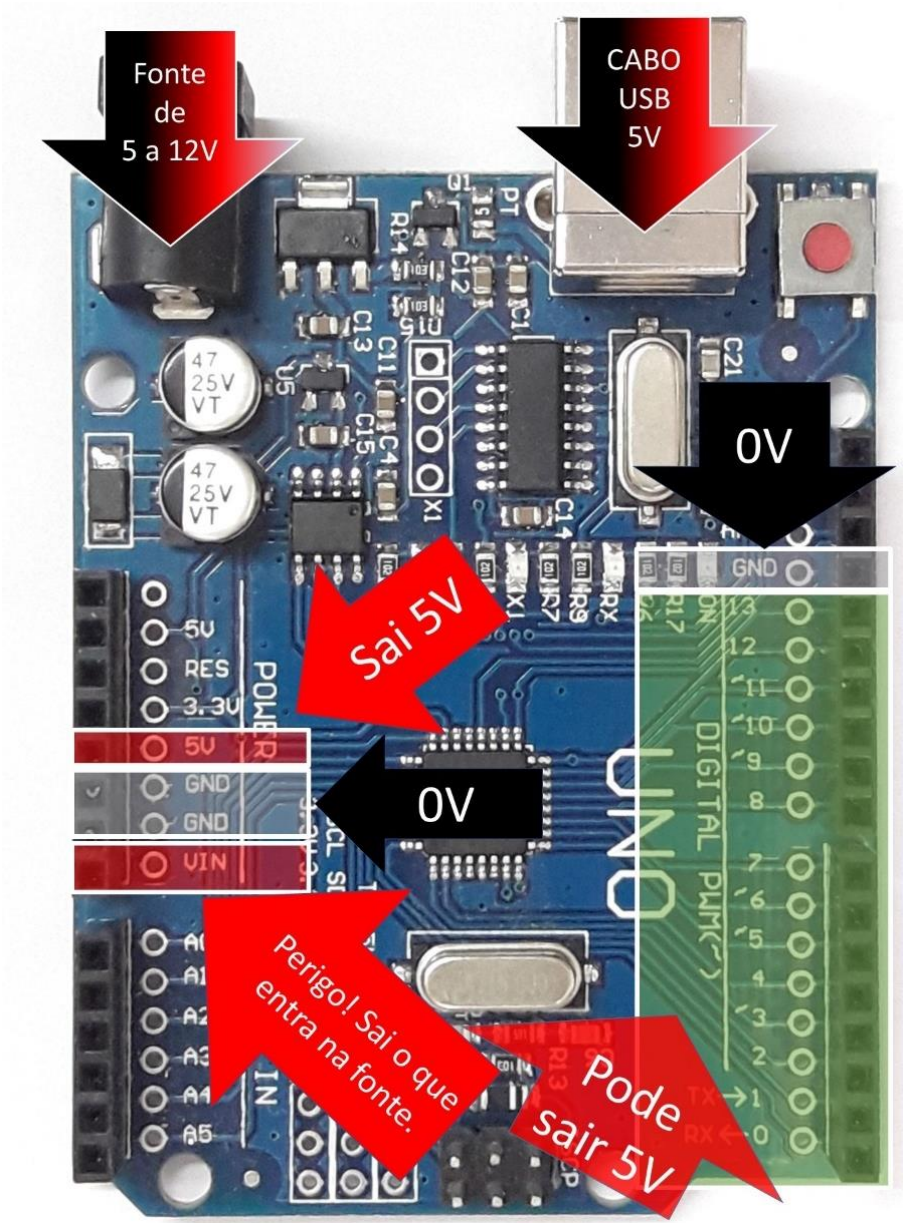


Figura 1.32 – Tensões no Arduino Uno

Para entender melhor, verifique, a seguir, o *Modelo – Erros de curtos, tensões erradas e polaridades trocadas* no Link:

<https://www.tinkercad.com/things/91GvZleRdp3>

Na seção anterior foi comentado sobre os termos *digital* e *analógico*. Então é importante fazer uma breve explanação, como segue:

Na área da tecnologia, **digital** significa um sistema que funciona apenas com dois estados[17], por exemplo:

- Ligado ou desligado
- 1 ou 0
- 5V ou 0V

Já na tecnologia **analógica,** tem-se um sistema com uma variação de possibilidades dentro de um máximo e um mínimo suportado pelo equipamento. A Figura 1.33 ilustra uma comparação do sinal digital com o sinal analógico considerando um sistema que trabalhe entre 0 e 5 volts.

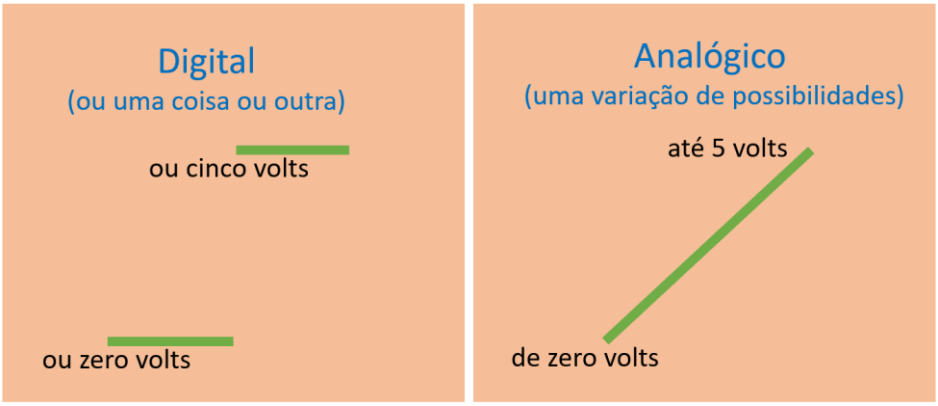


Figura 1.33 – Digital x Analógico

[17] Isso é porque no digital é utilizado um sistema binário. Ou seja, apenas com zeros e uns.

Por exemplo: Considerando a figura anterior, seria possível, no sistema digital, eu considerar 2,37V? Não, pois no sistema digital ou é 0 ou é 5 volts. E se fosse no sistema analógico, eu poderia ter 2,37V? Sim, pois pode existir qualquer valor dentro de um máximo e um mínimo suportado.

Para o iniciante no Arduino basta pensar o seguinte: Usa-se a ideia de digital para indicar algo ligado ou algo desligado; e usa-se a ideia de analógico para indicar uma variação de possibilidades.

1.5 Materiais e componentes para montagens práticas

Para entender o funcionamento do que estamos indicando neste livro não existe problema em utilizar-se do simulador no Tinkercad, porém, se você deseja, de fato, adentrar nessa área e poder produzir estes aparatos eletrônicos, terá que adquirir algumas coisas. Você pode comprar estes elementos em lojas de equipamentos eletrônicos em sua cidade ou em sites pela internet. A seguir, passo uma lista com os preços aproximados, em reais, que você pode adquirir no Brasil, para realizar os experimentos que se seguem neste livro e poder praticar um pouco mais, conforme sua criatividade. Obs.: Se importar da China, o valor pode cair para mais da metade.

Quadro 1.1 – Listagem de componentes básicos para a elaboração dos aparatos deste livro

Quantidade e descrição	Como pode vir escrito no site	Valor total (R$)
1 Arduino Uno R3 ou compatível, com cabo UBS	Arduino Uno R3 - Compatível + Cabo USB 2.0 - A-B	60,00
10 resistores de 150 ohms	Resistor 150R 5% (1/4W)	1,00
10 resistores de 220 ohms	Resistor 220R 5% (1/4W)	1,00
10 resistores de 330 ohms	Resistor 330R 5% (1/4W)	1,00
10 resistores de 1Kilo ohms	Resistor 1K 5% (1/4W)	1,00
1 Protoboard mini	Protoboard 170 pontos	5,00
1 Conjunto com 40 cabos com conectores macho-macho.	Jumper Premium 40p x 20cm - Macho / Macho	9,00

3 LEDs vermelhos	LED Difuso 5mm Vermelho	1,50
3 LEDs amarelos	LED Difuso 5mm Amarelo	1,50
3 LEDs verdes	LED Difuso 5mm Verde	1,50
1 botão de apertar de cor verde	Chave Push button verde	5,00
1 botão de apertar de cor vermelha	Chave Push button vermelho	5,00
2 metros de cabos flexíveis	Cabo Flexível 26 AWG (0,14mm). Ou compre um metro de cabo de rede e retire os cabos internos.	1,00
1 potenciômetro linear	Potenciômetro Linear de 1K (1000Ω) ou outro valor qualquer.	2,00
1 Autofalante pequeno de 8 ou 4 ohms, usado.	É melhor procurar como sucata de eletrônica em caixas de som de computador etc.	0,00
	Total final (R$):	**95,50**

Se você desejar que suas conexões elétricas fiquem mais firmes, indica-se que adquira um Ferro de solda de 20W e Solda (Solda Estanho, Tubinho, 22g). Mas, para os experimentos deste livro, a soldagem não é essencial. Se desejar, pode adquirir ainda um multímetro, alicates etc.

1.6 Resumo

Neste capítulo entendemos sobre o hardware do Arduino, incluindo os diversos tipos de placas e tivemos uma visão geral da placa *Arduino Uno*. Também entramos em contato com o simulador *Tinkercad*. Vimos ainda sobre a *protoboard*, como um elemento para facilitar as ligações na prototipagem eletrônica. Conhecemos os conectores Dupont e a existência de outros componentes e equipamentos como o resistor, o LED, o autofalante, botões e fontes de alimentação para o Arduino. Aprendemos um pouco sobre a corrente elétrica, a polaridade de alguns componentes e as tensões existentes nas conexões do Arduino Uno. Ao final, listamos os materiais necessários para a realização de experimentos físicos, reais.

2. Tocador de duas notas

Agora que já entendeu algumas coisas básicas gerais, podemos iniciar com uma aplicação prática que é um gerador sonoro de duas notas musicais. Trata-se de uma montagem com o Arduino para tocar uma sequência de duas notas em loop infinito.

Assista ao vídeo *Tocador de duas notas executando* no seguinte link para entender melhor a proposta. O link é o seguinte:

<https://youtu.be/haGovAilgLA>

Para a elaboração deste dispositivo, temos que trabalhar em duas frentes: o hardware e o software. Iniciemos com a primeira!

2.1 Separe os materiais

Você pode realizar o experimento de forma real (utilizando componentes reais) ou utilizando um simulador virtual, on-line no site https://www.tinkercad.com. Neste caso, entre em **Circuits** -> **Criar novo circuito**. Então vá em **componentes** – **Todos** e acrescente os seguintes, da Figura 2.1:

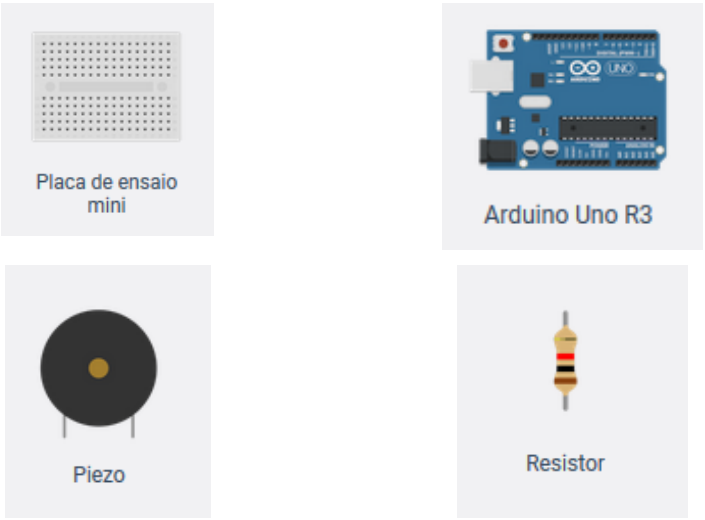


Figura 2.1 – Componentes no TinkerCad para montar o circuito virtual.

Se for fazer uma montagem real, utilize um autofalante, um resistor, uma placa de ensaio e um cabo com conector Dupont macho-macho, como ilustra a Figura 2.2.

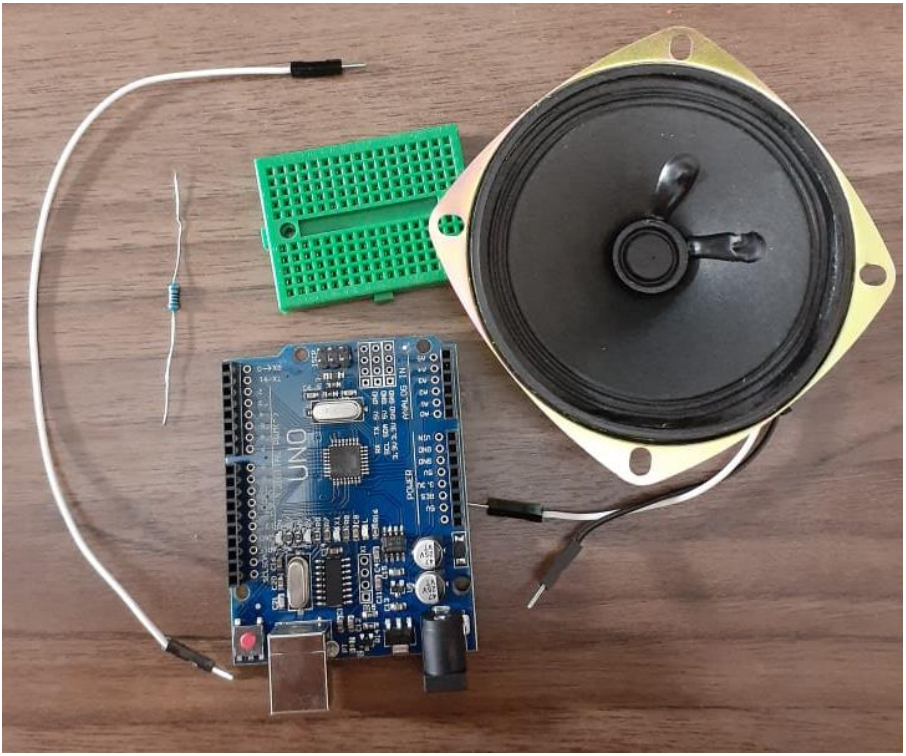


Figura 2.2 – Materiais para uma montagem real

Para os valores do resistor, experimente valores entre 150 e 270 ohms. O objetivo deste resistor é não permitir que a corrente no autofalante seja muito elevada, o que poderia resultar em danificação (queima) do Arduino. Isso, mesmo. Aqui o risco não é queimar o autofalante e sim, o Arduino. Ao colocar o resistor, além de diminuir a corrente, o volume do som irá também diminuir. Colocando resistores maiores, é possível baixar ainda mais volume do som. Experimente utilizar resistências maiores, com até 1000 ohms, para ouvir o resultado.

2.2 Monte o circuito

Para o circuito, conecte um dos terminais do autofalante no *GND* do Arduino. Além disso, conecte o outro terminal do autofalante em um dos terminais do resistor. E por fim, conecte este terminal restante do resistor ao pino *10* do Arduino. Veja as Figuras 2.3 e 2.4 com a montagem no Tinkercad e a montagem real, respectivamente.

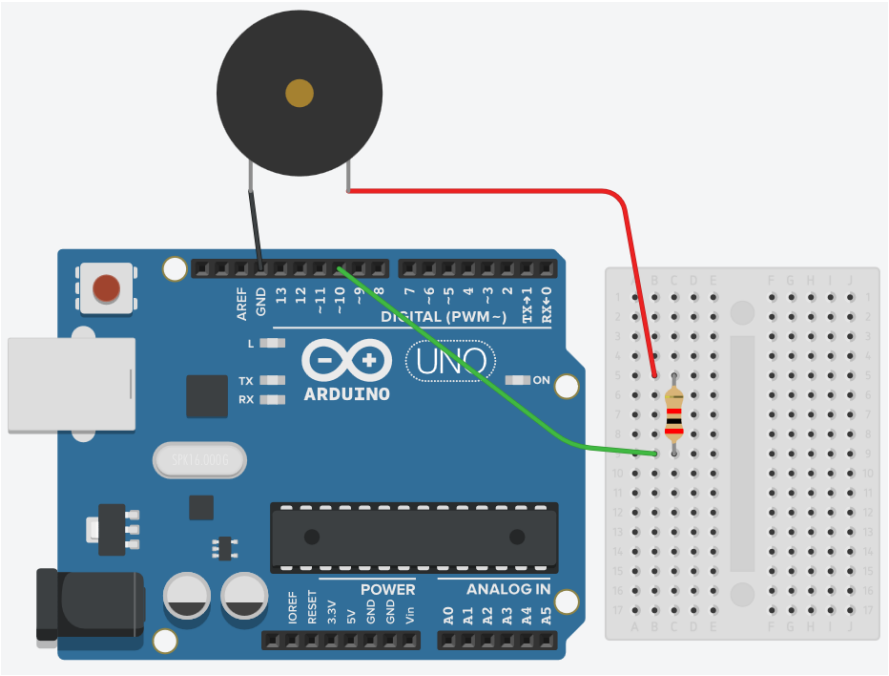


Figura 2.3 – Circuito em uma montagem simulada

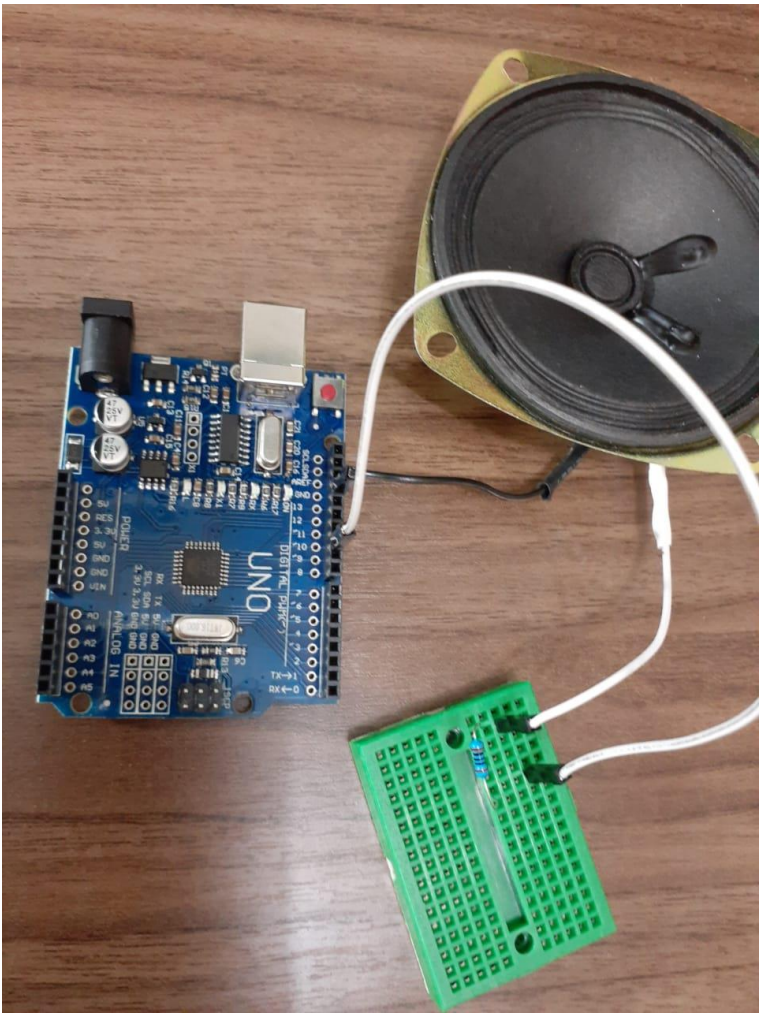


Figura 2.4 – Circuito em uma montagem real

Para entender melhor a montagem no Tinkercad, assista ao vídeo *Escolhendo e montando o hardware para o tocador no Tinkercad* disponível no seguinte link:

<https://youtu.be/gvbm2VKtbnw>

Para ver a montagem real, assista ao vídeo *Montando o hardware para o tocador* disponível no seguinte link:

<https://youtu.be/jKXrDPeE7hg>

2.3 Escreva o código (Sketch)

Se você está fazendo a montagem virtual no https://www.tinkercad.com, clique em **Código,** depois (onde aparece *Blocos*) escolha a opção **Texto** e digite o código a seguir, em cor azul. Tem que ficar igualzinho, senão, não funciona.

Se você está fazendo a montagem real, inicialmente você deve baixar, gratuitamente, o *Ambiente Integrado de Desenvolvimento* (no inglês *Integrated development Environment* - IDE) do Arduino no site oficial, cujo link é: https://www.arduino.cc.

Após instalar, entre no programa de IDE do Arduino em seu computador e crie um novo **Arquivo** e digite o código a seguir. Copie com muito cuidado! Depois, salve o arquivo.

```
//Meu primeiro sketch em linguagem C, para Arduino.
void setup()
{
}
void loop()
{
tone (10, 131, 500);
delay (1000);
tone (10, 196, 500);
delay (500);
}
```

Na IDE do Arduino, o código, também conhecido como *sketch*, irá ficar como está na Figura 2.5.

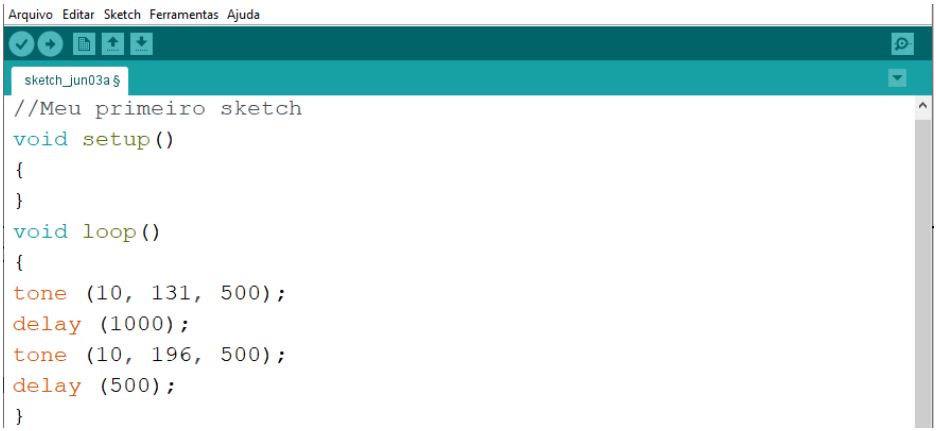


Figura 2.5 – Sketch (código) na IDE do Arduino

2.3.1 Entendendo o código

Para entender o código, considere a Figura 2.6. Leia com atenção!

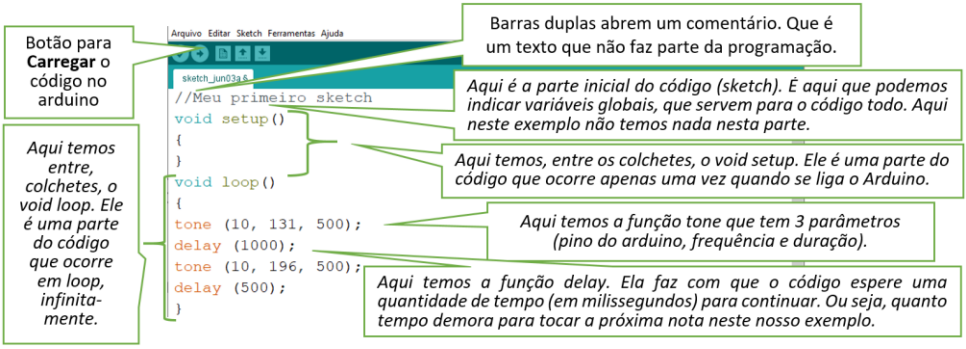


Figura 2.6 – Sketch comentado

Entenda o que o código faz:

a) O código sempre é lido de **cima para baixo**. Isso ocorre muito rápido, de forma que, para a percepção humana, a execução das linhas seria algo **simultâneo**. Mas, na verdade, é uma linha por vez.
b) Então, a primeira coisa que vemos é ***//*** (**comentário**) ou seja, ele vai ignorar tudo o que estiver à direita, na linha.

c) Depois, ele vai descendo até chegar no **void setup** e irá executar tudo o que estiver entre os colchetes. Ou seja, aqui não vai fazer nada, pois está sem nada escrito.
d) Depois, ele vai para o **void loop** e executa, infinitamente, o que estiver entre colchetes.
e) No *void loop*, inicialmente ele executa a função **tone**, mandando para o pino 10 do Arduino um sinal de áudio de 131 hertz que terá a duração máxima de 500 milissegundos.
f) Ai, antes de ir para a outra linha, ele faz um repouso (delay) de 1000 milissegundos.
g) E assim por diante. Ou seja, mais uma linha de comando com **tone** e outra com **delay**.
h) Então, volta para o início de **void loop** e o fica repetindo infinitamente.

2.3.2 Aspectos musicais considerados

Uma partitura com o resultado musical do código seria a seguinte (Figura 2.7):

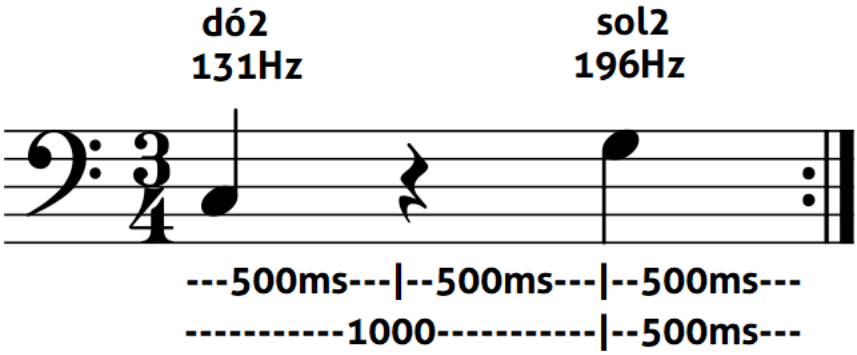


Figura 2.7 – Partitura da peça musical executada no andamento ♩=120

Em termos práticos, na execução, temos a emissão da nota dó2, com frequência 131 (Hz) e com uma duração de 500 ms. Então, 1000ms depois no início da linha anterior, inicia uma outra nota. A nota sol2 (196Hz). Essa outra nota também tem a duração de 500ms. Então, tudo repete.

Entenda que as frequências se referem a notas musicais. Para ficar mais fácil, apresentamos aqui algumas delas entre o dó2 e o dó3:

- Dó *261*
- Réb *277*
- Ré *293*
- Mib *311*
- Mi *330*
- Fá *349*
- Fá# *370*
- Sol *392*
- Láb *415*
- Lá *440*
- Sib *466*
- Si *494*
- Dó' *523*

Para ver uma tabela com os valores das frequências das notas musicais em outras oitavas procure no apêndice deste livro ou acesse o link:

<https://sites.google.com/site/aprendamusicacomigo/musica-e-tecnologia>

No site indicado anteriormente, a frequência está bastante precisa, com 4 casas decimais. Para você, basta pegar os 3 primeiros números para nosso experimento. Fica mais simples!

Obs.: Existe uma discrepância de 1 oitava entre o sistema internacional e o sistema de oitavas utilizado no Brasil (que é o sistema franco-belga). Para nós, o dó central é o dó3. No sistema internacional o dó central é chamado de dó4.

Sobre as durações, entenda que a duração (os milissegundos) de cada figura musical varia conforme o andamento. O andamento resultante do exemplo é 120 tempos por minuto (♩=120), pois cada tempo possui 0,5 segundos (500 milissegundos).

Então, você deve considerar que para cada andamento existe uma tabela diferente de durações das notas e dos *delays* entre os inícios delas. No andamento ♩=120, temos:

- 4 tempos: 2000
- 2 tempos: 1000
- 1 tempo: 500
- ½ tempo: 250
- ¼ de tempo: 125

Para saber as durações das figuras musicais em qualquer andamento utilize a seguinte calculadora: https://scratch.mit.edu/projects/314480551. Outra opção é verificar o apêndice em *Fórmula para encontrar a duração das figuras musicais*.

Assista ao seguinte vídeo intitulado *Entendendo o Sketch tocador de 2 notas,* para ter uma visão mais clara do funcionamento do código. O link é o seguinte:

<https://youtu.be/MkotspRT9ro>

2.4 Executando o código

Agora, você deve executar o código para verificar o funcionamento de tudo. Se for no simulador do *Tinkercad* basta clicar em *Iniciar simulação*. Se for em uma aplicação real, siga os seguintes passos:

1. Conecte o Arduino ao computador por meio do cabo USB.
2. Abra a IDE do Arduino com o Sketch que você deseja executar.
3. Vá no menu *Ferramentas* e em *Placa*. Selecione sua placa, que no nosso caso agora é *Arduino UNO*.
4. Depois, volte no menu *Ferramentas* e vá em *Porta* e procure pela porta que está sendo utilizada pelo Arduino. Se não souber qual é a porta, uma dica é desconectar o Arduino e verificar qual porta desaparece. Então, depois você conecta novamente o Arduino e confere a porta nova que apareceu - será a porta referente a ele!
5. Por fim, vá no menu *Sketch* e clique em *Carregar*. Pronto! O código será carregado no Arduino e o Arduino irá executar o

código que está carregado nele, para sempre, ou até que você carregue outro código, nele.

6. Para reiniciar a execução do Arduino você pode desconectar e conectar novamente o cabo USB; carregar novamente o Sketch ou, idealmente, clicar no botão *reset* (vermelho) no Arduino, que fica próximo ao conector USB.
7. Se algo não funcionar, verifique se digitou algo errado. Se ainda não funcionar experimente desconectar o Arduino, fechar a IDE e depois reiniciar os passos anteriores.
8. Observe que, depois que o código é carregado no Arduino, ele ficará armazenado na memória do Arduino e não necessita mais do computador. Basta ligar o Arduino a alguma fonte de alimentação adequada.

Assista ao seguinte vídeo para reforçar o entendimento sobre a montagem virtual do *Tocador de duas notas*:

<https://youtu.be/kGHUOstPvS8>

Se desejar entender melhor como se carrega um sketch no Arduino na montagem real do *Tocador de duas notas* assista ao vídeo intitulado *Carregando o Sketch no Arduino*, no link:

<https://youtu.be/p2GzMNaf4d8>

2.5 Desafio para músicos

Utilizando o mesmo circuito indicado no item anterior, modifique o sketch (programa) de forma que ele execute a seguinte partitura:

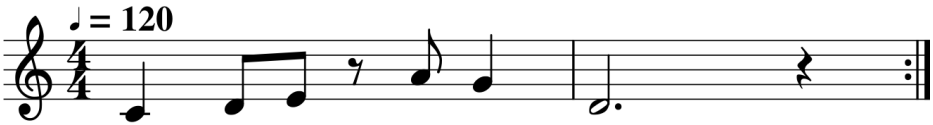


Figura 4.8 – Partitura para desafio

Uma dica: Lembre-se que uma coisa é a duração da nota musical e outra coisa é o *delay* entre o início de uma nota o início da outra nota. O fundamental é lembrar que a duração dentro da função *tone* não indica quanto tempo demorará para ir para a próxima linha e sim, simplesmente, quanto tempo durará, no máximo, a nota emitida.

Outra dica: Se você somar todos delas do void loop deste desafio o resultado correto deverá ser 4000 milissegundos.

2.6 Ideias para uso didático

Se você deseja utilizar o que aprendeu em uma aula de música, considere as seguintes sugestões e invente outras:

1-Alunos indicam diferentes frequências para cada uma das notas e professor executa para a turma. Pode ser explorada a relação da oitava com a frequência.

2-Criação de melodias utilizando mais notas. Aqui, podem ser trabalhadas questões sobre andamento e duração das figuras musicais.

2.7 Ideias de variação na programação

Para variar a programação, experimente:

1-Colocar algum conteúdo no void setup para que seja executado no início.

2- Na função tone (pino, frequência, duração) alterar a frequência e a duração).

3- Na função delay (duração) alterar a duração.

2.8 Resumo

Neste segundo capítulo, por meio de um hardware com um autofalante, um Arduino e um resistor, exploramos a linguagem de programação do Arduino (que é basicamente a linguagem C/C++, geral da computação, com alguns recursos específicos). Aprendemos sobre os seguintes elementos: void setup, void loop,

tone, delay, comentário com // e ponto e vírgula para finalizar cada linha de código. Além disso, vimos como carregar um programa (também chamado de *sketch*) na placa do Arduino.

3 Tocador com 3 LEDs

Na sessão anterior, aprendemos a trabalhar com um autofalante no Arduino intermediado por um resistor, em um dos terminais. Aqui, na presente sessão, será trabalhado o uso de LEDs. Iremos chamar o aparato de *Tocador com 3 LEDs*. Siga a montagem a seguir.

Material necessário:

- 1 LED vermelho
- 1 LED amarelo
- 1 LED verde
- 3 resistores de 220Ω
- 1 resistor de 150Ω
- 1 arduino Uno
- 1 autofalante
- 7 cabos flexíveis (ou conectores Dupont, macho-macho) de cores variadas.
- 1 mini placa de ensaio (opcional)

Faça as seguintes ligações:

1-Ligue o terminal mais comprido do **LED vermelho** ao pino 6 – porém, intermediado por um resistor de 330 ohms. O terminal mais curto deste LED deve ser ligado ao GND do Arduino.

2-Ligue o terminal mais comprido do **LED amarelo** ao pino 5. Porém, intermediado por um **resistor de 330 ohms**. O terminal mais curto deste LED deve ser ligado ao GND do Arduino.

3-Ligue o terminal mais comprido do **LED vermelho** ao pino 3. Porém, intermediado por um **resistor de 330 ohms**. O terminal mais curto deste LED deve ser ligado ao GND do Arduino.

4-Ligue um cabo em um dos terminais do **autofalante** intermediado por um **resistor qualquer de 150 a 270 ohms**. O outro terminal do autofalante fica ligado no GND do Arduino.

Se você observar bem, o fato é que o terminal mais curto de todos os LEDs e mais um dos terminais do autofalante estarão todos juntos, ligados ao GND do Arduino.

Para entender melhor, observe as ligações na Figura 3.1, que apresenta a montagem virtual e ainda as Figura 3.2 e 3.3, que apresentam a montagem real.

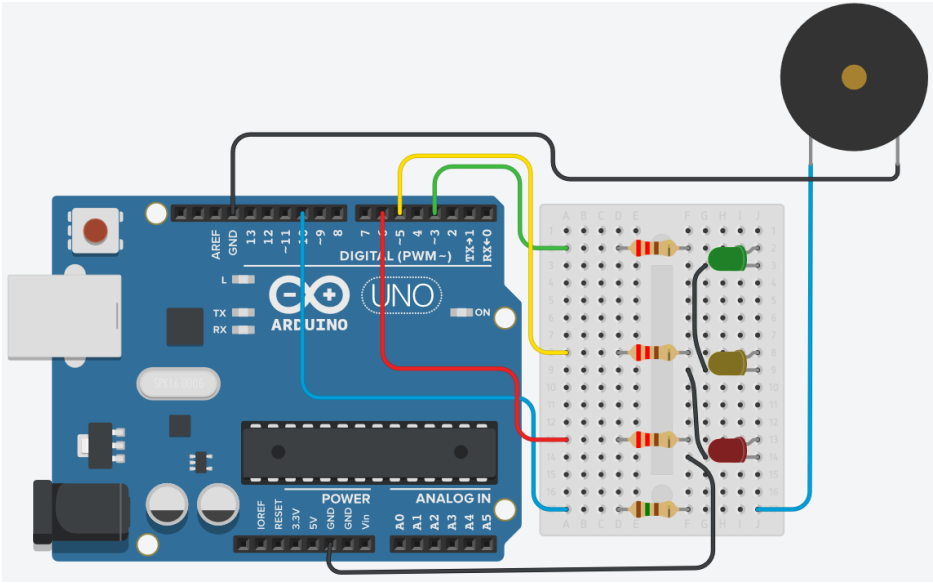


Figura 3.1 – Montagem virtual

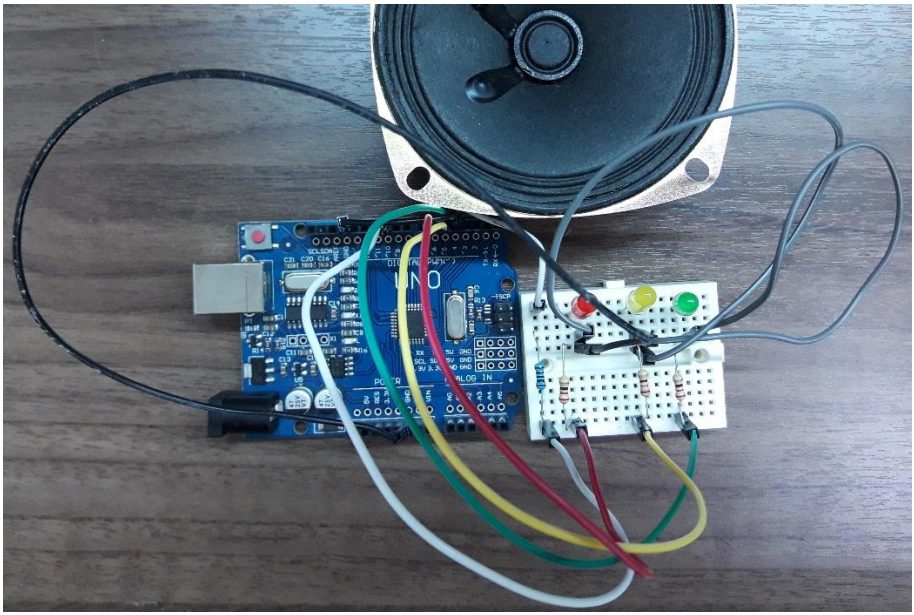


Figura 3.2 – Montagem real

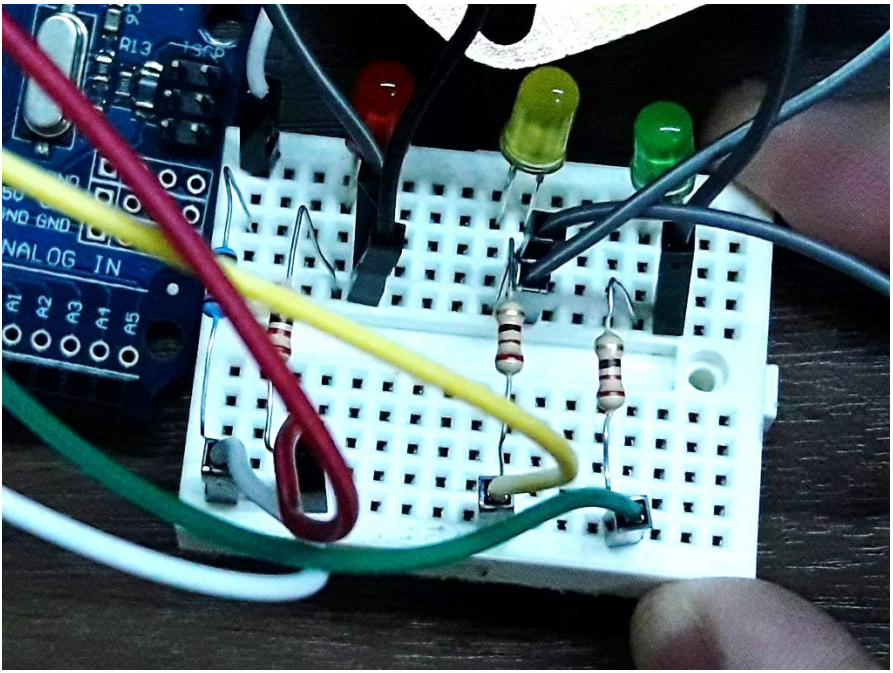


Figura 3.3 – Detalhe da protoboard na montagem real

Para ter acesso à montagem virtual, acesse o link:

<https://www.tinkercad.com/things/5ShuXnGQ0ra>

Para assistir ao vídeo *Montagem real do Tocador com 3LEDs*, utilize o seguinte link:

<https://youtu.be/cZS3rz6aAxQ>

3.1 Metrônomo visual

Anteriormente, vimos uma aplicação utilizando as funções *tone* e *delay* para tocar melodias em um autofalante. Agora, iremos explorar algo visual por meio deste projeto simples de metrônomo. Assista o vídeo *Metrônomo visual*, neste link para ter uma ideia inicial da aplicação:

<https://youtu.be/qKGF5RkgO5w>

Serão utilizados apenas os LEDs, mas não é problema deixar o autofalante já conectado, conforme foi indicado.

Este é um metrônomo visual, apenas com LEDs. Está em compasso 4/4 e em um andamento fixo de 100 tempos por minuto (♩=100). O LED vermelho é o tempo 1; o amarelo, o 2 e o 4; e o LED verde é o tempo 3.

Na IDE do Arduino, escreva o seguinte código:

```
//Metrônomo
int vermelho = 6;
int amarelo = 5;
int verde = 3;
int tempo = 600;
void setup()
{
  pinMode(vermelho, OUTPUT);
  pinMode(amarelo, OUTPUT);
  pinMode(verde, OUTPUT);
}
void loop()
{
  //Tempo 1
```

```
  digitalWrite (vermelho, 1);
  delay (tempo);
  digitalWrite (vermelho, 0);

  //Tempo 2
  digitalWrite (amarelo, 1);
  delay (tempo);
  digitalWrite (amarelo, 0);

  //Tempo 3
  digitalWrite (verde, 1);
  delay (tempo);
  digitalWrite (verde, 0);

  //Tempo 4
  digitalWrite (amarelo, 1);
  delay (tempo);
  digitalWrite (amarelo, 0);
}
```

Siga o código em azul e a seguinte explicação para entender como funciona cada linha:

Inicialmente, são atribuídos os valores 6, 5 e 3 para as variáveis que se referem aos pinos dos LEDs vermelho, amarelo e verde. Depois disso, é atribuído o valor 600 para a variável *tempo*.

No *void setup,* os pinos dos LEDs são configurados como saídas (OUTPUT), por meio da função *pinMode*. Isso faz com que o Arduino tenha certeza de que estes pinos atuarão como saídas digitais.

No *void loop* ocorre a emissão luminosa em cada tempo musical. São três linhas de comando para cada tempo musical. Inicia-se com a função *digitalWrite*, especificando-se o pino referente à cor do LED e depois, que o pino estará ligado (1).

Após o LED ser lidado, ocorre um delay (aqui indicado pela variável *tempo*). O cálculo é que, com um delay de 600 milissegundos temos a pulsação referente ao um andamento de 120 tempos por minuto (♩=120). Obs.: Para outros valores de delay referentes a andamentos musicais procurem em https://scratch.mit.edu/projects/314480551. Apenas para simplificar os experimentos iniciais, indica-se, no Quadro 3.1, algumas durações de figuras musicais, em milissegundos, no andamento de 100 tempos por minuto (♩=100).

Quadro 3.1 – Duração das figuras musicais no andamento ♩=100

Figura musical	Duração
Semibreve	2400
Mínima	1200
Tercina de mínima	800
Semínima	600
Tercina de semínima	400
Colcheia	300
Tercina de Colcheia	200
Semicolcheia	150

Voltando ao entendimento do código!

Após este *delay*, ocorre outra função *digitalWrite*, desta vez indicando que o respectivo LED irá desligar (0).

Observe que, gradualmente estamos aprendendo mais ideias e funções na linguagem C/C++ do Arduino. Então, resumindo, já conhecemos os seguintes elementos:

- ***//*** indica que tudo que estiver na direita em uma linha é um comentário.
- **;** indica o final de uma linha de código
- **Int qualquerPalavra = umNumeroQualquer** cria uma variável do tipo número inteiro e atribui um valor a ela.
- **void setup () { }** indica uma parte de código que funciona uma vez só.
- **void loop () { }** indica uma parte do código que fica repetindo, indefinidamente.

- **tone** (número do pino, frequência em hertz, duração em milissegundos);
- **delay** (duração em milissegundos);
- **pinMode** (número do pino, tipo de funcionamento do pino[18]).
- **digitalWrite** (número do pino, valor se está ligado ou desligado[19]).

3.1.1 Ideias para uso didático

Se você deseja utilizar o que aprendeu em uma aula de música, considere as seguintes sugestões e invente outras:

1-Estudantes devem marcar o tempo com as mãos conforme a visualização dos LEDs. Professor varia o andamento para vários exercícios.

2-Estudantes devem descobrir qual o compasso está sendo executado, conforme a marcação do tempo 1 pelo LED vermelho e outras marcações com os outros LEDs.

3.1.2 Ideias de variação na programação

Para variar a programação, experimente:

1-Mudar o valor atribuído à variável int tempo.

2-Mudar os tempos nos quais cada LED acende.

3-Acender mais de um LED ao mesmo tempo.

4-Mudar a quantidade de tempo do compasso.

---

[18] As opções para o tipo de funcionamento são: OUTPUT, INPUT e INPUT_PULLUP.

[19] Utilize 1 para ligado e 0 para desligado. Ou então, HIGH para ligado e LOW para desligado.

3.2 Metrônomo visual e sonoro

Para acrescentar som ao metrônomo elaborado no item anterior, adicione as seguintes linhas de código referentes à variável *pinoDoAutofalante*[20] e à função *tone*.

1. Nas definições das variáveis acrescente:

```
int pinoDoAutofalante = 10;
```

2. Depois, no início do tempo 1 acrescente:

```
tone (pinoDoAutofalante, 880, 5);
```

3. E no início de cada um dos tempos seguintes acrescente:

```
tone (pinoDoAutofalante, 440, 5);
```

O que se observa com essas linhas adicionais é que se está emitindo, no tempo 1, uma nota musical (lá4), com uma duração bem curta, 5 milissegundos. E nos demais tempos, um lá oitava abaixo, com a mesma duração. Ou seja, o ouvinte ouvirá cliques em cada tempo, para além de LEDs acendendo e apagando.

Assista o vídeo *Metrônomo visual e sonoro*, neste link, para obter um entendimento melhor do procedimento:

<https://youtu.be/8VBfmEm56do>

[20] Uma curiosidade: O leitor atento pode ter notado que, quando utilizamos variáveis formadas por várias palavras (por exemplo: *pinoDoAutofalante*), elas estão sempre juntas (sem espaço) e o nome da primeira palavra delas normalmente inicia em letra minúscula e depois as demais iniciam em maiúscula. Isso é um padrão estético utilizado por muitos programadores chamado de *lowerCamelCase*. Mas, fazer isso não é nenhuma obrigação. O que importa que é as variáveis sejam sem espaços e sem caracteres especiais e ainda não sejam palavras reservadas do linguem de programação. Por exemplo: você não poderia escrever `int void = 1;`, pois *void* é uma palavra reservada.

3.2.1 Ideias para uso didático

Se você deseja utilizar o que aprendeu em uma aula de música, considere as seguintes sugestões e ainda crie outras:

1-Estudantes indicam um andamento e tentam marcar o tempo. Depois o professor liga o Arduino para a turma conferir se a marcação estava certa ou errada.

2-Exploração de andamentos diversos e compassos variados.

3-Experimente construir um metrônomo gigante ligando cada LED, de auto brilho, dentro de uma garrafa PET e fixando tudo em um suporte. Isso pode se tornar uma ferramenta bastante interativa com a turma.

3.2.2 Ideias de variação na programação

Para variar a programação, experimente:

1-Mudar o valor atribuído à variável int tempo.

2-Mudar os tempos nos quais cada LED acende.

3-Acender mais de um LED ao mesmo tempo.

4-Mudar a quantidade de tempo do compasso.

5-Acrescentar a seguinte linha de código logo antes da chave final do void loop para ir acelerando o andamento, a cada compasso:

```
tempo = tempo - 10;
```

6-Acrescentar a seguinte linha de código logo antes da chave final do void loop para ir ficando mais lento o andamento, a cada compasso:

```
tempo = tempo + 10;
```

3.3 Randômicos

Vamos agora aprender uma nova função chamada de **random**. Ela serve para gerar números aleatórios entre um valor mínimo e um valor máximo. Ela possui dois parâmetros, como segue:

random (valor mínimo**,** valor máximo**)**

Para entender, na prática, iremos agora elaborar um sketch que faça com que o Arduino fique tocando sons em frequências aleatórias. Para o hardware vamos utilizar o mesmo circuito do *Tocador com 3 LEDs*. Para o software, utilize o seguinte código:

```
// Tocador de pulsos sonoros de frequências aleatórias

// atribuição de variáveis para pinos e para o delay
int vermelho = 6;
int amarelo = 5;
int verde = 3;
int pinoDoAutofalante = 10;

int pulso= 200;

void setup()
{
  pinMode(vermelho, OUTPUT);
  pinMode(amarelo, OUTPUT);
  pinMode(verde, OUTPUT);


//Essa, a seguir, é uma vinheta inicial de abertura do programa
  tone (pinoDoAutofalante, 80, 500);
  digitalWrite (amarelo, 1);
  delay (500);
  digitalWrite (amarelo, 0);
   delay (1000);
}
```

```
void loop()
{
  int a = random (220, 440);

  tone (pinoDoAutofalante, a, 100);
  digitalWrite (vermelho, 1);
  delay (pulso);

  digitalWrite (vermelho, 0);
  tone (pinoDoAutofalante, a * 2, 100);
  digitalWrite (verde, 1);
  delay (pulso);

  digitalWrite (verde, 0);
}
```

Entenda o que o sketch faz:

Inicialmente, em marrom, são criadas variáveis e atribuídos valores a elas. Algumas delas são referentes a pinos onde os LEDs estão conectados (*vermelho*, *amarelo* e *verde*), uma outra refere-se ao pino do autofalante (*pinoDoAutofalante*) e a última à duração da pulsação que será emitida (*pulso*).

Em azul claro é indicado que cada pino de cada LED é uma saída, por meio da função *pinMode*.

Depois temos uma parte que é uma espécie de vinheta de abertura sonora e visual do programa. Em verde, vemos que é emitido um sinal sonoro enquanto o LED amarelo é ligado. Depois deste som emitido, o LED apaga e espera-se um segundo até que entre no void loop.

No void loop iniciamos criando uma variável de número inteiro chamada de ***a*** e atribuímos um número aleatório, entre 220 e 440). Ou seja, a cada vez que iniciar o loop, um novo sorteio é realizado e o valor de ***a*** é modificado.

Depois disso é emitido um som na frequência da variável ***a***. "Paralelamente"[21] a isso, o LED vermelho é ligado e logo depois desligado. Então, uma segunda nota é emitida, na frequência do mesmo valor de ***a*** só que multiplicado[22] por 2 (indicado por *a * 2*). Ou seja, será o dobro da frequência anterior, ou seja, a nota soará oitava acima. Paralelamente a essa segunda nota, o LED verde é aceso e depois apagado.

Então, tudo é repetido no void loop, para sempre.

Experimente modificar coisas para variar o código. Como será que fica a melodia gerada ao modificarmos o valor de multiplicação de 2 para 3 ou para 4?

E se formos mais radicais e mudarmos para valores como 9/8, ou 81/64, ou 4/3, ou 3/2, ou 27/16, ou 243/128[23]? Será que você consegue perceber o resultado musical dessa façanha? Porém, existe um problema computacional para revolvermos antes dessa aventura. Você lembra da variável do tipo número inteiro que utilizamos? No caso era a ***a***. Pois bem! Quando multiplicarmos este ***a*** por uma destas frações, irá resultar em um número que não é inteiro. Então o programa não vai calcular corretamente. Ou seja, não dá para usar aqui uma variável do tipo *int* (inteira). Devemos utilizar outro tipo de variável chamada de *float* que é utilizada para números "quebrados". Então, você deve fazer a seguinte mudança no código

Substitua a linha:

```
int a = random (220, 440);
```

[21] O "paralelamente" está entre aspas pois, na realidade, não é paralelamente, já que o código sempre lê uma linha de cada vez, na sequência. O que ocorre é que isso é tão rápido que não conseguimos perceber que uma coisa ocorreu depois da outra.

[22] Em computação, no lugar do **xis**, utiliza-se o **asterisco** para indicar a multiplicação.

[23] Se estiver curioso sobre estas frações procure no apêndice por algo relacionado a elas.

por esta:

```
float a = random (220, 440);
```

Agora sim, você irá poder testar seu experimento de acústica musical mudando os valores de multiplicação. Boa sorte!

3.3.1 Ideias para uso didático

Se você deseja utilizar o que aprendeu em uma aula de música, considere as seguintes sugestões e ainda crie outras:

1-Compositor maluco. Brinque com os alunos criando composições aleatórias modificando os diversos parâmetros, inclusive com outras variáveis aleatórias.

3.3.2 Ideias de variação na programação

Para variar a programação, experimente:

1-Criar uma terceira nota em outra oitava de forma a utilizar os 3 LEDs.

2-Crie uma variável randômica para o pulso acrescentando a seguinte linha antes da chave final do void loop:

```
pulso = random (10, 250);
```

Assista o vídeo *Randômicos*, neste link, para obter um entendimento melhor do sketch:

<https://youtu.be/BF6f0LQBOqA>

3.4. IFs

Agora vamos explorar uma estrutura condicional chamada de *IF*. Vamos entender na prática, fazendo um *tocador aleatório de notas da tríade de sol menor*. Para ter uma visão geral do funcionamento do projeto, assista o vídeo *IFs*, no seguinte link:

<https://youtu.be/BULB5PDZGzI>

Para o hardware vamos utilizar o mesmo circuito (o Tocador de 3LEDs). Para o software, utilize o seguinte código e aproveite para ler os comentários, nele, para entender o seu funcionamento:

```
//Este é um tocador aleatório de notas da tríade de sol menor

//Variáveis para a definição das portas do Arduino
int pinoDoLEDVermelho = 6;
int pinoDoLEDAmarelo = 5;
int pinoDoLEDVerde = 3;
int pinoDoAutofalante = 10;

//Variável para a duração da nota e do delay
int duracao = 500;

//A seguir temos o void setup que é uma parte do programa que é executada inicialmente e uma única vez.
void setup()
{
  //A seguir temos as definições sobre os pinos do Arduino que devem emitir um sinal elétrico para o LED.
pinMode (pinoDoLEDVermelho, OUTPUT);
pinMode (pinoDoLEDAmarelo, OUTPUT);
pinMode (pinoDoLEDVerde, OUTPUT);
//A seguir é emitido um sinal sonoro, como se fosse uma vinheta de inicialização do programa.
tone (pinoDoAutofalante, 98, 1000);
```

```
delay (2000);
}

// A seguir temos o void loop que fica repetindo, para sempre, os comandos em seu interior.
void loop()
{
//Aqui, no início do loop é criada uma variável e atribuído um valor randômico entre 1 e 3 a ela.
int a = random (1 , 4);

//A seguir temos 3 ifs. Um para cada uma das três possibilidades da variável "a".

//Se for sorteada a Fundamental da tríade
if (a == 1)
{
tone (pinoDoAutofalante, 98, duracao);
digitalWrite (pinoDoLEDVermelho, 1);
delay (duracao);
digitalWrite (pinoDoLEDVermelho, 0);
}

// Se for sorteada a Terça da tríade
if (a == 2)
{
tone (pinoDoAutofalante, 117, duracao);
```

```
digitalWrite (pinoDoLEDAmarelo, 1);
delay (duracao);
digitalWrite (pinoDoLEDAmarelo, 0);
}

// Se for sorteada a Quinta da tríade
if (a == 3)
{
tone (pinoDoAutofalante, 147, duracao);
digitalWrite (pinoDoLEDVerde, 1);
delay (duracao);
digitalWrite (pinoDoLEDVerde, 0);
}
delay (duracao);
}
```

Observando o sketch apresentado vemos que a estrutura do if é a seguinte:

If (algumaVariavel alguma condição matemática um valor)

{

As linhas de código do que ocorre dentro do if;

}

As condições matemáticas (**operadores de comparação**) normalmente utilizadas são as seguintes:

!= (diferente de)

< (menor que)

<= (menor que ou igual a)

== (igual a)

> (maior que)

>= (maior que ou igual a)

No nosso exemplo utilizou-se o comparador ==, ou seja, *igual a*.

3.4.1 Ideias para uso didático

Se você deseja utilizar o que aprendeu em uma aula de música, considere a seguinte sugestão e ainda crie outras:

1-Jogo quem acerta a nota. Crie um delay de alguns segundos entre a nota emitida e o acendimento do LED correspondente. Isso permitirá que o estudante busque acertar qual a nota do acorde foi tocada e depois conferir o resultado por meio do LED.

3.4.2 Ideias de variação na programação

Para variar a programação, experimente:

1-Criar as notas da tríade em outras oitavas. Ou seja, no exemplo, além do sol grave poderia ocorrer um sol agudo e, mesmo assim, o LED verde acender, para ambos os casos, e assim por diante.

2-Crie um quarto *if* contendo apenas um delay e deixe o valor máximo do random em 5.

3.5 Resumo

Neste terceiro capítulo utilizamos uma montagem com um autofalante, 3 LEDs e alguns resistores. Em termos de software aprendemos sobre: int, float, pinMode, digitalWrite, random, estrutura if e comparador == (igual a).

4. Teclado musical simples

Neste capítulo iremos instruir a construção, a programação e a utilização de um teclado musical, simples como o ilustrado na Figura 4.1.

Figura 4.1 – Teclado musical simples

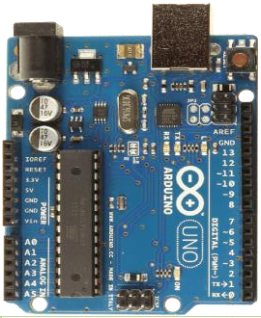
Para assistir a uma demonstração do instrumento sendo executado, acesse o vídeo *Amazing Grace com o TGL*, no link:

<https://youtu.be/lqXbOzkFFYY>

4.1 Montagem do Hardware do teclado musical

Para a construção indica-se a seguinte listagem de materiais, conforme o Quadro 4.1.

Quadro 4.1 – Material necessário para o teclado musical simples

Quantidade	Descrição	Aparência
16	Parafuso auto-atarraxante, Philips. De 20x3mm.	
1	Placa de Arduino Uno	
20	Cabos flexíveis com conectores Dupont macho e macho	
1	Resistor de 150 a 220 ohms, 5%, 1/4W	
1	Caixa de som (pode ser uma caixa de som velha de computador)	

1	Lata de leite em pó, ou semelhante, vazia.	
2	Haste de cotonete	
1	Haste metálica (raio de bicicleta, arrame grosso ou similar) com cerca de 20 cm	
1	Placa de compensado de 20x20x3mm	

1	Ferramentas diversas para a montagem: 1 tesoura, 1 prego, 1 martelo, 1 chave Philips, 1 lixa (opcional) 1 ferro de soldar (opcional) e 1 pedaço de estanho para solda (opcional). 1 kit com tinta guache e pinceis (opcional)	

A seguir, estão indicados os passos a serem realizados para a construção do teclado musical simples.

4.1.1 Passo 1 – Placas

Desencape uma lata de leite em pó, ou qualquer lata de metal, que não seja pintada.

Depois, com uma tesoura forte, recorte a lateral da lata de maneira a formar 9 placas metálicas de 10x1,5cm.

Em seguida, com um alicate, faça duas dobras: uma 90º para baixo, a 3 cm de uma das extremidades; e outra, 90º para cima, a 2 cm da extremidade. A Figura 4.2 ilustra como é para ficar cada placa.

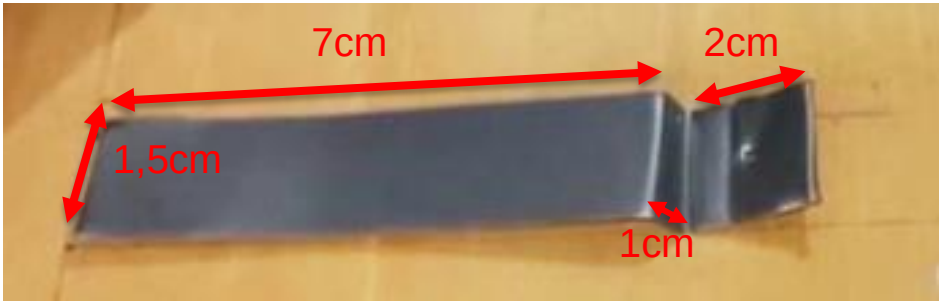


Figura 4.2 – Medidas de uma placa cortada e dobrada

4.1.2 Passo 2 – Compensado

Serre um pedaço de folha de compensado de 25x20x2cm. Ou seja: 25 centímetros de largura, 20 de altura e 2 centímetros de espessura. Se você não tiver material ou aptidão para serrar madeira, recomenda-se que encomende a peça em alguma marcenaria.

Na peça, desenhe, à lápis, um gabarito de onde as placas ficarão dispostas. Siga as medidas da Figura 4.3.

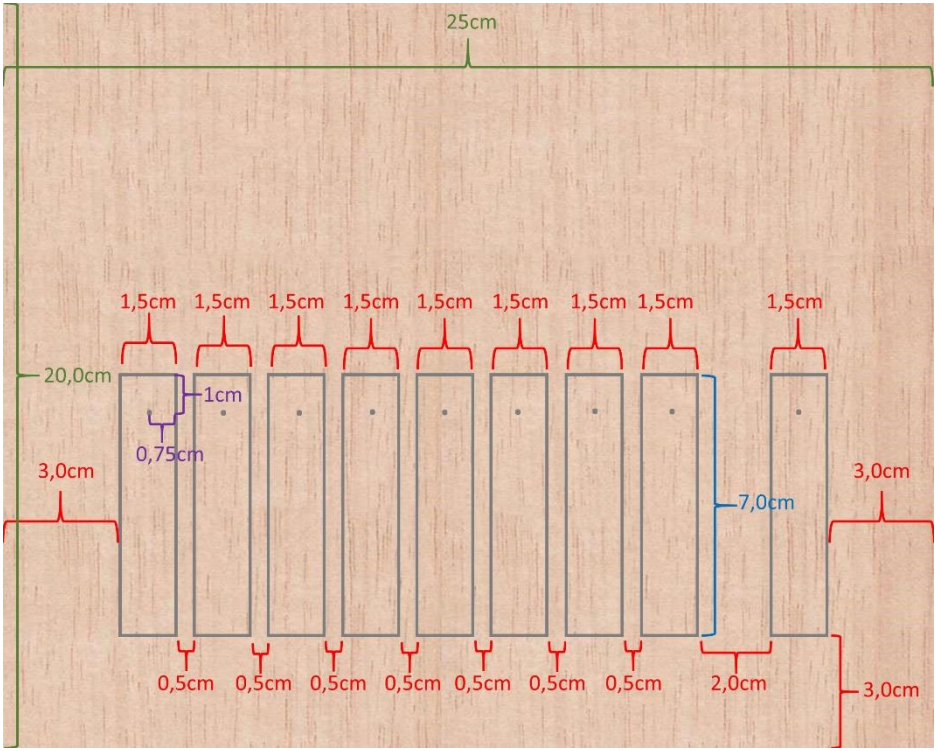


Figura 4.3 – Gabarito para posicionamento das placas

Depois de marcar, posicione as teclas para ver se está tudo correto, como na Figura 4.4.

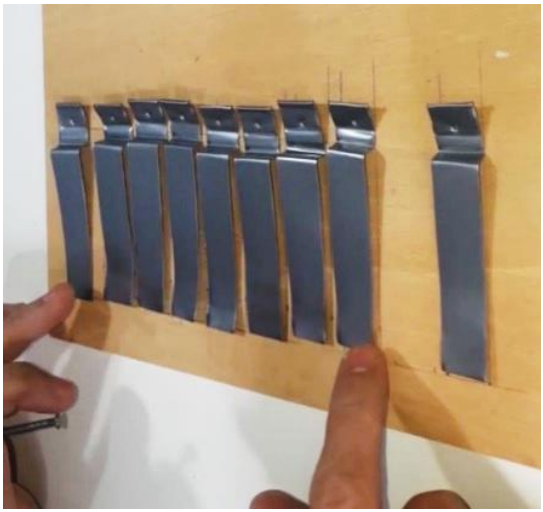


Figura 4.4 – Verificação do posicionamento das teclas

4.1.3 Passo 3 – Preparar os cabos

Agora, prepare os cabos. Você deve cortar a extremidade de 10 cabos com conectores Dupont (jumpers). Ou seja, devem ficar preparados 10 cabos com em uma extremidade apenas o cabo e na outra com o conector Dupont. É para ficar como na Figura 4.5.

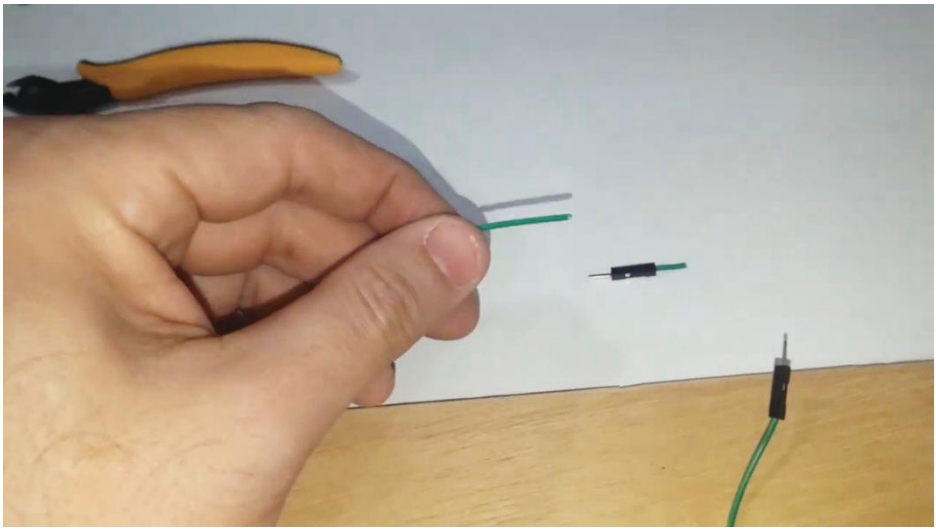


Figura 4.5 – Um cabo Dupont com uma das pontas cortadas

Depois de cortar as pontas dos cabos, descasque a capa do cabo para ficar cerca de 1cm de cabo aparente na ponta. Este desencapado será a parte que você irá, depois, conectar ao parafuso da placa metálica ou ao parafuso da haste metálica.

4.1.4 Passo 4 – Pré-fixe as placas

Realize um furo no compensado para cada placa metálica, utilizando o próprio parafuso de fixação e uma chave Philips. Ficará como ilustra a Figura 4.6.

Figura 4.6 – Furo dos parafusos no local adequado

Em seguida, com um prego, faça um pequeno orifício a 1 cm da extremidade de cada placa, ao centro. Depois, aparafuse cada placa em seu respectivo lugar. Não necessita parafusar o parafuso até o final (vide a Figura 4.7).

Figura 4.7 – Placas semi aparafusadas no local

4.1.5 Passo 5 – Fixação dos cabos nas placas

Agora, um a um, retire os parafusos e insira no orifício as pontas desencapadas dos cabos (Figura 4.8).

Figura 4.8 – Inserindo a ponta do cabo no orifício da placa e do compensado

Em seguida, coloque novamente o parafuso, com o cabo dentro do orifício. Tenha muito cuidado para não quebrar o cabo nesta operação. A Figura 4.9 ilustra a operação já com 4 cabos afixados nas placas, com os parafusos totalmente atarraxados e com as placas firmes nos locais. Faça isso para todas as 9 placas.

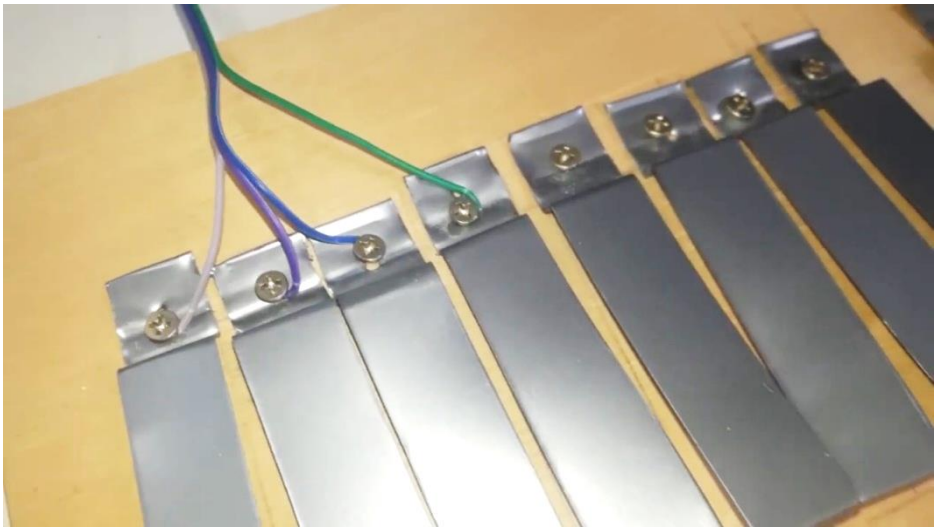


Figura 4.9 – Algumas placas já fixadas, com os cabos

4.1.6 Passo 6 – Haste metálica de contato

Com um alicate, dobre as extremidades da haste metálica (o arame) de forma que fique esticada e firme em dois parafusos (Figura 4.10).

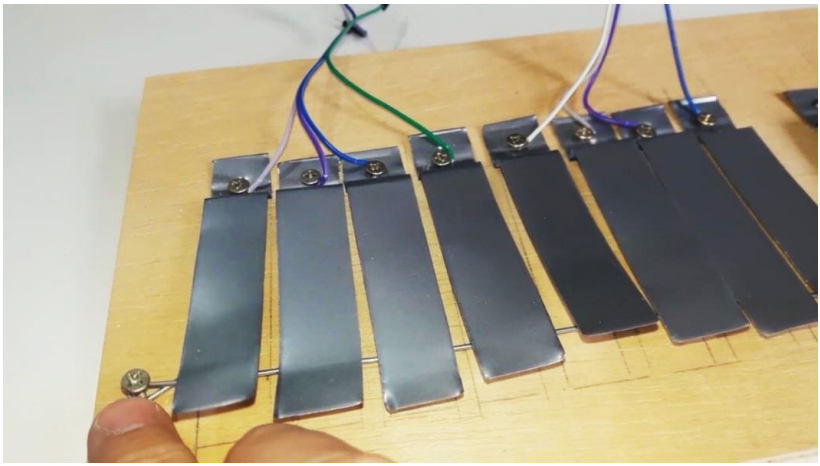


Figura 4.10 – Haste posicionada

Antes disso, enrole algumas voltas de papel alumínio na haste metálica para melhorar o contato elétrico e amenizar o ruído das teclas.

Por fim, fixe um décimo cabo em um dos parafusos da haste metálica, conforme o procedimento realizado nas 9 placas (Figura 4.11).

Figura 4.11 – Cabo da haste posicionado

4.1.7 Passo 7 – Caixa de som

Para preparar a caixa de som (autofalante[24]), inicie desencapando a pontas dos cabos da caixa de som. Em seguida, em um dos cabos, conecte uma das sobras de cabos com conector Dupont que havia cortado anteriormente. Ao outro cabo da caixa de som, conecte um dos terminais resistor; e em seguida, conecte o outro terminal do resistor à outra ponta de cabo Dupont que você havia cortado no passado.

Para realizar estas conexões, escolha **uma** das seguintes opções:

1. Enrole, com os dedos, os cabos e depois coloque um pedaço de **haste flexível (cotonete)** para firmar um pouco a ligação e fazer a isolação elétrica (Figura 4.12).
2. Enrole, com os dedos, os cabos e depois coloque um pedaço **fita isolante** para firmar um pouco a ligação e fazer a isolação elétrica.
3. Enrole com os dedos os cabos e depois **solde** a ligação com ferro de soldar e estanho. Em seguida, utilize fita isolante ou um isolante termo retrátil para fazer a isolação elétrica (Figura 4.13).

[24] O ideal é deixar o autofalante dentro de uma caixa de som para que a acústica da caixa deixe o som mais encorpado.

Figura 4.12 – Ligação com haste flexível (cotonete)

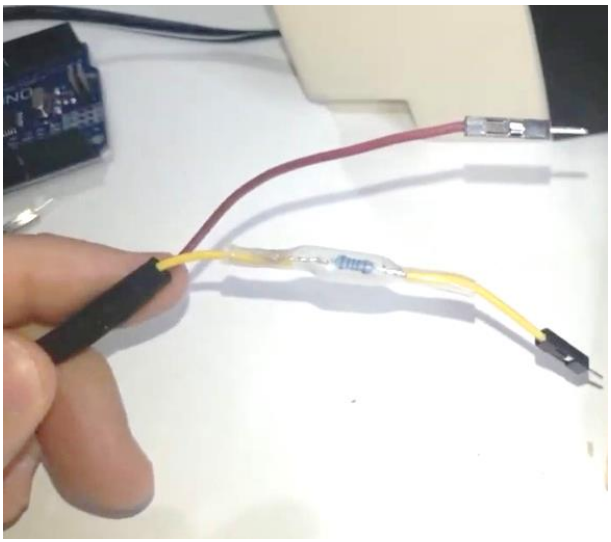


Figura 4.13 – Ligação com soldagem e isolante termo retrátil transparente cobrindo o resistor; e preto, cobrindo os cabos

4.1.8 Passo 8 – Saboneteira

Coloque o Arduino Uno dentro da saboneteira para medir onde irá realizar os furos. A ideia é fazer orifícios para passar os 12 cabos (9 das teclas, 1 da haste e 2 da caixa de som). Além disso, em uma das extremidades da saboneteira e crie dois furos maiores para passar o conector USB e o cabo da fonte de alimentação. Por fim, fixe a saboneteira na placa de compensado, com dois parafusos. Cuidado para não se cortar ou se queimar na operação!

4.1.9 Passo 9 – Acabamento

Se desejar, pode realizar um acabamento no teclado conforme as seguintes indicações:

1. Passar cola quente na extremidade superior das placas para ficarem mais firmes e para os cabos ficarem mais protegidos (Figura 4.14)

Figura 4.14 – Cola quente, para maior resistência

2. Pintar tudo, exceto os locais de contato elétrico (obviamente) – Vide a Figura 4.15 para verificar o saco plástico cobrindo a haste metálica para garantir que não seja pintada. Para pintar partes de plástico, indica-se passar uma camada de *tinta primer para plásticos* antes de fazer a pintura definitiva. A Figura 4.16 Ilustra a pintura final com a cor branca.

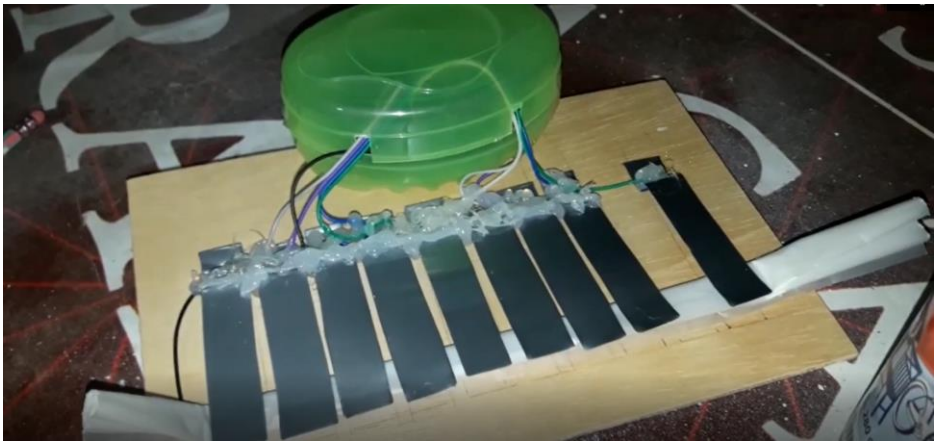


Figura 4.15 – Protegendo a haste da pintura

Figura 4.16 – Pintura com a cor branca

3. Recortar a placa de EVA e colar tiras sobre as teclas; e fazer outras ornamentações conforme seu gosto estético. Além disso, se desejar, pode desmontar a caixa de som, retirar o autofalante, fazer alguns orifícios na tampa da saboneteira, e colar o autofalante na saboneteira com cola quente. Isso possibilitará o teclado ficar mais compacto, embora, o seu som possa piorar, significantemente (Figura 4.17).

Figura 4.17 – Aparência final do teclado

4.1.10 Passo 10 – Conexão do Arduino

Aparafuse o Arduino na saboneteira.

Depois, cuidadosamente, realize as ligações dos cabos nos pinos do Arduino, conforme indicado na Figura 4.18: Teclas de 1 a 8 nos pinos de 2 a 9; tecla de Função no pino 10; O cabo o autofalante ligado no pino 11; o outro cabo do autofalante no GND[25]; O cabo das hastes em outra entrada GND.

[25] A bem da verdade não importa qual lado você liga o autofalante, basta um lado ficar no GND e o outro no pino 11.

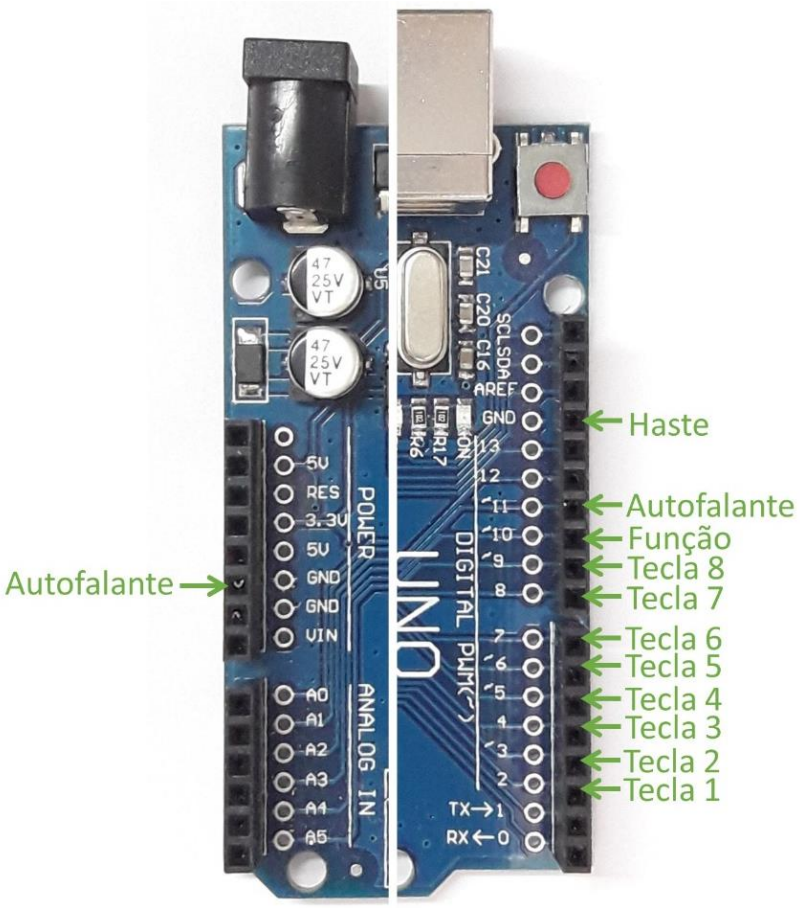


Figura 4.18 – Pinos e cabos no Arduino

4.1.11 Tutorial completo em vídeo sobre o TGL

Para ver o tutorial completo sobre a construção e utilização deste teclado musical assista ao vídeo *TGL- Teclado Musical Eletrônico Digital Artesanal com 8 teclas e seletor de funções* no seguinte link:

<https://youtu.be/xTvESCEAdks>

4.2 Preparando o Software do teclado musical

Abra o software IDE do Arduino em seu computador e crie um novo sketch (Menu: File → New).

Salve o Arquivo (Menu: File → Save) com o seguinte nome: TecladoMusicalMonofonicoDe8Notas.

Selecione o conteúdo do sketch, pré-criado pela IDE e o apague (*Delete*).

Em seguida, selecione **todo** o código com a fonte em cor azul, a seguir, e cole no sketch da IDE. Ignore as cores do fundo código, elas não serão nada para o Arduino, só estão aqui para facilitar o seu entendimento ao perceber as várias partes do código.

```
/*        TGL - Teclado Musical Monofônico de 8 notas

            Glauber Santiago 2020

 Este é um sketch de um teclado musical monofônico de 8 notas. Possui 8 teclas para as notas de dó 3 a dó 4, e mais uma tecla de função.

Enquanto se pressiona a tecla função, as seguintes funções são selecionadas ao se apertar, simultaneamente, uma das teclas de 1 a 8:

Tecla 1: Oitava 1

Tecla 2: Oitava 2

Tecla 3: Oitava 3

Tecla 4: Oitava 4

Tecla 5: Oitava 5

Tecla 6: Função adicional 1 - pulso da nota com a oitava 1

Tecla 7: Função adicional 2 - pulso da nota com a quinta e mais a oitava 1

Tecla 8: Função adicional desligada - volta à execução normal da nota única por tecla.


Para o hardware necessita-se de:

9 Botões que, idealmente, sejam manufaturados artesanalmente sob o formato de teclas
```

```
        1 Arduino Uno
        1 Resistor de 150 a 220 ohms
        1 Autofalante de 4 ou 8 ohms (ou qualquer um de uma caixa de som de computador)

        Para a ligação, ligue um dos terminais de cada tecla nos pinos de 2 a 10 do Arduino.
        O outro terminal das teclas vai para o GND do Arduino.
        Um terminal do autofalante vai no pino 11 do Arduino, intermediado pelo resistor, e o outro vai direto no GND.
        */

//A seguir, definições para os pinos das teclas e do autofalante.
#define tecla1 2
#define tecla2 3
#define tecla3 4
#define tecla4 5
#define tecla5 6
#define tecla6 7
#define tecla7 8
#define tecla8 9
#define teclaDeFuncao 10
#define autofalante 11

// A seguir, definições para as frequências das notas na oitava 3.
#define do3 261.63
#define re3 293.66
#define mi3 329.63
#define fa3 349.23
#define sol3 392.00
```

```
#define la3 440.00
#define si3 493.88
#define do4 523.25


//Inicialização da variável que faz a multiplicação para gerar as diferentes oitavas.
float oitavador = 1;


//Inicialização da variável utilizada para as "funções adicionais".
int funcaoAdicional = 0;
```

void setup() // O void setup é a parte do programa que é executada uma vez, logo que o Arduino é ligado.

{

```
  //Indicação que os pinos das teclas estão no modo INPUT_PULLUP)
  pinMode (tecla1, INPUT_PULLUP);
  pinMode (tecla2, INPUT_PULLUP);
  pinMode (tecla3, INPUT_PULLUP);
  pinMode (tecla4, INPUT_PULLUP);
  pinMode (tecla5, INPUT_PULLUP);
  pinMode (tecla6, INPUT_PULLUP);
  pinMode (tecla7, INPUT_PULLUP);
  pinMode (tecla8, INPUT_PULLUP);
  pinMode (teclaDeFuncao, INPUT_PULLUP);
```

}

void loop() // O void loop é a parte do programa que fica repetindo, indefinidamente.

{

```
  //A seguir temos os comandos de "enquanto uma tecla for pressionada" uma nota é tocada, conforme cada tecla.
```

```
 while (digitalRead (tecla1) == LOW) // Aqui, "LOW" significa que a tecla está sendo pressionada.
 {
  // A seguir, a função "tone" executará, no pino "autofalante", uma determinada frequência multiplicada pelo oitavador.
  tone (autofalante, do3 * oitavador);
  if (funcaoAdicional == 1) //Esta é a função adicional 1. Nela, a nota na oitava 1 é adicionada como um pulso.
  { delay (40);
    tone (autofalante, do3 * 0.25);
    delay (40);
  }
  if (funcaoAdicional == 2) //Esta é a função adicional 2. Nela, uma nota em intervalo musical de quinta justa é adicionada e depois outra nota na oitava grave, como um pulso.
  { delay (30);
    tone (autofalante, do3 * 1.5);
    delay (20);
    tone (autofalante, do3 * 2);
    delay (30);
  }
 }
```

```
 while (digitalRead (tecla2) == LOW)
 {
  tone (autofalante, re3 * oitavador);
  if (funcaoAdicional == 1)
  { delay (40);
    tone (autofalante, re3 * 0.25);
    delay (40);
  }
```

```
  if (funcaoAdicional == 2)
  { delay (30);
    tone (autofalante, re3 * 1.5);
    delay (20);
    tone (autofalante, re3 * 2);
    delay (30);
  }
}
```

```
while (digitalRead (tecla3) == LOW)
{
  tone (autofalante, mi3 * oitavador);

  if (funcaoAdicional == 1)
  { delay (40);
    tone (autofalante, mi3 * 0.25);
    delay (40);
  }
  if (funcaoAdicional == 2)
  { delay (30);
    tone (autofalante, mi3 * 1.5);
    delay (20);
    tone (autofalante, mi3 * 2);
    delay (30);
  }
}
```

```
while (digitalRead (tecla4) == LOW)
{
```

```
  tone (autofalante, fa3 * oitavador);
  if (funcaoAdicional == 1)
  { delay (40);
    tone (autofalante, fa3 * 0.25);
    delay (40);
  }
  if (funcaoAdicional == 2)
  { delay (30);
    tone (autofalante, fa3 * 1.5);
    delay (20);
    tone (autofalante, fa3 * 2);
    delay (30);
  }
}
```

```
while (digitalRead (tecla5) == LOW)
{
  tone (autofalante, sol3 * oitavador);
  if (funcaoAdicional == 1)
  { delay (40);
    tone (autofalante, sol3 * 0.25);
    delay (40);
  }
  if (funcaoAdicional == 2)
  { delay (30);
    tone (autofalante, sol3 * 1.5);
    delay (20);
    tone (autofalante, sol3 * 2);
    delay (30);
```

```
  }
}
```

```
while (digitalRead (tecla6) == LOW)
{
  tone (autofalante, la3 * oitavador);
  if (funcaoAdicional == 1)
  { delay (40);
    tone (autofalante, la3 * 0.25);
    delay (40);
  }
  if (funcaoAdicional == 2)
  { delay (30);
    tone (autofalante, la3 * 1.5);
    delay (20);
    tone (autofalante, la3 * 2);
    delay (30);
  }
}
```

```
while (digitalRead (tecla7) == LOW)
{
  tone (autofalante, si3 * oitavador);
  if (funcaoAdicional == 1)
  { delay (40);
    tone (autofalante, si3 * 0.25);
    delay (40);
  }
  if (funcaoAdicional == 2)
```

```
  { delay (30);
    tone (autofalante, si3 * 1.5);
    delay (20);
    tone (autofalante, si3 * 2);
    delay (30);
  }
}
```

```
while (digitalRead (tecla8) == LOW)
{
  tone (autofalante, do4 * oitavador);
  if (funcaoAdicional == 1)
  { delay (40);
    tone (autofalante, do4 * 0.25);
    delay (40);
  }
  if (funcaoAdicional == 2)
  { delay (30);
    tone (autofalante, do4 * 1.5);
    delay (20);
    tone (autofalante, do4 * 2);
    delay (30);
  }
}
```

```
// A seguir, temos as ocorrências para cada tecla, quando a tecla função está ativada.
while (digitalRead(teclaDeFuncao) == LOW) // tecla de função ativada.
{
  delay (20);
```

```
  if (digitalRead(tecla1) == LOW) {oitavador = 0.25;}
  if (digitalRead(tecla2) == LOW) {oitavador = 0.5;}
  if (digitalRead(tecla3) == LOW) {oitavador = 1;}
  if (digitalRead(tecla4) == LOW) {oitavador = 2;}
  if (digitalRead(tecla5) == LOW) {oitavador = 4;}
  if (digitalRead(tecla6) == LOW) {funcaoAdicional = 1;}
  if (digitalRead(tecla7) == LOW) {funcaoAdicional = 2;}
  if (digitalRead(tecla8) == LOW) {funcaoAdicional = 0;}
 }

delay (10); // Faz uma pequena pausa no loop para não exigir muito do processamento.

 noTone (autofalante); // Desliga o som do autofalante.

} //Esta é a chave final do void loop.
```

Depois de copiar e colar todo este código em azul, acima, salve o sketch (Menu: File → Save).

4.3 Carregando o Software do teclado musical

1. Conecte o Arduino ao computador por meio do cabo USB.
2. Abra a IDE do Arduino com o Sketch que você deseja executar (TecladoMusicalMonofonicoDe8Notas).
3. Vá no menu *Ferramentas* e em *Placa*. Selecione sua placa, que no nosso caso agora é *Arduino UNO*.
4. Depois, volte no menu *Ferramentas* e vá em *Porta* e procure pela porta que está sendo utilizada pelo Arduino. Se não souber qual é a porta, uma dica é desconectar o Arduino e verificar qual porta desaparece. Então, depois você conecta novamente o Arduino e confere a porta nova que apareceu - será a porta referente a ele!

5. Por fim, vá no menu *Sketch* e clique em *Carregar*. Pronto! O código será carregado no Arduino e o Arduino irá executar o código que está carregado nele, para sempre, ou até que você o carregue com outro código.
6. Para reiniciar a execução do Arduino, você pode desconectar e conectar novamente o cabo USB; carregar novamente o Sketch; ou, idealmente, clicar no botão *reset* (vermelho) no Arduino, que fica próximo ao conector USB.
7. Se algo não funcionar, verifique se digitou algo errado. Se ainda não funcionar, experimente desconectar o Arduino, fechar a IDE e depois reiniciar os passos anteriores.
8. Observe que depois que o código é carregado no Arduino ele ficará armazenado na memória do Arduino e não necessita mais do computador. Basta ligar o Arduino a alguma fonte de alimentação adequada.

4.4 Entendendo o sketch do teclado musical

Se você seguiu todas as etapas anteriores já deve estar utilizando seu teclado musical e realizando ajustes, melhorias e personalizações. Mas, apenas para um melhor aprendizado, iremos apresentar neste tópico o funcionamento do sketch. Vamos lá?

Em linhas gerais o *Teclado Musical Monofônico de 8 Notas* funciona assim:

Logo no início são feitas algumas definições e atribuições de valores a algumas variáveis a serem utilizadas no programa.

Depois (no void setup) é informado ao Arduino que os pinos das teclas são entradas digitais. Ou seja, quando se fizer uma ligação elétrica entre um destes pinos e o GND, o Arduino saberá que algo mudou, ou seja, a tecla foi pressionada.

Depois disso, o programa entra em um loop infinito que é onde, de fato, o teclado funciona. A lógica do teclado tocar é a seguinte:

Enquanto uma tecla tal (de 1 a 8) for pressionada, ele vai tocar uma nota na frequência definida para a nota, conforme a oitava estabelecida. Se for indicada a função adicional 1, ele irá executar, além desta nota uma outra nota, na oitava 1. Se for escolhida a função adicional 2, ele executará ainda uma nota a mais, uma quinta acima da oitava central.

Enquanto a tecla de função for pressionada, as teclas anteriores perdem a função de tocar e realizam operações diferentes como mudar a oitava e mudar função adicional. Vejamos em detalhe!

Esta, a seguir, é a primeira parte do sketch, na verdade é apenas um grande comentário aberto pela instrução /* e finalizado pela instrução */. Não interfere em nada na programação.

```
/*      TGL - Teclado Musical Monofônico de 8 notas

...

 */
```

Depois, temos uma parte que utiliza o elemento chamado de *#define*. O objetivo é criar algumas definições, atribuições, sobre o número dos pinos. Por exemplo, com #define tecla1 2, em qualquer lugar do sketch que eu utilizar o termo tecla1, o programa entenderá que isto é o número 2.

```
//A seguir, definições para os pinos das teclas e do autofalante.
#define tecla1 2
#define tecla2 3
…
#define teclaDeFuncao 10
#define autofalante 11
```

Após as definições dos pinos do Arduino, temos as definições da frequência das notas entre dó 3 e dó 4. Por exemplo:

com #define do3 261.63 temos que, toda vez que eu utilizar o termo do3, o Arduino vai entender que é 261.63.

```
// A seguir, definições para as frequências das notas na oitava 3.
#define do3 261.63
…
#define do4 523.25
```

O próximo trecho do código é a criação de duas variáveis. A primeira é do tipo *float* que permite números fracionados (com virgula) – na verdade são números com pontos pois, no inglês o uso de ponto e de virgula nos números é o contrário do que no português. A outra variável criada é do tipo *int*, para números inteiros.

```
//Inicialização da variável que faz a multiplicação para gerar as diferentes oitavas.
float oitavador = 1;
//Inicialização da variável utilizada para as "funções adicionais".
int funcaoAdicional = 0;
```

Então, o código inicia do void setup. Nesta parte, o que ocorre é apenas a indicação que os pinos das teclas são entradas do tipo pullup[26], ou seja, quando **nada** estiver ligado nos pinos respectivos o sinal estará em **5V** (*pull up* significada puxar para cima) e, quando a tecla for **pressionada**, o GND (que vem da haste metálica) é ligado no pino e ele fica em **0V**. Estes estados de 5V e de 0V é que possibilitarão a ideia de tecla solta e de tecla pressionada no teclado musical.

[26] A ideia de se utilizar este tipo de entrada *pullup* e não simplesmente a entrada normal, *INPUT*, é que com o tipo de entrada normal a parte eletrônica teria que ser mais complexa para funcionar: Teria que ser colocado um resistor entre cada tecla e o pino. Para entender este aspecto é necessário um aprofundamento na área da eletrônica, o que foge ao enfoque deste livro.

```
void setup() // O void setup é a parte do programa que é executada uma vez, logo que o Arduino é ligado.
{
  //Indicação que os pinos das teclas estão no modo INPUT_PULLUP)
  pinMode (tecla1, INPUT_PULLUP);
...
  pinMode (teclaDeFuncao, INPUT_PULLUP);
}
```

Depois de executado uma vez o void setup, o sketch entra no void loop, como segue:

```
void loop() // O void loop é a parte do programa que fica repetindo, indefinidamente.
```

Agora, para ficar mas didático, vamos pular para o final do sketch e falar sobre a tecla função. A primeira estrutura que vemos é a while. A linha completa é a seguinte:

```
while (digitalRead(teclaDeFuncao) == LOW)
```

Ou seja, enquanto[27] a leitura digital[28] do pino teclaDeFuncao for igual[29] a 0V[30], o sketch ficará fazendo apenas o que está dentro da estrutura while. Ficará preso para sempre no while, até que a condição deixe de ser atendida.

```
// A seguir, temos as ocorrências para cada tecla, quando a tecla função está ativada.
while (digitalRead(teclaDeFuncao) == LOW) // tecla de função ativada.
```

---

[27] while

[28] digitalRead

[29] ==

[30] LOW. É importante lembrar que o pino em INPUT_PULLUP fica em 0V quando ligado, ou seja, a tecla pressionada.

```
{
  delay (20);
```

Dentro do while temos diversos ifs. Cada um funcionando de forma semelhante: modificando o valor de uma variável. Os 5 primeiros, alteram o valor da variável oitavador e os três últimos, alteram a variável funcaoAdicional. Vejamos, por exemplo, a linha:

```
if (digitalRead(tecla1) == LOW) {oitavador = 0.25;}
```

Nesta linha, se[31] a leitura digital[32] do pino tecla1 for igual[33] a OV[34], a variável oitavador passará a valer 0.25. Para as demais linhas o processo é semelhante.

```
  if (digitalRead(tecla1) == LOW) {oitavador = 0.25;}
  if (digitalRead(tecla2) == LOW) {oitavador = 0.5;}
  if (digitalRead(tecla3) == LOW) {oitavador = 1;}
  if (digitalRead(tecla4) == LOW) {oitavador = 2;}
  if (digitalRead(tecla5) == LOW) {oitavador = 4;}
  if (digitalRead(tecla6) == LOW) {funcaoAdicional = 1;}
  if (digitalRead(tecla7) == LOW) {funcaoAdicional = 2;}
  if (digitalRead(tecla8) == LOW) {funcaoAdicional = 0;}
}
```

Agora, voltemos ao início do void loop para entender o que ocorre quando outra tecla, que não a tecla de função, é pressionada. Utilizaremos o exemplo da tecla 2, cujo código é o seguinte:

```
while (digitalRead (tecla2) == LOW)
```

---

31 if

32 digitalRead

33 ==

34 LOW

```
{
  tone (autofalante, re3 * oitavador);
  if (funcaoAdicional == 1)
  { delay (40);
    tone (autofalante, re3 * 0.25);
    delay (40);
  }
  if (funcaoAdicional == 2)
  { delay (30);
    tone (autofalante, re3 * 1.5);
    delay (20);
    tone (autofalante, re3 * 2);
    delay (30);
  }
}
```

Pelo que já aprendemos com a linha while (digitalRead (tecla2) == LOW); já sabemos que, enquanto o pino tecla2 estiver ligado (tecla 2 pressionada) irá ocorrer os elementos que seguem.

A linha seguinte é: tone (autofalante, re3 * oitavador);

Ou seja, o Arduino irá enviar um sinal sonoro[35] saindo do pino autofalante, a uma frequência de re3 multiplicada pela variável oitavador. Assim, se a variável oitavador estiver como 1, a frequência reproduzida será a frequência padrão de re3 multiplicada por 1. Ou seja, sem alteração de oitava.

No caso da variável oitavador estar em 0.25, a frequência resultante será 2 oitavas abaixo; se for 0.5, será, oitava abaixo; se for 2, será oitava acima; e 4, duas oitavas acima.

Se a variável funcaoAdicional estiver como qualquer número diferente de 1 ou de 2, apenas isso que foi narrado anteriormente

[35] tone

é que ocorrerá no void loop. Porém, se funcaoAdicional estiver em 1 ou em 2, ocorrerá algo a mais, como segue:

```
if (funcaoAdicional == 1)
{ delay (40);
  tone (autofalante, re3 * 0.25);
  delay (40);
}
```

Pela linha delay (40); temos que é dado um tempo de 40 milissegundos, ou seja, a nota emitida antes de entrar neste if irá ser emitida por 40 milissegundo, já que logo em seguida é emitida uma outra nota, conforme segue:

tone (autofalante, re3 * 0.25); temos que é emitida a nota da tecla, só que na oitava 1 (ou seja, duas oitavas abaixo da oitava 3, que é a padrão).

Esta segunda nota, também, dura 40 milissegundos, devido à próxima linha ser delay (40);.

Então, o void loop ocorre novamente. Ou seja, para o ouvinte, a sensação é de uma nota ser tocada e depois a mesma nota soar na oitava grave, como um pulso alternado. Isso é um **trêmulo**, na terminologia musical.

No caso de a função adicional ser 2, além da nota inicial que é tocada no início do void loop as seguintes coisas ocorrem:

```
if (funcaoAdicional == 2)
{ delay (30);
  tone (autofalante, re3 * 1.5);
  delay (20);
  tone (autofalante, re3 * 2);
  delay (30);
}
```

Um tempo de 30 milissegundos, ou seja, a nota inicial do loop durará 30 milissegundos. Depois é executada uma nota na

frequência da oitava central, mas multiplicado por 1.5. – Esta multiplicação resulta na nota uma quinta acima. Por exemplo, se a nota é o ré3 esta outra nota será lá3. Então, é dado um tempo de 20 milissegundos e uma terceira nota é emitida. No caso, é uma nota oitava acima à nota da oitava central. Por fim, tem-se uma duração de 30 milissegundos, antes que o void loop reinicie. Isso resulta em um pulso bastante interessante, sonoramente falando.

4.5 Utilizando o simulador virtual do TGL

Para ter acesso à versão do TGL no simulador utilize o seguinte link:

<https://www.tinkercad.com/things/82nq66KkwNb>

A Figura 4.19 ilustra com é esta simulação.

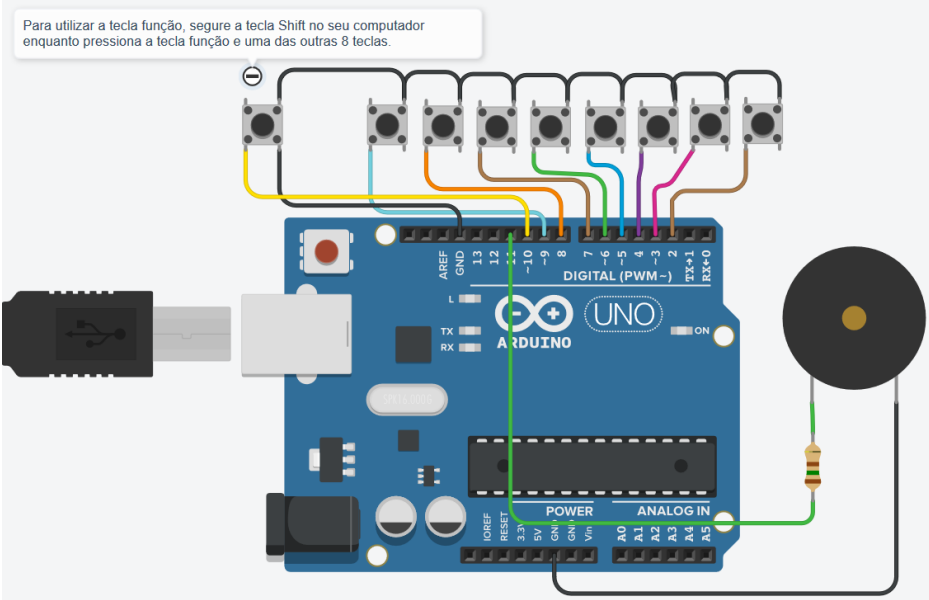


Figura 4.19 – TGL no simulador

Salienta-se que, na figura, a tecla 1 corresponde ao botão na direita, e assim por diante. Observa-se ainda que o Tinkercad não é muito bom na execução sonora, e o som sai um pouco falhado. Na figura, pode-se ainda ver a ligação dos cabos. É interessante notar que no teclado físico que foi construído a haste metálica é o equivalente aos cabos pretos que aparecem acima dos botões.

4.6 Ideias de variação na construção

Utilizando tudo o que aprendeu neste processo, experimente criar teclados de formatos e tamanhos diferentes. E se o teclado fosse todo separado? Cada tecla em uma placa de compensado separada? Pense nisso!

E se tivesse mais notas? Para isso bastaria ter mais linhas na programação para dar conta de mais portas digitais[36] e mais definições de frequência para as notas.

4.7 Ideias para uso didático

Se você deseja utilizar o que aprendeu em uma aula de música, considere as seguintes sugestões e ainda crie outras:

1-Utilizar como instrumento musical para tocar melodias.

2-Se for construído com as teclas separadas, cada aluno pode ficar responsável por acionar uma tecla, simulando a proposta de um coral de sinos, só que sem som de sino. É claro!

---

[36] Uma dica de *expert*: Na verdade ,é mentira que o Arduino Uno tem apenas 14 portas digitais, de 0 a 13. Isso, porque as portas de A0 a A5 também podem ser utilizadas como portas digitais, bastando para isso que sejam chamadas de 14 a 19. Por exemplo:

```
#define tecla10 12
#define tecla11 13
#define tecla12 14
#define tecla13 15
#define tecla14 16
#define tecla15 17
#define tecla16 18
#define tecla17 19
```

4.8 Ideias de variação na programação

Para variar a programação, experimente:

1-Utilizar outras notas para as teclas, modificando a frequência dos *#defines*.

2-Criar outros efeitos musicais nas funções adicionais.

4.9 Resumo

Neste quarto capítulo construímos um teclado musical simples. Em termos de programação vimos a simbologia para comentários longos (/* */), #define, while, noTone, digitalRead, LOW e HIGH.

5. Módulo BotPotLED Gl

Para otimizar a criação de aplicações musicais com o Arduino, foi concebido o Módulo BotPotLED Gl. Trata-se de um kit com 2 botões, (daí o *Bot*) um potenciômetro (daí o *Pot*) e 3 LEDs (daí o *LED*) elaborado pelo autor deste texto (daí Gl). Também faz parte do kit um Arduino com um autofalante e uma mini placa de ensaio. Vide as Figuras de 5.1 a 5.4 com algumas fotos do kit.

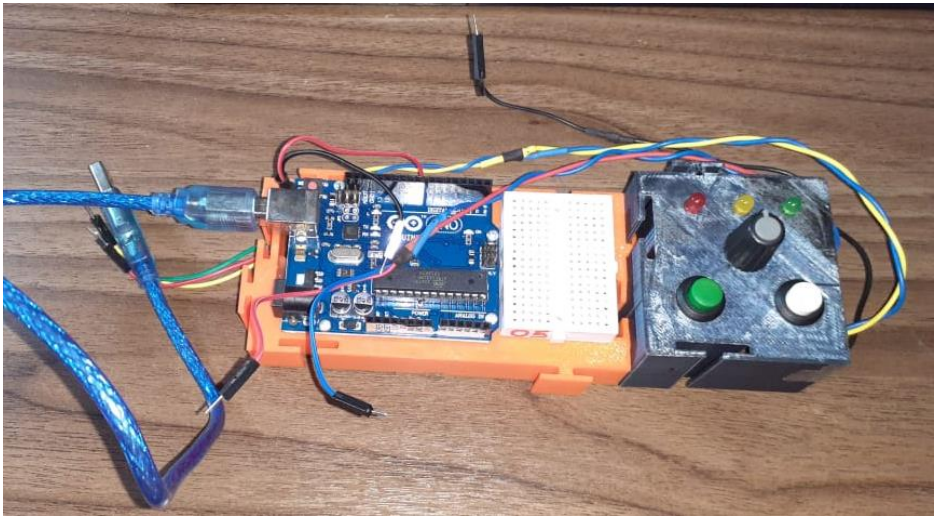


Figura 5.1 – Módulo BotPotLED Gl com os cabos soltos

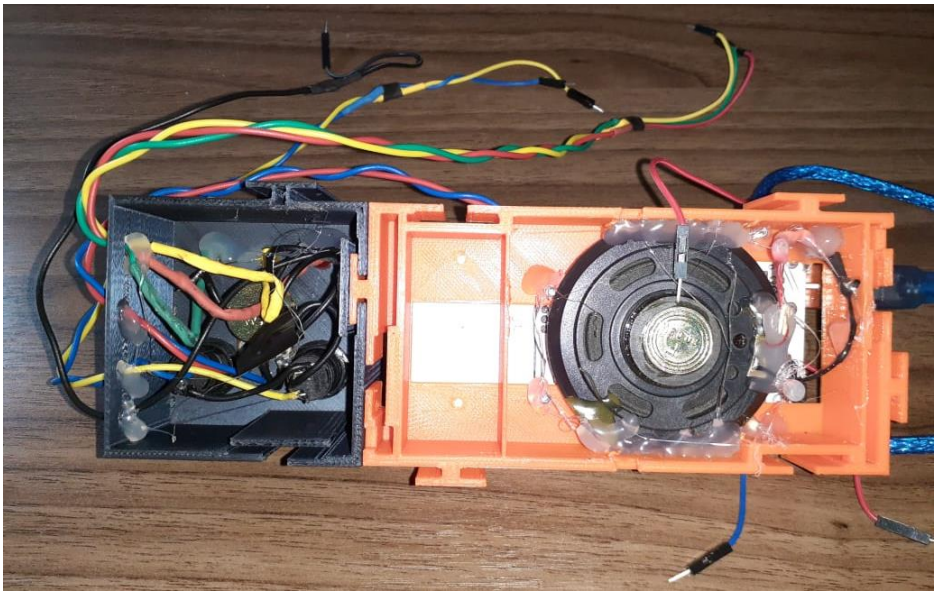


Figura 5.2 – Módulo BotPotLED Gl visto de baixo

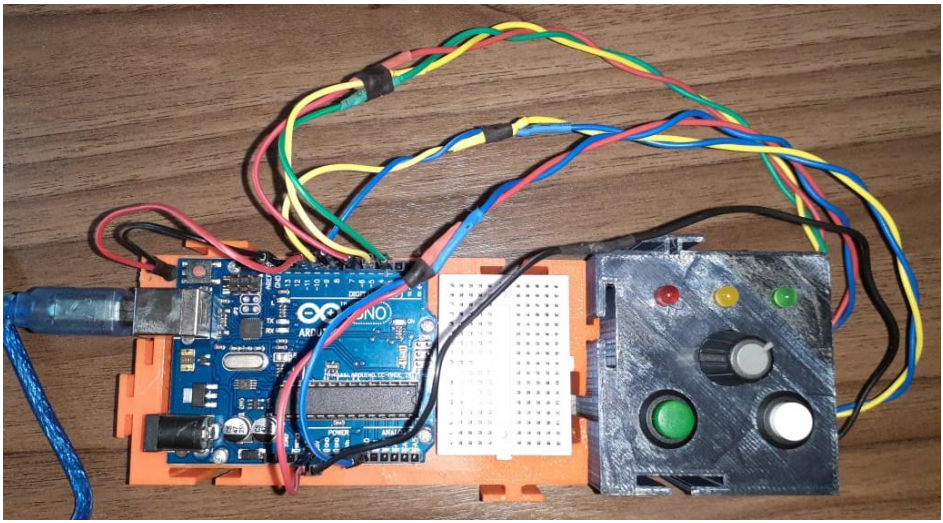


Figura 5.3 – Módulo BotPotLED Gl com os cabos conectados

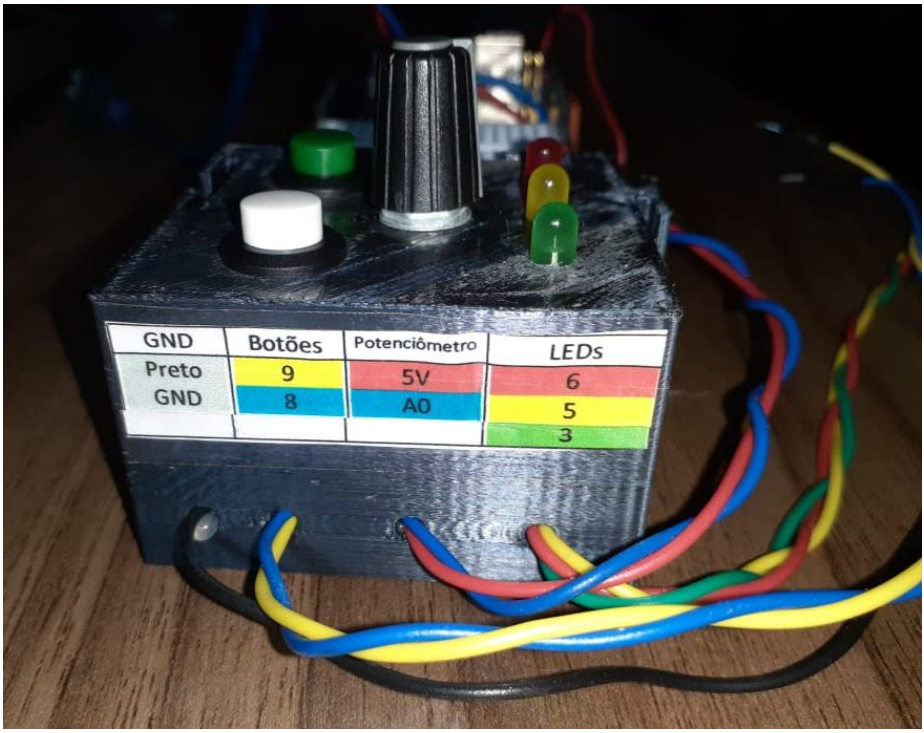


Figura 5.4 – Módulo BotPotLED GI visão lateral com legendas dos cabos

Neste capítulo não serão realizadas descrições pormenorizadas da parte computacional de programação. A ideia é apenas fornecer ao leitor algumas ferramentas práticas para uso no ensino musical. Para que a parte teórica seja compreendida será necessário que o interessado busque por cursos específicos sobre programação para Arduino. Indica-se o canal no *youtube* Brincando com Ideias. Grande parte do que aprendi para elaborar este livro veio das aulas disponibilizadas no canal.

5.1 Entendendo o circuito

Para o iniciante, pode ser confuso e difícil acompanhar toda essa explicação sobre o circuito, então não se preocupe se deixar de entender algo. Não é necessário o entendimento total de este item para poder utilizar o Módulo *BotPotLED GI* em seus experimentos.

No tinkercad foi elaborada uma versão virtual do kit que está disponibilizado neste link:

<https://www.tinkercad.com/things/9stGfKXvWci>

Os seguintes elementos estão presentes:

1 Arduino UNO.

1 autofalante, de 4 ou 8 ohms (ou um Buzzer), intermediado em um dos terminais por um resistor de 150 a 270 ohms. O outro terminal do autofalante fica ligado no GND do Arduino.

3 LEDs: LED vermelho: Positivo, intermediado por um resistor de 330 ohms, no pino 6; LED amarelo: Positivo, intermediado por um resistor de 330 ohms, no pino 5; LED verde: Positivo, intermediado por um resistor de 330 ohms, no pino 3.

1 potenciômetro. Conector do centro no pino analógico (A0) do Arduino. Os pinos laterais do potenciômetro vão para o GND e o 5v do Arduino.

2 pushbuttons entre cada pino, conforme a indicação a seguir, e o GND. Ou seja, um terminal do pushbutton no GND e o outro em pino. Conforme segue: Botão 1 no pino 9 e Botão 2 no pino 8.

No módulo, os cabos externos para conexão com o Arduino estão dispostos de forma alinhada com as partes: LEDs (3 cabos), Potenciômetro (2 cabos) e Botões (2 cabos). Além dos respectivos conjuntos de cabos, existe um cabo de terra (GND) de cor preta saindo do módulo.

Resumo da pinagem no Arduino:

0: Nada

1: Nada

2: Nada

3: Positivo do LED verde intermediado por resistor. (Cabo verde dos LEDs do Módulo BotPotLED GI)

4: Nada

5: Positivo do LED amarelo intermediado por resistor. (Cabo amarelo dos LEDs do Módulo BotPotLED GI)

6: Positivo do LED vermelho intermediado por resistor. (Cabo vermelho dos LEDs do Módulo BotPotLED GI)

7: Nada

8: Botão 2. (Cabo amarelo dos Botões do Módulo BotPotLED GI)

9: Botão 1. (Cabo azul dos Botões do Módulo BotPotLED GI)

10: Terminal do autofalante intermediado por resistor. O Resistor é dispensável se autofalante tiver mais que 8 ohms.

11: Nada

12: Nada

13: Nada

GND: Cabo negativo do autofalante. (Cabo preto do Módulo BotPotLED GI)

AREF: Nada

A5: Nada

A4: Nada

A3: Nada

A2: Nada

A1: Nada

A0: Contato central do potenciômetro. (Cabo azul do potenciômetro do Módulo BotPotLED GI)

Vin: Nada

GND: Nada

GND: Cabo Preto dos vários sinais negativos. (Cabo preto do Módulo BotPotLED GI)

5V: Positivo (um dos terminais laterais) do potenciômetro. (Cabo vermelho do potenciômetro do Módulo BotPotLED GI)

3.3V: Nada

Reset: Nada

IOREF: Nada

5.2 Materiais utilizados

Se você for elaborar um kit deste necessitará dos seguintes materiais:

- 2 push buttons
- 1 potenciômetro
- 1 LED vermelho
- 1 LED amarelo
- 1 LED verde
- 3 resistores de 220Ω
- 1 resistor de 150Ω
- 1 arduino Uno
- 1 autofalante
- 1 cabos flexíveis (ou conectores Dupont macho-macho) de cores variadas
- 1 mini placa de ensaio (opcional)
- 1 ferro de soldar, solda e outros acessórios para a soldagem (opcional)

Para o kit também foram modeladas partes plásticas em impressora 3D. Para ter acesso a essas modelagens vide os seguintes links:

https://www.tinkercad.com/things/gqlfBbVMcaw

https://www.tinkercad.com/things/2RFEbfzop7H

Se você já tem acesso a um dos kits, basta realizar as conexões conforme indicadas na legenda nas laterais, como as ilustradas nas Figuras 5.5 e 5.6.

Figura 5.5 – Adesivos indicativos nas laterais do Arduino

GND	Botões	Potenciômetro	LEDs
Preto	9	5V	6
GND	8	A0	5
			3

Figura 5.6 – Adesivo indicativo na lateral do BotPotLED GI

Depois, é só partir para a prática, criando sketches conforme indicados nos itens seguintes ou conforme sua criatividade.

5.3 Tocador de pulso com escala cromática

Trata-se de um instrumento musical. As notas, escolhidas pelo manejo do potenciômetro, são emitidas em pulsos de duração variável, conforme o controle do usuário.

Assista ao vídeo chamado *Demonstração do Tocador de pulso com escala cromática*, no seguinte link:

<https://youtu.be/0_1vmb61hGg>

Para carregar o programa no Arduino, baixe o arquivo *01_TocaPulsoEscalaCromatica.zip* que está na página do projeto no site da comunidade virtual do Arduino, no seguinte link:

<https://create.arduino.cc/projecthub/GlauberSantiago/modulo-botpotled-gl-cf504b>

Depois de baixar e descompactar toda a pasta, abra o arquivo *01_TocaPulsoEscalaCromatica.ino* que o Arduino já vai abrir os demais arquivos necessários para o funcionamento do programa. Observe que, diferentemente do que ocorria nos capítulos anteriores do livro, aqui são utilizados múltiplos arquivos para o Sketch.

A Figura 5.7 mostra como aparece a pasta, já descompactada, e seus arquivos no momento de abertura do sketch na IDE do Arduino.

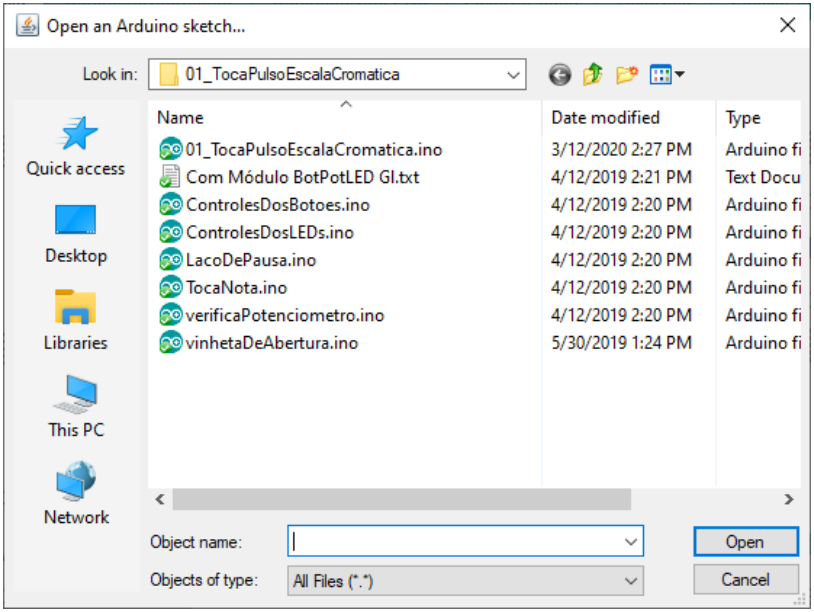


Figura 5.7 – Arquivos da pasta utilizados pelo sketch

Após abrir o arquivo cujo nome é o mesmo da pasta, o sketch aparecerá conforme Figura 5.8. Observe que aparecem diversas abas, correspondentes aos vários arquivos com as partes do sketch que estão na pasta.

Figura 5.8 – Aparência do sketch com múltiplos arquivos

Carregue o código no Arduino, normalmente.

Ao ser ligado o Arduino, é emitida uma vinheta de abertura visual e sonora. Ao final da vinheta, é emitido o intervalo melódico de primeira justa, indicando 1, que representa o presente sketch elaborado para o Módulo *BotPotLED Gl*.

Após a abertura, o sistema fica em modo "pausa", com os LEDs oscilando sua luminosidade e sem nenhum som sendo emitido.

Ao se pressionar o botão 1 (o da esquerda) o modo "pausa" é interrompido. Então, conforme a posição do potenciômetro, é emitida uma nota da escala cromática temperada em um pulso constante.

A duração do pulso é modificada com os botões. O botão 1 aumenta da duração do pulso (fica mais lento) e o Botão 2 diminui a sua duração (ficando mais rápido).

Para se retornar ao modo "pause" é necessário o acionamento simultâneo dos botões 1 e 2.

Resumindo os recursos:

- LED vermelho -> No modo pausa, fica piscando de forma intermitente. Fora do modo pausa, indica o acionamento do botão 1.
- LED amarelo -> No modo pausa, fica piscando de forma intermitente. Fora do modo pausa, indica o início de cada pulso.
- LED verde -> No modo pausa, fica piscando de forma intermitente.
- Potenciômetro -> Varia a nota musical emitida.
- Botão 1 -> Aumenta a duração do pulso ou sai do modo "pausa".
- Botão 2 -> Diminui a duração do pulso ou sai do modo "pausa".

5.3.1 Ideias para uso didático

1-Duelo de músicos em uníssono: Um usuário seleciona uma nota para ser tocada, então, outro deve buscar tocar a mesma nota em outro Módulo (ou em outro instrumento musical).

2-Duelo de músicos em terça maior: Um usuário seleciona uma nota para ser tocada, então outro deve buscar tocar a nota uma terça maior acima em outro Módulo.

3-Uso como instrumento musical em circunstância de apreciação musical relativas a frequência, andamento, duração e timbres sintetizados.

5.3.2 Ideias de variação na programação

A-Modificar as variáveis "notaMaisGrave" e "notaMaisAguda".

1. Procure (ctrl+f) por "notaMaisGrave =" e por "notaMaisAguda =".
2. Modifique os valores. O número 60 é o dó3 (dó central). 61 é o dó# e assim por diante. Outros valores são: 96 = dó6; 84 = dó5; 72 = dó4; 60 = dó3 (central); 48 = dó2; 36 = dó1; etc.

B-Modificar a quantidade de notas diferentes emitidas por pulso, oitavando, por exemplo.

1. Procure por "void loop".
2. Acrescente mais comandos como:

```
tone(pinoDoAutofalante, 440 * pow(2.0, (nota - 57) / 12.0), duracao);
delay (duracao);
```

Na linha do *tone*, modifique o número **57** para cima ou para baixo para mudar a nota. + 12 fica oitava acima e -12, fica oitava abaixo.

Em "duracao" você pode seguir as seguintes diretrizes com indicações de durações para o andamento, ♩=100.

- Semibreve: 2400
- Mínima: 1200
- Semínima: 600
- Colcheia: 300
- Semicolcheia: 125

C-Executar uma tríade arpejada no lugar de uma nota apenas.

1. Siga os mesmos passos do item B, anterior.

D-Criar motivo rítmico para ser executado em cada pulso.

1. Siga os mesmos passos do item B, anterior.

E-Mudar melodia da vinheta de abertura.

1. Procure por "void vinhetaAbertura".
2. Siga os mesmos passos do item B2, anterior.

5.4 Tocador de pulso com frequências

Este sketch se trata de um instrumento musical. As frequências escolhidas pelo manejo do potenciômetro são emitidas em pulsos de duração variável, conforme o controle do usuário.

Assista ao vídeo chamado *Demonstração do Tocador de pulso com frequências*, no seguinte link:

<https://youtu.be/PH8UzGnKOL8>

Para usar, baixe o arquivo *02_TocaPulsoFrequencias.zip* que está na página do projeto, no seguinte link:

<https://create.arduino.cc/projecthub/GlauberSantiago/modulo-botpotled-gl-cf504b>

Depois de baixar e descompactar toda a pasta, abra o arquivo *02_TocaPulsoFrequencias.ino* que o Arduino já vai abrir os demais arquivos necessários para o funcionamento do programa. Carregue o sketch no Arduino e bom uso!

Ao ser ligado o Arduino, é emitida uma vinheta de abertura visual e sonora. Ao final da vinheta, é emitido o intervalo melódico de segunda maior, indicando "2", que representa o presente sketch elaborado para o Módulo BotPotLED Gl.

Após a abertura, o sistema fica em modo "pausa", com os LEDs oscilando sua luminosidade e sem nenhum som sendo emitido.

Ao se pressionar o botão 1 (o da esquerda), o modo "pausa" é interrompido. Então, conforme a posição do potenciômetro, é emitida uma nota na frequência em um pulso constante.

A duração do pulso é modificada com os botões. O botão 1 aumenta da duração do pulso (fica mais lento) e o botão 2 diminui a sua duração (ficando mais rápido).

Para se retornar ao modo "pausa" é necessário o acionamento simultâneo dos botões 1 e 2.

Resumindo os recursos:

- LED vermelho -> No modo pausa, fica piscando de forma intermitente. Fora do modo pausa, indica o acionamento do botão 1.
- LED amarelo -> No modo pausa, fica piscando de forma intermitente. Fora do modo pausa indica o início de cada pulso.
- LED verde -> No modo pausa, fica piscando de forma intermitente.
- Potenciômetro -> Varia a nota musical emitida.
- Botão 1 -> Aumenta a duração do pulso ou sai do modo "pausa".
- Botão 2 -> Diminui a duração do pulso ou sai do modo "pausa".

5.4.1 Ideias para uso didático

1-Duelo de músicos em uníssono: Em um módulo, ou um outro instrumento, um usuário seleciona uma frequência (ou nota) para ser tocada, então outro deve buscar tocar a mesma frequência em outro Módulo.

2-Duelo de músicos em oitava: Em um módulo, ou um outro instrumento, um usuário seleciona uma frequência (ou nota) para ser tocada, então outro deve buscar tocar a mesma frequência uma oitava acima, em outro Módulo.

3-Uso como instrumento musical em circunstância de apreciação musical relativas à frequência (incluindo batimentos), andamento, duração e timbres sintetizados.

5.4.2 Ideias de variação na programação

A-Modificar as variáveis "frequenciaMaisBaixa" e "frequenciaMaisAlta".

1. Procure (ctrl+f) por "frequenciaMaisBaixa =" e por "frequenciaMaisAlta =".
2. Modifique os valores entre 40 e 5000, por exemplo. Para ver os valores das frequências das notas musicais vide o link: https://sites.google.com/site/aprendamusicacomigo/musica-e-tecnologia

B-Modificar a quantidade de notas diferentes emitidas por pulso, oitavando.

1. Procure (ctrl+f) "void loop".
2. Procure pelas linhas a seguir:

```
tone(pinoDoAutofalante, resultado, duracaoDoPulso - 20);
digitalWrite (pinoDoLEDAmarelo, HIGH);
delay (duracaoDoPulso * 0.5);
```

3. Após esta última acrescente:

```
tone(pinoDoAutofalante, resultado, duracaoDoPulso - 20);
delay (duracaoDoPulso * 0.5);
```

4. Para ficar oitava acima, multiplique por 2 o resultado. Para oitava abaixo, multiplique por 0.5.
5. Modifique o multiplicador da duracaoDoPulso para outras durações.

C-Criar motivo rítmico para ser executado em cada pulso.

1. Siga os mesmos passos do item B, anterior.

E-Mudar melodia da vinheta de abertura.

1. Siga os mesmos passos do item B2, anterior.

5.5 Gerador melódico

Trata-se de um gerador de melodias para serem executada sem loop (repetição). São propostas 20 células rítmicas pré concebidas, em compasso 4/4, a serem escolhidas pelo usuário. Além disso, são fornecidas 20 sequencias de notas, também selecionadas pelos usuários. Ao ser iniciada a sequência, conforme a rítmica e sequência de notas escolhida, o usuário pode controlar o andamento do loop. As frequências escolhidas pelo manejo do potenciômetro são emitidas em pulsos de duração variável, conforme o controle do usuário.

Assista ao vídeo chamado *Demonstração do Gerador melódico*, no seguinte link:

<https://youtu.be/hRe0lI5RNeQ>

Para instalar, baixe o arquivo *03_GeradorMelodico.zip* que está na página do projeto, no seguinte link:

<https://create.arduino.cc/projecthub/GlauberSantiago/modulo-botpotled-gl-cf504b>

Depois de baixar e descompactar toda a pasta, abra o arquivo *03_GeradorMelodico.ino* que o Arduino já vai abrir os demais arquivos necessários para o funcionamento do programa. Carregue o sketch no Arduino e bom uso!

Ao ser ligado o Arduino, é emitida uma vinheta de abertura visual e sonora. Ao final da vinheta, é emitido o intervalo melódico de terça maior, indicando "3", que representa o presente sketch elaborado para o Módulo BotPotLED Gl.

Após a abertura, o sistema fica em modo "pausa" para que se realize a seleção da rítmica e das notas. Neste caso, os LEDs ficam oscilando sua luminosidade e sem nenhum som sendo emitido.

No modo de "pausa" (seleção), quando o potenciômetro está totalmente para a ESQUERDA, ao se pressionar o botão 1 (o da esquerda) a rítmica é selecionada. Semelhantemente, no modo de "pausa" (seleção), quando o potenciômetro está totalmente para a DIREITA, ao se pressionar o botão 2 (o da direita) a sequência melódica é selecionada.

Em ambos os casos acima a identificação do item é dada por um intervalo musical melódico emitido. Para os números de 1 a 10 temos intervalos desde o uníssono até a décima. Para os números de 11 a 20 temos também intervalos de uníssono até a décima, só que, desta vez, tudo ocorre oitava acima.

Para se iniciar o modo de execução, o potenciômetro deve estar na posição central e em seguida, qualquer um dos botões deve ser pressionado.

No modo de execução, utilize o potenciômetro para variar o andamento. Além disso, clique em qualquer um dos botões para voltar ao modo "pausa" (seleção).

Obs.: A cada início de compasso, célula rítmica, é emitido um sinal visual com o LED amarelo.

Resumindo os recursos:

- LED vermelho -> No modo pausa, fica piscando de forma intermitente. Fora do modo pausa, indica o acionamento do botão 1.
- LED amarelo -> No modo pausa, fica piscando de forma intermitente. Fora do modo pausa, indica o início de cada compasso.
- LED verde -> No modo pausa, fica piscando de forma intermitente.
- Potenciômetro -> No modo pausa, para a esquerda habilita a seleção da rítmica pelo botão 1 e para a direita habilita a seleção da sequência de notas pelo botão 2. Na posição central, habilita a saída do modo de pausa mediante o acionamento de qualquer dos botões.
- Botão 1 -> No modo "pausa", quando o potenciômetro está para a esquerda, seleciona a rítmica e quando o potenciômetro está no centre, sai do modo de seleção. No modo "execução", retorna para o modo pausa.
- Botão 2 -> No modo "pausa", quando o potenciômetro está para a direita, seleciona a sequência de notas e quando o potenciômetro está no centro, sai do modo de seleção. No modo "execução", retorna para o modo pausa.

5.5.1 Ideias para uso didático

1-Ditados rítmicos e melódicos.

2-Apreciação musical e explanação de conceitos sobre andamento, rítmica, solfejo e timbres sintetizados para os estudantes.

5.5.2 Ideias de variação na programação

A-Modificar as células rítmicas propostas.

B-Modificar as sequências de notas propostas.

C-Modificar a quantidade de notas diferentes emitidas por pulso, oitavando.

D-Criar motivo rítmico para ser executado em cada pulso.

E-Mudar melodia da vinheta de abertura.

5.6 Resumo

Neste capítulo aprendemos a utilizar um circuito com autofalante, LEDs, resistores, potenciômetro e botões. Em termos de computação vimos como novidade a existência de sketchs com vários arquivos na pasta e sobre a necessidade de aprofundamento nos estudos para se entender a parte teórica da programação destas últimas aplicações.

6. Palavras finais

Este breve livro objetivou incentivar ao professor de música, leigo em computação e eletrônica, a adentrar nestas áreas de conhecimento de forma prática. Quem deseja se aprofundar deve buscar literatura adequada e videoaulas na internet sobre programação, eletrônica básica, microcontroladores, acústica e tecnologia musical.

Indica-se a seguinte *playlist* de vídeos do mesmo autor deste texto com dezenas de experimentos com Arduino, relativamente fáceis de serem executados pelos interessados. O link é o seguinte:
<https://www.youtube.com/watch?v=xMKfvw1J1Vk&list=PLRd7fQHMi_x667oswGrTxPR0ElZvown0a>

Veja, a seguir, algumas imagens ilustrativas e informações dos vídeos:

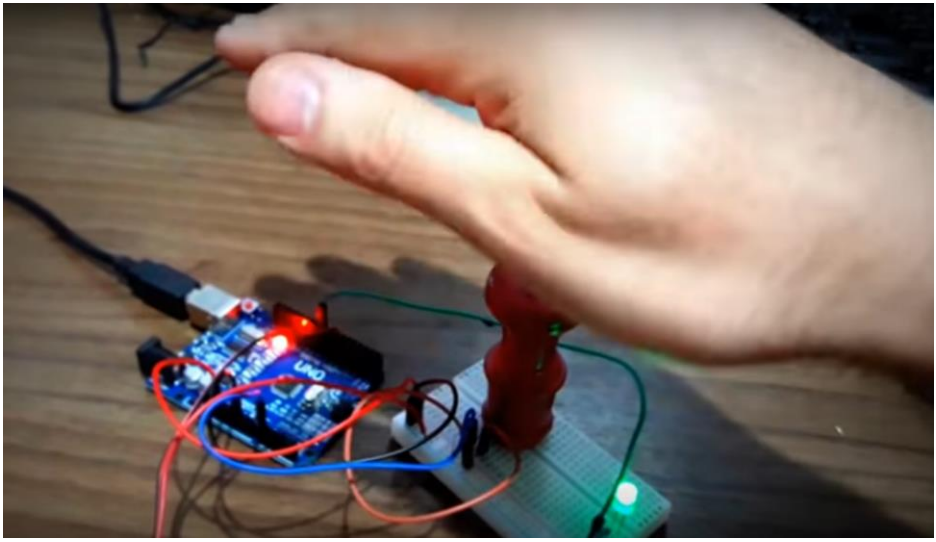


Figura 6.1 – Notas variando por meio de sensor de luminosidade

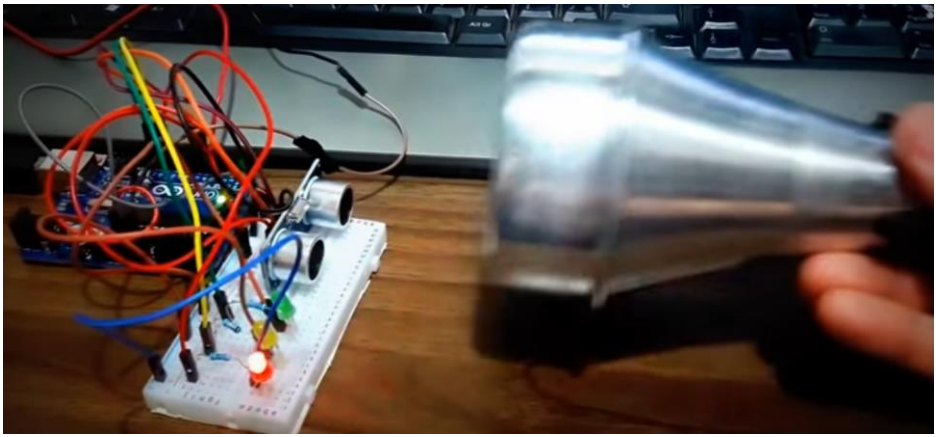


Figura 6.2 – Notas variando por meio de sensor de proximidade

Figura 6.3 – Percussionista autômato

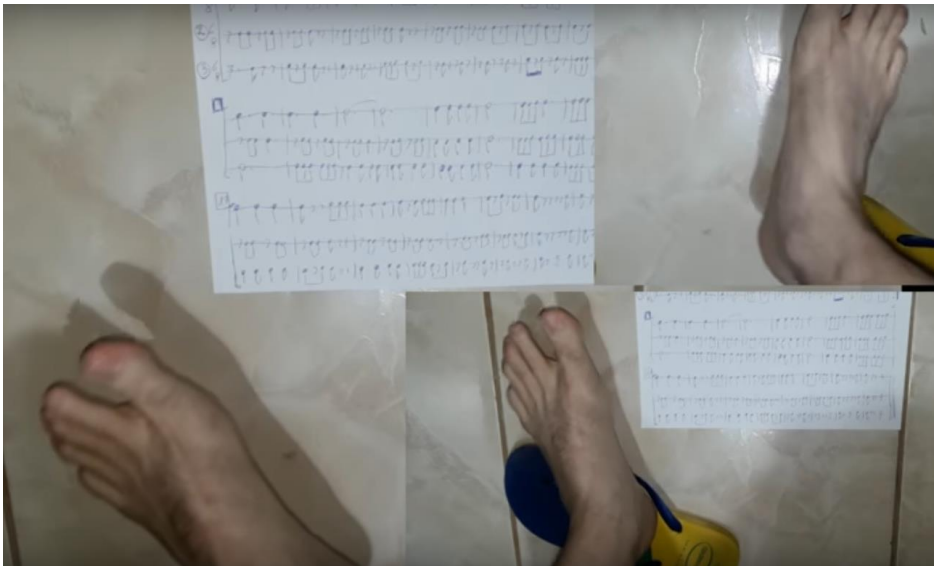


Figura 6.4 – Notas emitidas com os pés no chão

Figura 6.5 – Sensor de proximidade automotivo para músicos

Figura 6.6 – Pandeiro autômato

Figura 6.7 – Metrônomo visual e sonoro de um metro

Figura 6.8 – Tocador de várias formas

Figura 6.9 – Tocar notas usando frutas

Figura 6.10 – Maestro robótico

6.1 Indicações para aprofundamento

Para seu aprofundamento, indica-se alguns canais do Youtube:

Brincando com Ideias

https://www.youtube.com/channel/UCcGk83PAQ5aGR7IVlD_cBaw

Notes and Volts

https://www.youtube.com/user/NotesAndVolts

WR Kits

https://www.youtube.com/user/canalwrkits

Sobre bibliografias, indica-se estes livros:

BLUM, Jeremy. *Explorando o Arduino*: Técnicas e ferramentas para mágicas de engenharia. Rio de Janeiro: Alta books. 2016.

MONK, Simon. *30 Projetos com Arduino.* Porto Alegre: Bookman, 2014.

7 Apêndice

7.1 Definições de frequências das notas musicais

A seguir, é apresentado um quadro com as frequências das notas musicais no sistema temperado, afinadas com o Lá padrão em 440Hz. No início de cada linha, existe o nome da nota correspondente àquela linha. A cada coluna é considerada uma oitava, conforme a indicação da última linha. Observe que a notação das oitavas está no sistema padrão (EUA, Standard). No sistema franco-belga, que é o mais utilizado no Brasil, a oitava é um número a menos. Ou seja, por exemplo, o dó central é o dó 3, e não dó 4, como aparece no quadro.

Observe ainda que o quadro apresenta um fundo variando entre branco e cinza, lembrando as teclas brancas e pretas do piano. Além disso, a oitava central está grafada na cor vermelha.

Dó	16.3496	32.6992	65.3984	130.7968	261.5936	523.1872	1046.3744	2092.7488	4185.4976
Réb	17.3142	34.6284	69.2568	138.5136	277.0272	554.0544	1108.1088	2216.2176	4432.4352
Ré	18.3443	36.6886	73.3772	146.7544	293.5088	587.0176	1174.0352	2348.0704	4696.1408
Mib	19.4397	38.8794	77.7588	155.5176	311.0352	622.0704	1244.1408	2488.2816	4976.5632
Mi	20.6005	41.2010	82.4020	164.8040	329.6080	659.2160	1318.4320	2636.8640	5273.7280
Fá	21.8267	43.6534	87.3068	174.6136	349.2272	698.4544	1396.9088	2793.8176	5587.6352
Fá#	23.1183	46.2366	92.4732	184.9464	369.8928	739.7856	1479.5712	2959.1424	5918.2848
Sol	24.4917	48.9834	97.9668	195.9336	391.8672	783.7344	1567.4688	3134.9376	6269.8752
Láb	25.9468	51.8936	103.7872	207.5744	415.1488	830.2976	1660.5952	3321.1904	6642.3808
Lá	27.5000	55.0000	110.0000	220.0000	440.0000	880.0000	1760.0000	3520.0000	7040.0000
Sib	29.1350	58.2700	116.5400	233.0800	466.1600	932.3200	1864.6400	3729.2800	7458.5600
Si	30.8680	61.7360	123.4720	246.9440	493.8880	987.7760	1975.5520	3951.1040	7902.2080
Oitava	0	1	2	3	4	5	6	7	8

7.2 Frações para os intervalos musicais

O quadro a seguir apresenta frações que representam os intervalos referentes às notas da escala cromática. São valores aproximados, considerando a escala temperada. Está com o exemplo da escala de dó, mas serve para qualquer escala. Para utilizar, basta multiplicar as frações pela frequência da nota inicial. Por exemplo, se a nota inicial for o lá 440Hz, para descobrir as notas da escala cromática de lá, basta ir multiplicando cada fração por 440. Então, sib seria 440 * (16/15) = 469. Mas não necessita fazer a conta, deixe o Arduino pensar um pouco por você.

Nota da escala cromática	1	2	3	4	5	6	7	8	9	10	11	12	13
Intervalos em Semitons	0	1	2	3	4	5	6	7	8	9	10	11	12
Qualificação do intervalo	uni.	2ªm	2ªM	3ªm	3ªM	4ªJ	4ªA	5ªJ	5ªA	6ªM	7ªm	7ªM	8ªJ
Exemplo em dó	Dó	Dó#	Ré	Mib	Mi	Fá	Fá#	Sol	Sol#	Lá	Sib	Si	Dó'
Fração para multiplicar	1	16/15	9/8	6/5	5/4	4/3	10/7	3/2	8/5	5/3	16/9	15/8	2

Para o uso no Arduino, só é necessário ter cuidado pois, se estes valores forem atribuídos a variáveis, deve ser utilizada a variável do tipo *float,* uma vez que estes números são quebrados.

7.3 Fórmula para encontrar a duração das figuras musicais

Para saber a duração do tempo musical (beat, em inglês) em segundos, utilize a seguinte fórmula:

Duração de um tempo musical = 60/Andamento

Para transformar a duração em segundos para milissegundos, basta multiplicar por 1000.

Por exemplo: No andamento ♩=100 ficaria assim:

Duração do tempo musical = 60/100 = 0,6 segundos = 600 milissegundos.

Para saber a duração das figuras musicais em compassos simples tendo como padrão a duração de um tempo musical,

basta multiplicar a duração do tempo pelo valor da figura musical, conforme o seguinte quadro:

Figura musical	Multiplicador
Semibreve	4
Mínima	2
Tercina de mínima	1.3333
Semínima	1
Tercina de semínima	0.6666
Colcheia	0.5
Tercina de Colcheia	0.3333
Semicolcheia	0.25
Tercina de semicolcheia	0.1666
Fusa	0.125
Tercina de fusa	0.0833
Semifusa	0.0625
Tercina de semifusa	0.0416

Por exemplo: No andamento ♩=100 a semifusa dura 600 * 0.625 que é 37.5 milissegundos.

Para o uso no Arduino, só é necessário ter cuidado pois, se estes valores forem atribuídos a variáveis, deve ser utilizada a variável do tipo *float,* uma vez que estes números são quebrados.

7.4 Principais estruturas e funções computacionais utilizadas neste livro

A seguir são apresentadas as principais estruturas e funções computacionais utilizadas neste livro.

Elemento e parâmetros	Para que serve
tone (pino**,** frequência**)** ou **tone (**pino**,** frequência**,** duração**)**	Emite um sinal sonoro (onda quadrada) pelo pino indicado, na frequência (em hertz) indicada e (se indicado, na duração (sem milissegundos). Exemplo: tone (11, 440, 1000);
noTone (pino)	Desliga o som emitido por uma porta. Exemplo: noTone (11);
delay (duração**)**	Para o código conforme a duração indicada, em milissegundos. Exemplo: delay (1000);
pinMode (pino**,** modo**)** Os modos podem ser: INPUT, INPUT_PULLUP ou OUTPUT	Indica o modo de funcionamento dos pinos digitais conforme um dos 3 modos. Exemplo: pinMode (6, INPUT_PULLUP);

digitalRead(pino**)**	Lê se o pino digital está em HIGH ou em LOW. Exemplo: digitalRead(6);
digitalWrite (pino**,** valor**)**	Envia um sinal de 5 Volts (HIGH) ou de 0 Volts (LOW) pelo pino indicado. Exemplo: digitalWrite (8, LOW);
random (valor mínimo**,** valor máximo**)**	Escolhe um valor qualquer entre os valores indicados. Exemplo: int a = random (220, 440);
pow(base**,** expoente**)**	Faz o cálculo da potência conforme a base e o expoente indicados). Exemplo: tone(11, 440 * pow(2.0, (nota - 57) / 12.0), duracao);
int nomeDaVariavel = valor	Declara (cria) uma variável que é um número inteiro e lhe atribui um valor. Exemplo: int minhaVariavel = 1;
float nomeDaVariavel = valor	Declara (cria) uma variável com ponto flutuante (números quebrados) e lhe atribui um valor. Exemplo: float minhaVariavel = 1.5;
#define nomeDaConstante valor	Define o valor de uma constante para ser utilizada no sketch. Cuidado, pois, ela não utiliza a igualdade como em *int* e em *float* e nem é seguida de ponto e vírgula. Exemplo: #define do3 261.63
LOW	Indica que um valor digital está em nível lógico baixo, ou seja, 0 volts. Pode ser substituído pelo valor 0. Exemplo: digitalWrite (8, LOW); ou digitalWrite (8, 0);
HIGH	Indica que um valor digital está em nível lógico alto, ou seja, 5 volts. Pode ser substituído pelo valor 1. Exemplo: digitalWrite (8, HIGH); ou digitalWrite (8, 1);
void setup () { *código* **}**	Inicia uma parte do código que é executada assim que o Arduino é iniciado. Exemplo: void setup () { *aqui vem o código desejado* }
void loop () { *código* **}**	Inicia uma parte do código que é executada em loop, logo após o void setup. Exemplo: void loop () { *aqui vem o código desejado* }
if (*condição***) {** *código* **}**	Verifica uma condição e, **se** ela for atendida, realiza um código entre chaves. Exemplo: if (funcaoAdicional == 0) {oitavador = 3;}
when (*condição***) {** *código* **}**	Verifica uma condição e, **quando** ela for atendida, realiza um código entre chaves. Exemplo: when (minhaVariavel == 6) {tone (11, 440);}
//	Inicia um comentário de linha. Exemplo: // Este é um comentário que deve ficar todo na mesma linha.
/*	Inicia um trecho de comentário. Exemplo: /* Aqui iniciou-se um trecho de comentário que continuará.
***/**	Finaliza um trecho de comentário iniciado. Exemplo: Aqui está terminando um comentário /*
;	Indica a finalização de uma linha de código. Exemplo: tone (11, 440, 1000);
{ }	As chaves marcam o início e o fim de um trecho específico do código nos voids, ifs etc. Exemplo: void setup { tone (11, 440, 100); }
Operadores de comparação: **!=** *diferente de* **<** *menor que* **<=** *menor que ou igual a* **==** *igual a* **>** *maior que* **>=** *maior que ou igual a*	Estes operadores são utilizados para realizar comparações matemáticas: Exemplo: if (funcaoAdicional == 0) {oitavador = 3;}

Sobre o autor

O prof. Dr. Glauber Santiago é bacharel em Direito, mestre em Engenharia de Produção com dissertação voltada para a gestão da qualidade em organizações musicais e doutor, também, em Engenharia de Produção com tese versando sobre uma proposta de diagnóstico das competências do educador musical em projetos de curso de graduação. Atua na área musical como professor do Departamento de Artes e Comunicação (DAC) da Universidade Federal de São Carlos (UFSCar) nas áreas de percepção e notação musical, linguagem e estruturação musical, criação musical, entre outras. Musicalmente também atua como arranjador, compositor e produtor musical. Em sua produção de material didático destacam-se alguns livros e métodos para o ensino de teclado, flauta doce, Xilofone Orff, e introdução à Harmonia Tradicional; ambientes virtuais de aprendizagem; vídeo aulas sobre música; e diversos softwares relativos ao aprendizado musical. Como pesquisador tem atuado com temas voltados para a educação musical, tecnologias e EaD. É líder do grupo de pesquisa *Tecnologias aplicadas ao ensino e aprendizagem musical* e vice-líder do *Grupo de Estudos e Pesquisas sobre Inovação em Educação, Tecnologias e Linguagens* (*Grupo Horizonte*). Atualmente também é coordenador da Coordenadoria de Inovações em Tecnologias na Educação (CITE) da Secretaria Geral de Educação a Distância da UFSCar (SEaD).

Link para o currículo Lattes:
http://lattes.cnpq.br/9385173410103898

Site:
https://sites.google.com/site/aprendamusicacomigo/

Canal do YouTube:
https://goo.gl/1MDWo0