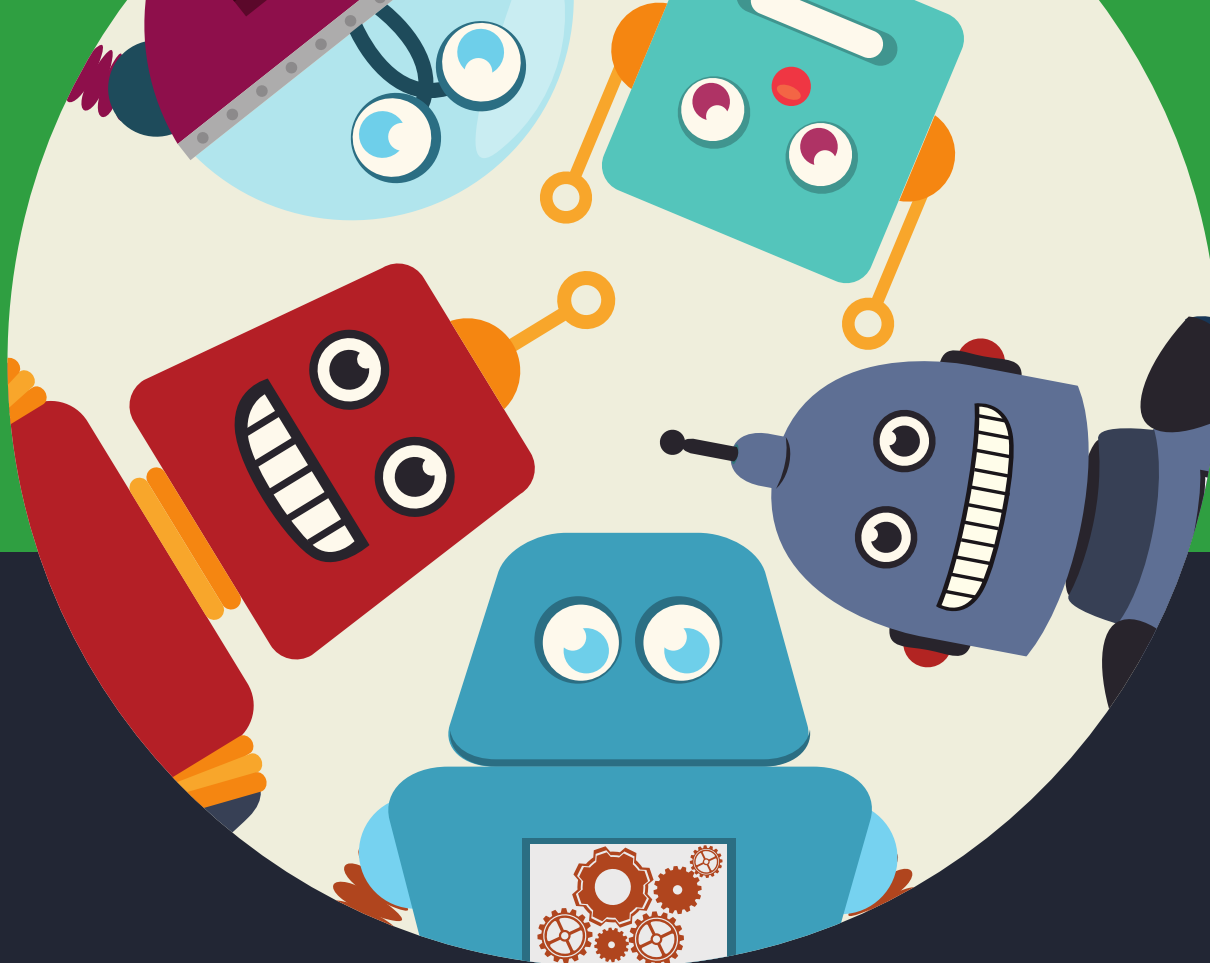




ROBÓTICA EDUCACIONAL

*Uma abordagem prática no ensino de lógica
de programação*



INSTITUTO
FEDERAL
Tocantins



PROFEPT
PROGRAMA DE PÓS-GRADUAÇÃO EM
EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

Paulo Ricardo da Silva Pontes
Valci Ferreira Victor

ROBÓTICA EDUCACIONAL



*Uma abordagem prática no ensino de lógica
de programação*

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Pontes, Paulo Ricardo da Silva

Robótica educacional : uma abordagem prática no ensino de lógica de programação [livro eletrônico] / Paulo Ricardo da Silva Pontes, Valci Ferreira Victor. -- 1. ed. -- Araguaína, TO : Instituto Federal Tocantins, 2021.

PDF

Bibliografia

ISBN 978-65-00-18540-9

1. Educação 2. Lógica - Estudo e ensino
3. Programação (Computadores) - Estudo e ensino
4. Robótica - Estudo e ensino - Inovações tecnológicas 5. Tecnologia educacional I. Victor, Valci Ferreira. II. Título.

21-58727

CDD-372.358

Índices para catálogo sistemático:

1. Robótica : Estudo e ensino 372.358

Maria Alice Ferreira - Bibliotecária - CRB-8/7964

Agradecimentos

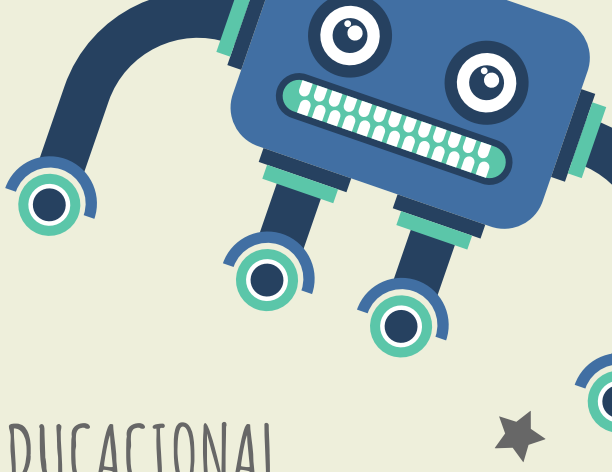
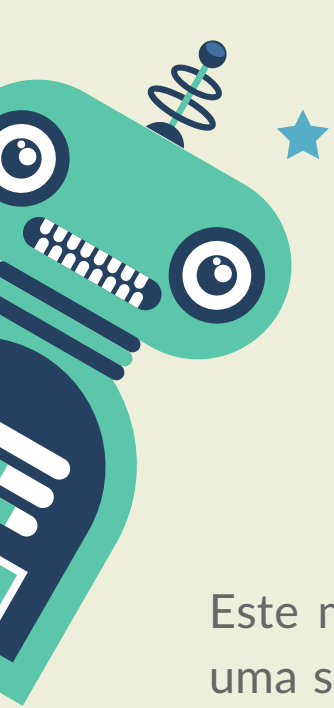
Primeiramente, quero agradecer a Deus por esse trabalho e aos meus pais Ireman Alves Pontes e Aliene da Silva Pontes, aos meus irmãos Fernando da Silva Pontes e Israel da Silva Pontes.

Em especial, agradecer minha esposa Heid Karla Pereira de Oliveira e aos meus filhos João Ricardo de Oliveira Pontes e Maria Antônia de Oliveira Pontes, pelos seus carinhos e companheirismos.

Quero agradecer ao meu orientador professor Dr. Valci Ferreira Victor e ao curso de pós-graduação em Educação Profissional e Tecnológica do Instituto Federal do Tocantins e todos os professores e colegas da turma de 2019 que fizeram parte do programa. Agradecer também ao Instituto Federal do Tocantins pelo apoio financeiro com o programa de qualificação aos servidores.

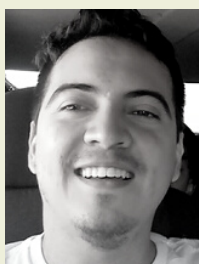
Quero agradecer aos professores e colegas Gelson Schneider e José Roberto Cruz e Silva pelo apoio na avaliação deste trabalho e pelas valiosas dicas.

Muito obrigado!



APRESENTAÇÃO DO PRODUTO EDUCACIONAL

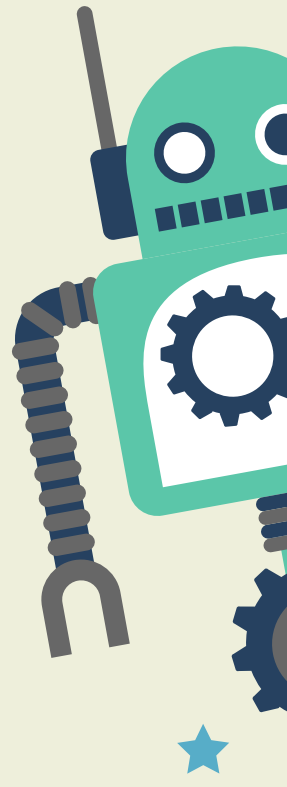
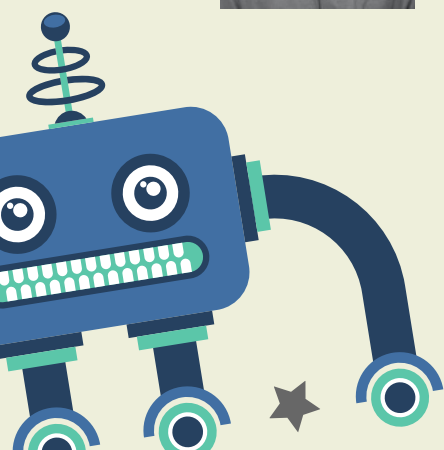
Este material tem como proposta o desenvolvimento de uma sequência didática para auxiliar no ensino de Lógica de programação. Para isso, foram desenvolvidas etapas de ensino de algoritmos com o uso de componentes eletrônicos, para um aprendizado mais voltado para a prática e contextualização do ensino. Portanto, o material busca relacionar de forma significativa os conhecimentos vistos pelos alunos.



Paulo Ricardo da Silva Pontes mestrando do programa de pós-graduação em Educação Profissional e Tecnológica no IFTO, campus Palmas e atualmente professor do ensino básico, técnico e tecnológico do Instituto Federal do Tocantins, campus Araguaína.



Valci Ferreira Victor professor doutor no programa de pós-graduação em Educação Profissional e Tecnológica no Instituto Federal de Educação, Ciência e Tecnologia do Tocantins, campus Palmas.



Sumário

1 – Introdução.....	7
2 – Robótica Educacional.....	9
3 – Plataforma Arduino.....	11
3.1 – O que é Arduino?.....	11
3.2 – Recursos para usar com Arduino.....	12
3.2.1 – IDE Arduino.....	13
3.2.2 – Simulador Tinkercad.....	15
4 – Conceitos básicos em eletrônica.....	17
4.1 – Elementos de um circuito.....	17
4.2 – Apresentação de alguns componentes.....	19
5 – Projetos com Arduino.....	23
5.1 – Acendendo um LED.....	25
5.2 – Programando um circuito com Arduino.....	30
5.3 – Usando Protoboard.....	35
5.4 – Usando Fotorresistor.....	41
5.5 – Usando Potenciômetro.....	47
5.6 – Usando LCD.....	52
5.7 – Usando sensor de presença.....	59
5.8 – Usando Buzzer.....	65
5.9 – Usando sensor de temperatura.....	70
REFERÊNCIAS.....	75

1 – Introdução

Sabemos que é um grande desafio tanto para os professores como para os alunos, o aprendizado da disciplina de lógica de programação. A lógica consiste dos fundamentos necessários para o aprendizado da programação de computadores e dispositivos eletrônicos. Portanto, é necessário que o aluno desenvolva competências e habilidades, através do uso de algoritmos, para a construção de softwares, aplicações e dispositivos.

Este material tem como proposta inicial, auxiliar os professores e alunos na disciplina de lógica de programação, geralmente, ofertada nos cursos técnicos de informática, pelas instituições de educação profissional e tecnológica no Brasil.

A compreensão dos conceitos de algoritmos é de fundamental importância, para que o aluno possa dar seus primeiros passos em direção aos objetivos da disciplina. A capacidade de analisar, abstrair e compreender um problema, faz com que seja necessário ter uma organização lógica e sequencial das ações tomadas para se ter a solução do problema.

Com a grande propagação das tecnologias em diversas áreas, há uma grande necessidade de fomento do ensino da computação na educação, mais especificamente, na educação básica. Para isso, se faz necessário a implantação de disciplinas de computação nos currículos das escolas, favorecendo o aprendizado de tais tecnologias.

Outrossim, com o surgimento e expansão de dispositivos inteligentes e interconectados em redes, dando origem ao termo inglês *Internet of Things* (IOT) ou Internet das Coisas, é imprescindível que a nova geração de crianças, jovens e adultos, também chamados de nativos digitais, tenham capacidades de entender o funcionamento de tais dispositivos, para uma melhor compreensão dos seus funcionamentos.

Esse material é o resultado de uma pesquisa de mestrado realizada no ano de 2020, feita com uma turma composta de 31 alunos da 1ª série do curso técnico em informática integrado ao ensino médio do Instituto Federal do Tocantins, campus Araguaína. A turma participou de oficinas de robótica educacional, de forma remota, devido a crise sanitária do coronavírus ocorrida no ano de 2019 e nos anos seguintes. Para a execução da prática foi utilizado um ambiente de simulação, que permitiu a criação de uma sala virtual com compartilhamento de projetos entre o professor e alunos.

A pesquisa teve como objetivo o uso da robótica educacional para promover a contextualização e articulação da teoria com a prática no ensino-aprendizagem de lógica de programação, de forma que os alunos desenvolvam habilidades que favoreçam o raciocínio lógico, resolução de problemas, trabalho em equipe, dentre outras.

Portanto, pretendemos, através dos capítulos seguintes, tornar o aprendizado da lógica ou o desenvolvimento do raciocínio lógico de uma forma mais fácil, compreensível e contextualizado. Para isso, os conceitos da robótica educacional serão utilizados como suporte ao ensino e aprendizado da lógica de programação.

Para um melhor aproveitamento do material é recomendável que o aluno tenha uma atitude ativa e busque aplicar na prática os conceitos da disciplina de lógica vistos em sala de aula. Por outro lado, o professor tenha um papel de orientador, facilitando o processo de ensino-aprendizagem.

No capítulo 2 será mostrada a importância da robótica aplicada no contexto educacional. No capítulo 3 serão apresentados os artefatos tecnológicos e eletrônicos necessários para a construção dos projetos deste material. No capítulo 4 serão apresentados os conceitos fundamentais da eletrônica, para uma identificação e manipulação correta de componentes eletrônicos. Por fim, no capítulo 5 serão apresentados os projetos desenvolvidos nas oficinas de robótica, montagem e programação dos circuitos.

O material está disponibilizado para download, além do repositório educapes (<https://educapes.capes.gov.br>), também no seguinte endereço <http://www.roboticaeduc.com.br>. Este site foi criado com o propósito de ser mais um espaço com fins educacionais. Portanto, pretende-se ampliar para todos a divulgação do conhecimento da robótica, tornando assim um espaço acessível, democrático e inclusivo a todos os alunos de escolas públicas do Brasil e àqueles que queiram contribuir de alguma forma para a inclusão da computação nos currículos das escolas.

2 – Robótica Educacional

Atualmente, muito se tem visto o crescente interesse das crianças, adolescentes e jovens pela tecnologia. Arelado a isso, as escolas estão buscando formas de diversificar seus currículos com a introdução da robótica nas aulas, proporcionando um aprendizado interdisciplinar e baseado na resolução de problemas.

A robótica educacional consiste do uso de artefatos tecnológicos, como dispositivos eletrônicos programáveis, aplicados na educação básica ou superior. Para CAMPOS (2019, p. 28), “a utilização de instrumentos robóticos na educação (infantil, fundamental, média e superior) recebe o nome de robótica pedagógica ou educacional, que consiste na utilização de aspectos/abordagens da robótica industrial em um contexto no qual as atividades de construção, automação e controle de dispositivos robóticos propiciam aplicação concreta de conceitos, em um ambiente de ensino e de aprendizagem”.

A aprendizagem de conteúdos com o uso da robótica, torna as aulas mais divertidas, contextualizadas e práticas. Os professores buscam a experimentação de conceitos teóricos, através da construção de artefatos, de forma que o aluno participe de todo o processo. Com isso, ele compreende não somente o resultado final, mas também as particularidades individuais que fazem parte do processo.

Segundo CAMPOS (2019), a robótica na educação pode ser utilizada de três formas: a primeira abordagem se refere a aprendizagem de conceitos que estão relacionado diretamente com a robótica, como por exemplo, a programação de dispositivos e construção de objetos robóticos; a segunda abordagem está relacionada com a interdisciplinaridade, onde serão desenvolvidos projetos que relacionam conceitos diversos, como, matemática, física, lógica, dentre outros; e a terceira abordagem é a integração da primeira e segunda categoria, ou seja, são feitos projetos que envolvem tanto a aprendizagem da robótica, como também questões interdisciplinares.

Este material seguirá a terceira abordagem, onde, no decorrer da execução dos projetos, o aluno fará a identificação de componentes e montagem, e também aplicará os conceitos da lógica de programação, como, tipos de dados, declaração de variáveis, laços de repetição, estruturas de decisão, dentre outros.

Para uma experimentação prática, poderão ser utilizados kits de robótica, compostos de motores, polias, sensores, peças de sucata (metais, plásticos, madeiras, etc.) e microcontroladores que são controlados através de interfaces de comunicação e programados por uma linguagem de programação para a construção de dispositivos.

Para CAMPOS (2019, p. 32), é preciso “considerar que a robótica pressupõe três componentes: a utilização de conjuntos de montagem para a construção de dispositivos, o computador e uma linguagem de programação que permita dar movimento ao dispositivo construído”.

Existem diversos conjuntos ou kits de montagem de vários fabricantes que podem ser adquiridos pela internet. Esse material busca apresentar um microcontrolador de baixo custo, com ampla comunidade de desenvolvedores e usuários, de hardware e software *open-source*, chamado Arduino.

Segundo MCROBERTS (2015, p. 28) o conceito de *open-source*, “significa que o código, os esquemas, o projeto etc. são abertos e qualquer pessoa pode usá-los livremente para fazer o que desejar”. Isso significa que se pode fazer clones de um hardware e software e até modificá-los em sua estrutura física e lógica, sem nenhuma restrição com relação a propriedade intelectual ou patentes.

Há kits educacionais no mercado que nem sempre são as melhores opções, pois o preço da maioria dos produtos comercializados se torna inviável para muitas instituições. Portanto, no cenário onde a educação pública não tem muito investimento pelo poder público, a proposta de se trabalhar com componentes de baixo custo se torna indispensável (MORALES *et al*, 2017).

Diante do contexto, podemos perceber que a robótica educacional proporciona diversos benefícios no ambiente escolar. E para potencializar esses benefícios, esse material tem o objetivo de auxiliar os professores nas aulas de lógica de programação para o desenvolvimento do pensamento computacional dos alunos.

Segundo WING (2006), o pensamento computacional é a capacidade de resolver problemas utilizando fundamentos baseados na ciência da computação. Ela propôs que as estratégias utilizadas pelos cientistas da computação para a solução de problemas, deveriam ser aplicadas não somente à solução de problemas computacionais, mas também a outras disciplinas e à vida cotidiana.

A robótica aplicada na educação vem proporcionar o que Jeannete Wing defende, que é o desenvolvimento de habilidades como a abstração de problemas, o reconhecimento de padrões para representar problemas de novas maneiras, o pensamento algorítmico e a divisão de problemas em partes menores. Portanto, essas são habilidades que esse material pretende desenvolver nos alunos.

3 – Plataforma Arduino

Este capítulo consiste da apresentação da plataforma Arduino e dos recursos necessários para a implementação de códigos utilizando a linguagem de programação padrão.

3.1 – O que é Arduino?

Segundo o site oficial (www.arduino.cc), Arduino é uma plataforma eletrônica de código aberto baseada em hardware e software. O hardware é composto de um microcontrolador, memória RAM, memória Flash, portas digitais, portas analógicas, comunicação USB, dentre outras funcionalidades. O software consiste de um Ambiente de Desenvolvimento Integrado (IDE), que é programado através da linguagem de programação baseada na linguagem Wiring. Pode ser considerado também como um pequeno computador que é programado para processar entradas e saídas entre o dispositivo e os componentes externos conectados a ele. A Figura 3.1 mostra um exemplo de um Arduino, chamado UNO.

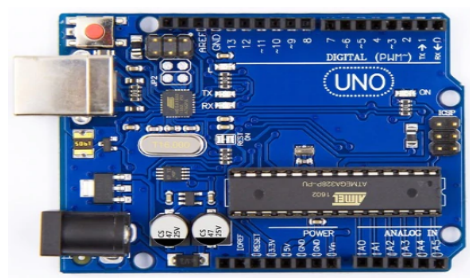


Figura 3.1: Exemplo de um Arduino UNO

Através da plataforma de prototipagem eletrônica Arduino, pode-se criar diversos projetos, desde pequenos exemplos, como a automatização da irrigação de uma horta, passando pelo controle de diversos dispositivos de uma casa. Conforme MCROBERTS (2015, p. 27) “o Arduino pode ser usado para desenvolver objetos interativos independentes ou ser conectado a um computador, a uma rede ou até mesmo à internet para recuperar e enviar dados do Arduino e trabalhar com eles”.

Segundo BANZI e SHILOH (2015, p. 17), o Arduino “foi projetado para artistas, designers e outros profissionais que queiram incorporar a computação física a seus projetos sem que para isso precisem ter se formado em Engenharia Elétrica”. Portanto, a ideia inicial é que o dispositivo seja utilizado por qualquer pessoa que queira criar algo diferente e que não seja experiente em componentes eletrônicos ou programação.

Na educação, professores e alunos usam o Arduino para construir instrumentos científicos de baixo custo, para comprovar os princípios teóricos da química e da física ou para iniciar a programação e a robótica.

Nos projetos de demonstração constante deste produto educacional, e-book, utilizaremos a placa (hardware) Arduino modelo Uno R3, Figura 3.2. Este é composto de 14 pinos digitais (0 a 13), que podem ser utilizados como entrada e saída. Os pinos 3, 5, 6, 9, 10 e 11 são rotulados com ~, pois são considerados pinos PWM (*Pulse Width Modulation*), na seção dos projetos com Arduino mostraremos o significado PWM e o seu funcionamento. Os pinos A0, A1, A2, A3, A4 e A5 são pinos de entradas analógicas, para serem utilizados com componentes analógicos, como por exemplo, um sensor de temperatura. Há pinos de 5V, 3,3 e GND (*Ground*), que são pinos para alimentar com energia diversos componentes que estarão no circuito e conectados à placa Arduino.

O Arduino Uno R3 possui como configuração um microprocessador ATmega328, com memória RAM de 2KB, memória Flash de 32 KB e um clock de 16MHz (OLIVEIRA *et al*, 2018).

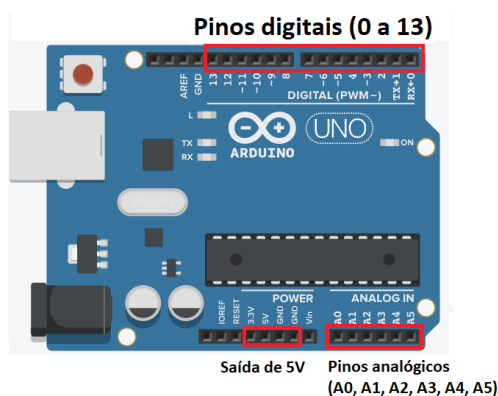


Figura 3.2: placa Arduino Uno R3.

3.2 – Recursos para usar com Arduino

Esta seção consiste em apresentar alguns recursos que serão necessários para implementar algoritmos no Arduino. Será apresentado o ambiente de desenvolvimento integrado (IDE) e como alternativa a não ter o dispositivo físico (hardware) do Arduino, será mostrado um simulador que oferece um ambiente completo com diversos componentes (sensores, motores, microcontrolador) para serem programados. Este recurso de simulação é muito importante, pois permite que escolas públicas sem recursos financeiros para adquirir kits de robótica, possam aprender da mesma forma como se estivessem utilizando o hardware Arduino, bastando ter apenas um computador com internet para acesso. No decorrer do material apresentaremos os projetos desenvolvidos no

ambiente de simulação, como uma forma de facilitar o processo de ensino. Vale destacar que toda montagem de circuitos e implementação de algoritmos realizado no ambiente de simulação, vale também para o dispositivo físico e a IDE do Arduino.

3.2.1 – IDE Arduino

Apresentado o hardware do Arduino e suas funcionalidades na seção anterior, esta seção tem o objetivo de apresentar o ambiente de desenvolvimento do Arduino (IDE), que é um software *open-source*, gratuito e que pode ser baixado no endereço <http://www.arduino.cc>. A IDE é multiplataforma, tendo versões para Windows, Linux e Mac OS X.

A Figura 3.3 apresenta a IDE que é composta de uma barra de título, barra de menus, barra de ferramentas, monitor serial e janela de saída. Na barra de ferramentas existem funcionalidades para verificação de erros de sintaxes, compilação de código para a placa Arduino, novo *sketch*, salvar e abrir. O monitor serial é utilizado para exibir informações na comunicação serial, que é feita entre o computador e a placa Arduino. Na janela de saída, são exibidas informações a respeito da compilação do código, erros de sintaxe, tamanho em *bytes* do *sketch*, dentre outras.

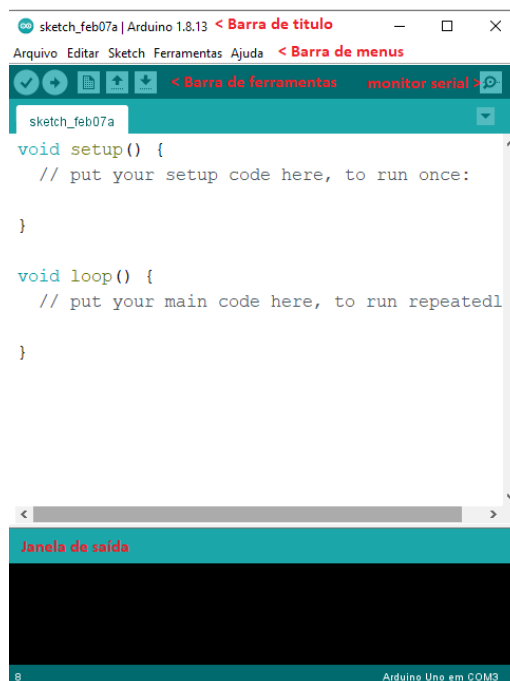


Figura 3.3: Ambiente de desenvolvimento do Arduino (IDE).

Todo novo programa é chamado de *sketch*, que possui duas funções básicas, *setup()* e *loop()*. A função *setup()* é chamada quando um *sketch* inicia. É usada para inicializar variáveis, configurar o modo dos pinos como *INPUT* ou *OUTPUT*, inicializar bibliotecas, etc. Ela será executada apenas uma vez, após a placa ser alimentada ou se acontecer um *reset*. Já a função *loop()* repete consecutivamente as instruções de código que estão dentro dela e enquanto a placa estiver ligada ou até que outro *sketch* seja carregado na memória do Arduino.

A palavra reservada *void* indica que as funções não apresentam um valor de retorno, sendo usadas exclusivamente para realizar a execução de um conjunto de instruções (OLIVEIRA *et al*, 2018).

Para realizar a compilação do programa, acesse o menu *Sketch* → *Verificar/Compilar* ou utilize o atalho *Ctrl+R*, como mostra a Figura 3.4. A cada nova compilação o código do *sketch* será transferido para a placa do Arduino e substituirá o código anterior armazenado na memória *flash*.

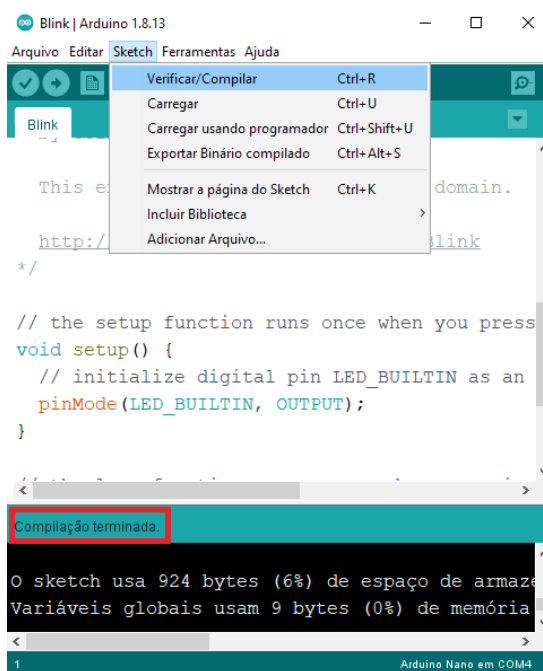


Figura 3.4: exemplo da compilação do sketch.

Na janela de saída é mostrado o resultado da compilação, caso não tenha nenhum erro no código, mostrará a mensagem ‘compilação terminada’ e informações sobre o tamanho em *bytes* do código armazenado na memória da placa Arduino.

3.2.2 – Simulador Tinkercad

O Tinkercad é um ambiente de simulação, gratuito, que oferece diversos componentes (sensores, motores, microcontrolador, dentre outros) para serem utilizados em um circuito programável. A Figura 3.5 mostra um exemplo do ambiente, utilizando um circuito com o componente LCD, protoboard e Arduino.

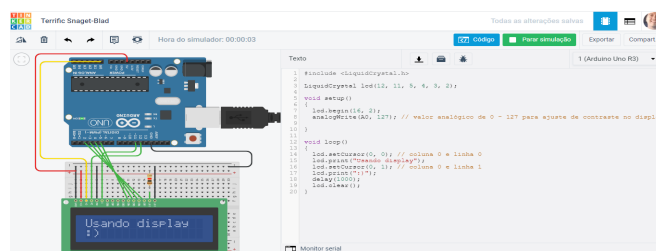


Figura 3.5: Exemplo do ambiente Tinkercad.

A plataforma pode ser acessada pelo seguinte endereço <http://www.tinkercad.com> e oferece diversos recursos, como modelagem 3D, programação de circuitos, dentre outros. Nosso foco será a criação de circuitos programáveis com Arduino. Ao entrar na página inicial é necessário criar uma conta para acessar o painel de controle. Há cadastro para educadores, alunos e conta pessoal. Caso o usuário seja um educador, ele pode criar uma sala de aula e incluir os alunos, passando um código de acesso para eles. Caso seja aluno, no cadastro é necessário digitar o código compartilhado pelo professor para entrar na turma.

Feito o cadastro na plataforma, o próximo passo será acessar o painel de controle, Figura 3.6, onde se encontra o menu lateral esquerdo ‘Circuitos’, clicando no menu, será exibido a lista de projetos criados. Caso não tenha nenhum projeto criado, o botão verde ‘Criar novo circuito’, cria um novo projeto.

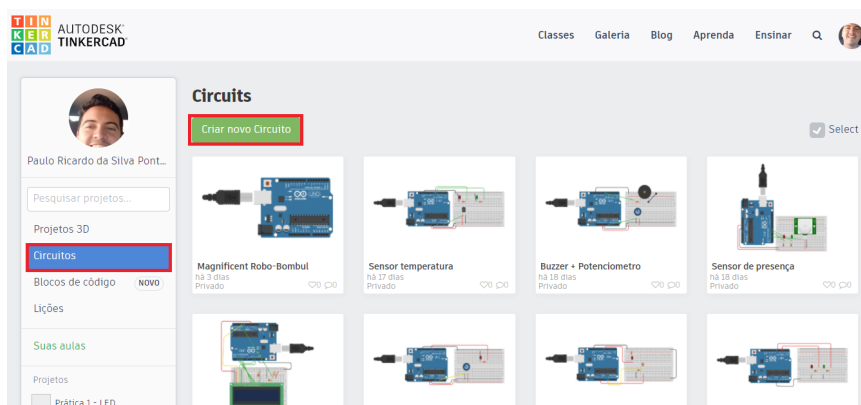


Figura 3.6: Painel de controle do Tinkercad.

Criado um novo projeto, é aberta a área de trabalho, Figura 3.7, onde se encontra localizado na parte superior a barra de título, que pode ser renomeada com o nome que identifique o projeto. A barra de ferramentas permite funções como rotação, exclusão, desfazer, refazer, anotação, código, iniciar simulação, exportar, compartilhar, dentre outros.

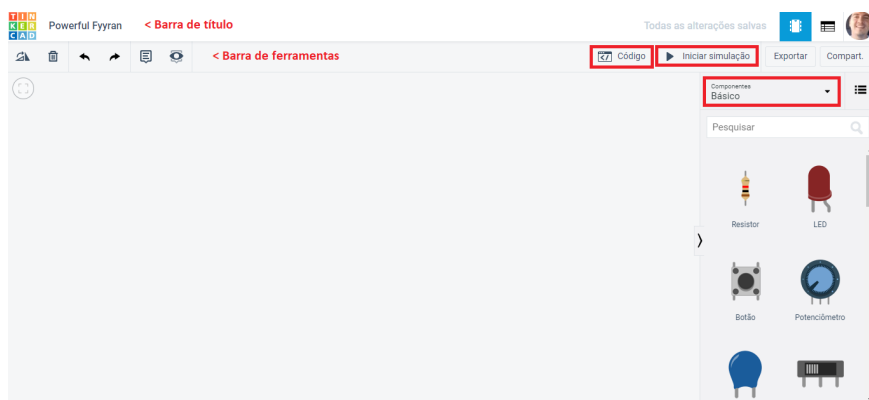


Figura 3.7: Área de trabalho do Tinkercad.

Na lateral direita é exibido o painel de componentes, onde se encontra um campo de pesquisa, para localizar um componente em específico. Para exibir todos os componentes na lista, basta clicar na lista suspensa e escolher a opção ‘*todos*’. Para a montagem do circuito, é necessário arrastar com o *mouse* o componente para o centro da tela.

Caso o circuito seja composto de um microcontrolador ou uma placa Arduino, será necessário programá-la. Para isso, basta clicar no botão ‘*Código*’ para inserir as instruções. Há duas formas de programar no ambiente Tinkercad, por blocos ou texto.

Nesse material utilizaremos a forma de programação por texto. Terminada a montagem do circuito e inserido os códigos, basta iniciar a simulação para ver o projeto funcionando. Caso tenha algum erro no código, serão exibidas mensagens de aviso indicando a localização do erro.

Portanto, a plataforma Tinkercad se torna uma ótima alternativa para ser utilizada em sala de aula, laboratórios, espaços *maker* das escolas públicas que não têm recursos financeiros para aquisição de componentes físicos para o aprendizado da robótica, basta ter computadores com acesso à internet para utilização de um ambiente completo.

4 – Conceitos básicos em eletrônica

A robótica consiste da manipulação de componentes eletrônicos e para a construção dos projetos desse material, é necessário saber identificar e entender o funcionamento de cada componente. Portanto, este capítulo consiste em apresentar alguns conceitos básicos sobre eletrônica. Os fundamentos da eletrônica se baseiam nos conceitos de corrente elétrica, tensão e resistência.

4.1 – Elementos de um circuito

Um circuito é composto de corrente, tensão e resistência. A corrente elétrica corresponde ao fluxo ordenado de cargas elétricas em um condutor. A tensão é a pressão exercida sobre os elétrons de um ponto A para um ponto B. Já a resistência corresponde a limitação de corrente elétrica em uma certa parte do circuito.

Para um melhor entendimento, vamos fazer uma analogia com um tanque d'água, conforme a Figura 4.1. A corrente elétrica corresponde ao fluxo de água que sai do tanque, a tensão corresponde a pressão exercida sobre o nível da água armazenada no tanque, a resistência corresponde ao diâmetro do furo que permite a saída de água. Portanto, quanto maior for o furo, menor é a resistência. Por outro lado, quanto menor for o furo, maior é a resistência sobre a água saindo do tanque.

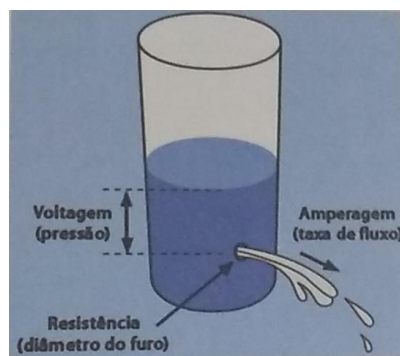


Figura 4.1: exemplo de analogia com um tanque d'água. Fonte: PLATT (2016).

A Figura 4.2 mostra um exemplo de um circuito, composto de Tensão (V), Corrente (I) e Resistência (R). Neste exemplo, a resistência pode ser uma lâmpada, que depois da passagem de

corrente elétrica, esta será convertida em energia luminosa, acendendo a lâmpada. Portanto, a lâmpada oferece certa resistência sobre a passagem da corrente elétrica.

É importante destacar que a resistência depende de diversos fatores, como o material, formato, comprimento, temperatura, dentre outros. Há certos tipos de materiais que oferecem diferentes níveis de resistência. Um exemplo típico de resistência, são os chuveiros elétricos que possuem uma resistência elétrica, que transformam a energia elétrica em energia térmica.

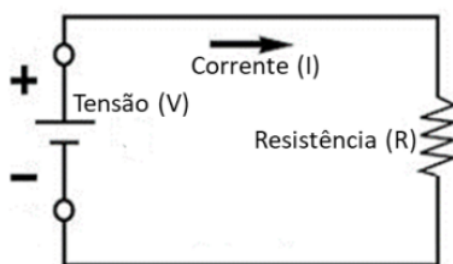


Figura 4.2: exemplo de um circuito. Fonte: OLIVEIRA et al (2018).

Podemos observar no exemplo da Figura 4.2, a tensão V corresponde a uma fonte de energia que possui dois polos, positivo e negativo, e a diferença de potencial entre esse polos, gera uma corrente elétrica, representada pela corrente I , que se desloca no circuito do maior potencial para o menor potencial. Segundo PLATT (2016, p. 31), “componentes eletrônicos de estado sólido geralmente requerem alimentação DC com tensões entre 5V e cerca de 20V, embora dispositivos modernos instalados sobre uma superfície podem usar menos de 2V”.

Nesse contexto, podemos estabelecer uma relação entre cada elemento do circuito: tensão, corrente e resistência. É o que deu origem a chamada lei de Ohm, que consiste na seguinte fórmula:

$$i = V / R$$

Onde, i corresponde a corrente em Amperes (A), tensão V é expressa em Volts (V) e resistência R é expressa em Ohms (Ω). Portanto, para exemplificar a aplicação da lei de Ohm, considere um circuito, Figura 4.3, em que a tensão é de 5V e uma resistência de 220 Ω . Aplicando a fórmula, temos:

$$i = 5 / 220$$

$$i = 0,022 \text{ A ou } i = 22\text{mA}$$

O circuito gerará uma corrente elétrica de 0,022 Amperes ou equivalente a 22 miliamperes.

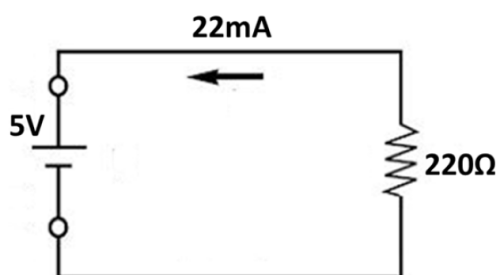


Figura 4.3: exemplo da aplicação da lei de Ohm. Fonte: OLIVEIRA et al (2018).

A tabela 4.1 mostra as derivações da unidade Ampere: miliampere, microampere, nanoampere e assim por diante.

Tabela 4.1: derivações do ampere.

Nome	Símbolo	Valor em A
Ampère	A	1
Deciampère	dA	10^{-1}
Centiampère	cA	10^{-2}
Miliampère	mA	10^{-3}
Microampère	μ A	10^{-6}
Nanoampère	nA	10^{-9}
Picoampère	pA	10^{-12}

Fonte: PLATT (2016).

4.2 – Apresentação de alguns componentes

Esta seção tem o objetivo de apresentar alguns componentes que podem fazer parte de um circuito e que servem para diversas finalidades.

As **baterias** e **pilhas**, Figura 4.4, são componentes básicos e essenciais, que fazem parte de um circuito. São fontes de energia com o objetivo de alimentar o circuito com uma corrente elétrica. A tensão pode variar conforme o tipo de bateria e pilha. Para componentes eletrônicos são utilizados baterias e pilhas de 12V, 9V e até 1.5V.



Figura 4.4: exemplos de pilhas e baterias. Fonte: PLATT (2016).

Os **Jumper cables** ou cabos de ligação, Figura 4.5, são necessários para fazer conexões e são utilizados para a transferência da corrente elétrica entre uma fonte de energia e os demais componentes do circuito. Existem os tipos macho e fêmea e de comprimentos variados.



Figura 4.5: exemplo de jumper cable do tipo macho. Fonte: PLATT (2016).

O **Diodo emissor de luz** ou LED (*Light-Emitting Diode*), Figura 4.6, tem como característica a emissão de luz quando uma corrente elétrica passa por ele. É um componente polarizado, desta forma, o Cátodo (lado chanfrado) sempre deve estar conectado ao polo negativo (ou terra) do circuito, se conectado invertido pode ser danificado.

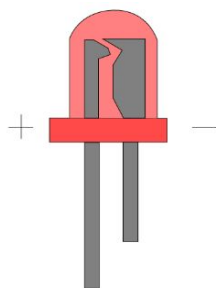


Figura 4.6: exemplo de LED. Fonte: PLATT (2016).

O terminal maior sempre indica o terminal positivo e o terminal menor indica o terminal negativo. A Figura 4.7, mostra os diferentes tipos de LEDs e as suas variadas cores.

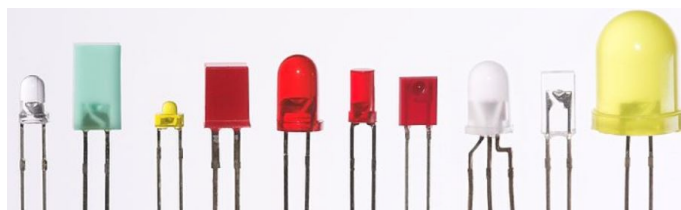


Figura 4.7: exemplo de diferentes tipos de LED.
Fonte: PLATT (2016).

Os **resistores**, Figura 4.8, são componentes que restringe ou limitam a passagem de corrente elétrica em um circuito. Eles são necessários do modo a evitar que determinados componentes eletrônicos recebam uma tensão ou corrente maior do que eles podem suportar evitando, desta forma, que os mesmos sejam danificados.

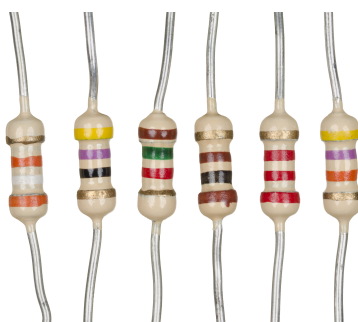


Figura 4.8: exemplo de resistores.
Fonte: PLATT (2016).

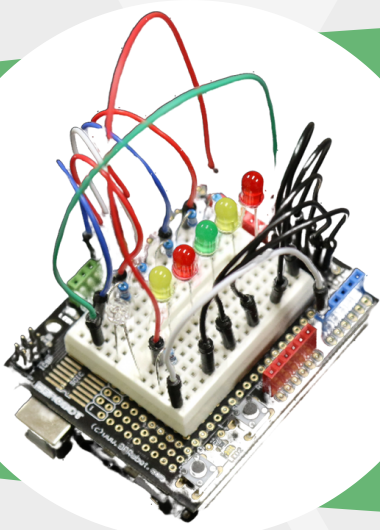
Como podemos observar na Figura 4.8, os resistores possuem faixas coloridas que identificam a sua resistência. A Figura 4.9 mostra a identificação de cada faixa, correspondente a sua cor.

4- Faixas					
1.0 K Ω $\pm$ 5%					
	1st	2nd	3rd	4th	
Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador	Decimal
Preto	0	0	0	1	1
Marron	1	1	1	10	10
Vermelho	2	2	2	100	100
Laranja	3	3	3	1K	1.000
Amarelo	4	4	4	10K	10.000
Verde	5	5	5	100K	100.000
Azul	6	6	6	1M	1.000.000
Violeta	7	7	7	10M	10.000.000
Cinza	8	8	8		100.000.000
Branco	9	9	9		1.000.000.000
Ouro				0.1	$\pm$ 5%
Prata				0.01	$\pm$ 10%
Branco					$\pm$ 20%

5- Faixas					
254 Ω $\pm$ 1 %					
	1st	2nd	3rd	4th	5th

Figura 4.9: código de cores do resistor.
Fonte: PLATT (2016).

Para a leitura correta de um resistor: as faixas coloridas são lidas sempre a partir daquela que está mais próxima da extremidade; a primeira faixa representa o primeiro algarismo que irá compor o valor da resistência; a segunda faixa indica o segundo algarismo; a terceira faixa indica qual o número de zeros do valor da resistência; a quarta faixa indica qual a margem de erro no valor da resistência.



Projetos com Arduino

Este capítulo consiste em apresentar os projetos que foram realizados na oficina junto aos alunos do curso técnico de informática, na disciplina de lógica de programação, no IFTO campus Araguaína.

As oficinas foram planejadas de forma que o conteúdo seja progressivo no nível de conhecimento aprendido. Foi demonstrado desde os conceitos de vários componentes até a montagem completa e programação do circuito.

Os projetos seguintes são:

- Acendendo um LED
- Programando um circuito com Arduino
- Usando Protoboard
- Usando Fotorresistor
- Usando Potenciômetro
- Usando LCD
- Usando sensor de presença
- Usando Buzzer
- Usando sensor de temperatura



Acendendo um LED

Objetivos de aprendizagem

- Montar um circuito simples não programável, compreender o funcionamento e as conexões dos elementos que fazem parte;
- Usar uma fonte de energia apropriada para que não queime componentes, de forma que seja compreendido o conceito de tensão, corrente e resistência;
- Construir circuitos e executar simulação no ambiente Tinkercad.

5.1 – Acendendo um LED

Exemplo 1: este projeto demonstra o uso de um Diodo Emissor de Luz (LED) e uma fonte de energia com uma tensão de 1.5V. O objetivo é mostrar um circuito em funcionamento com uma tensão apropriada para o fornecimento de corrente elétrica através dos componentes que fazem parte do circuito.

Material necessário

- 1 pilha de 1.5V.
- 1 Led (vermelho).
- Jumper cable.

Passo 1: montagem do circuito



Figura 5.1: exemplo do circuito com uma pilha de 1.5V e um LED vermelho.

O circuito é composto de um LED (vermelho), onde o ânodo (polo positivo do led) é conectado, através do fio vermelho, ao terminal positivo da pilha e o cátodo (polo negativo do led) é conectado, através do fio preto, ao terminal negativo da pilha.

Passo 2: Executando o projeto no Tinkercad

Para ver o projeto em execução no simulador Tinkercad, clique no botão ‘Iniciar simulação’, na parte superior direita da tela, como mostra a Figura 5.2.

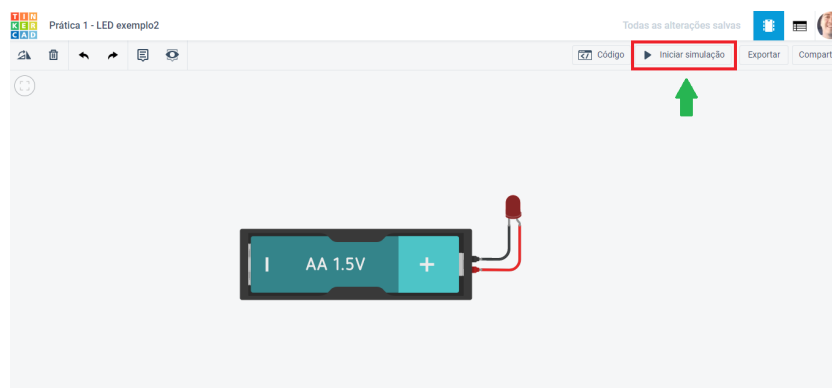


Figura 5.2: circuito com Led no ambiente Tinkercad.

Neste exemplo de projeto, podemos observar que não há nenhum tipo de programação, apenas mostra o funcionamento de um circuito com uma ligação direta entre um led e uma fonte de energia, como mostra na execução da simulação, na Figura 5.3.

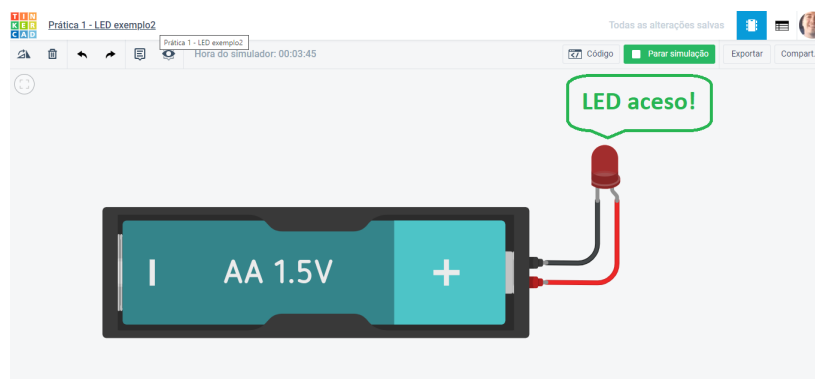


Figura 5.3: simulação iniciada.

Exemplo 2: este projeto demonstra o uso de um Diodo Emissor de Luz (LED), um Resistor de 1 kOhm e uma bateria de 9V. O objetivo é mostrar um circuito em funcionamento com uma resistência apropriada para o fornecimento de corrente elétrica através dos componentes que fazem parte do circuito.

Material necessário

- 1 bateria de 9V.
- 1 Led.
- 1 Resistor de 1 k Ω (marrom, preto e vermelho).
- Jumper cable.

Passo 1: montagem do circuito

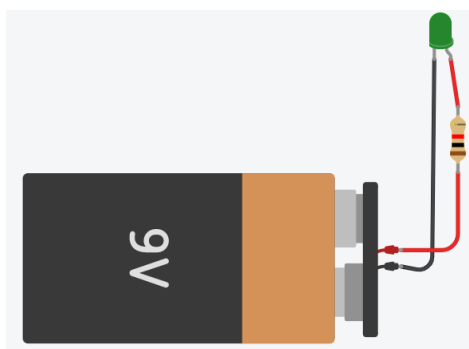


Figura 5.4: exemplo de um circuito com uma bateria de 9V, resistor de 1k ohms e um led verde.

O circuito é composto de um LED (verde), onde o ânodo (polo positivo do led) é conectado em um dos terminais do resistor e outro terminal do resistor é conectado através do fio vermelho ao terminal positivo da bateria, o cátodo (polo negativo do led) é conectado ao terminal negativo da pilha. O uso do resistor é apropriado para limitar a capacidade de corrente elétrica no componente led, pois, como o circuito oferece uma tensão de 9V, há uma quantidade de corrente elétrica muito alta e com isso o resistor impede que o LED queime.

Passo 2: Executando o projeto no Tinkercad

Para ver o projeto em execução no simulador Tinkercad, clique no botão ‘Iniciar simulação’, na parte superior direita da tela.

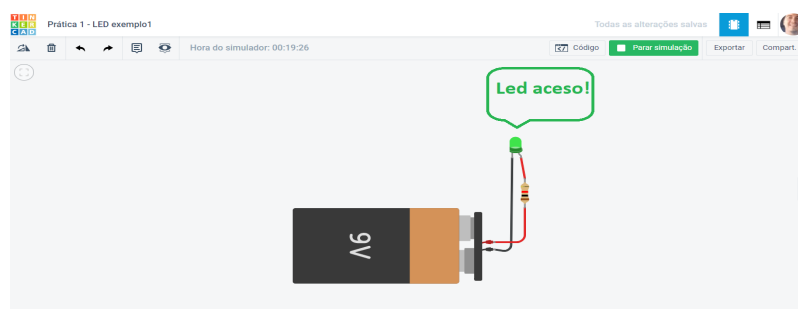
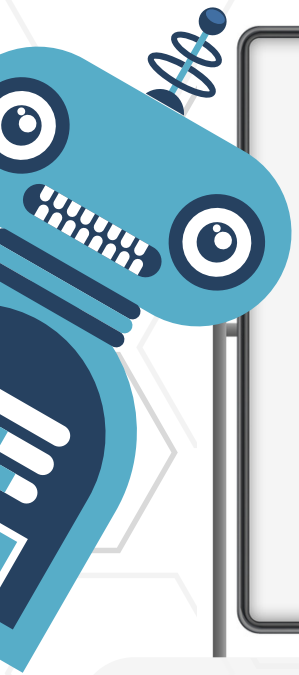


Figura 5.5: exemplo de um circuito com um led, um resistor de 1kOhm e uma bateria de 9V.



Hora de praticar!

Prática 1:

1. Monte um circuito com um LED e duas pilhas com tensão de 1.5V, cada, e faça as ligações apropriadas, utilizando jumper cable para ligação dos terminais das pilhas aos polos positivo e negativo do LED e veja o resultado. O que aconteceu? Por que isso aconteceu?
2. Utilizando a lei de Ohm, visto no capítulo 4, para montar um circuito com um LED e uma bateria com uma tensão de 9V e uma corrente elétrica de 0,020A. Qual seria o valor apropriado em Ohm do resistor, de forma que não queime o LED?
3. Monte um circuito com um LED, uma bateria com tensão de 9V e um resistor com o valor, em Ohm, do resultado da questão 2 e veja se o LED acendeu.



Programando um circuito com Arduino

Objetivos de aprendizagem

- Montar um circuito programável com Arduino;
- Declarar e inicializar variáveis;
- Uso de funções internas (*pinMode*, *digitalWrite* e *delay*) da linguagem de programação padrão;
- Compreender o fluxo de execução do programa;
- Uso das portas digitais.

5.2 – Programando um circuito com Arduino

Este projeto demonstra o uso de um Diodo Emissor de Luz (LED), através da programação do Arduino. O objetivo é iniciar a programação do Arduino, através de um algoritmo para a simulação de um pisca-pisca, utilizando conceitos de declaração e inicialização de variáveis, operador de atribuição, funções internas da linguagem e uso das portas digitais.

Material necessário

- 1 Arduino.
- 1 Led.
- 1 Resistor de 1 k Ω .
- Jumper cable.

Passo 1: montagem do circuito

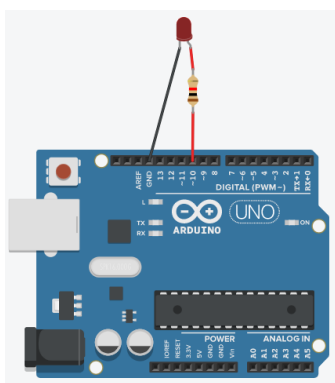


Figura 5.6: exemplo de um circuito com Arduino, led e um resistor.

O circuito é composto de um Arduino, um led (vermelho), um resistor de 1 k Ω .

- Conecte o fio vermelho, como mostra a Figura 5.6, que liga o terminal positivo (ânodo) do led em um dos terminais do resistor e o outro terminal do resistor é conectado na porta digital 10;
- Conecte o outro terminal negativo do led (cátodo), através do fio preto, direto na porta GND (Ground), a porta GND corresponde ao terra.

Passo 2: Código-fonte

Para inserir o código no ambiente Tinkercad, clique no botão ‘Código’, parte superior direita, conforme a imagem 5.7, e mude a opção de ‘Blocos’ para a opção ‘Texto’.

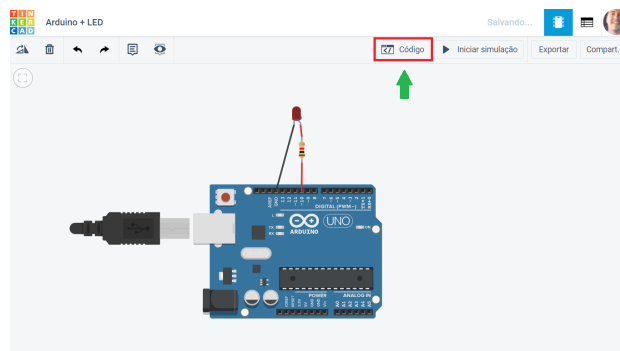


Figura 5.7: janela de código no ambiente Tinkercad.

Será aberta uma janela, perguntando se tem certeza que deseja fechar o editor de blocos. Clique em continuar.

```
1 // declara a variável chamada led com o valor da porta digital 10
2 int led = 10;
3
4 // função que inicializa o modo dos pinos utilizados
5 void setup()
6 {
7     // função que configura o pino 10 como OUTPUT
8     pinMode(led, OUTPUT);
9 }
10
11 // função que executa os algoritmos
12 void loop()
13 {
14     // liga o led
15     digitalWrite(led, HIGH);
16     // função que espera 1000 milisegundos de tempo
17     delay(1000);
18     // desliga o led
19     digitalWrite(led, LOW);
20     // função que espera 1000 milisegundos de tempo
21     delay(1000);
22 }
```

Figura 5.8: código-fonte do projeto.

No código da Figura 5.8, linha 2, é declarada uma variável do tipo de dado *int*, com o identificador chamado *led*, que recebe o valor 10. Essa variável, é responsável por armazenar o valor do pino digital 10, da placa Arduino. Na linha 5, a função `setup()`, será executada apenas uma vez, e é responsável por inicializar variáveis, configurar o modo dos pinos (*INPUT* ou *OUTPUT*), inicializar bibliotecas, etc.

Na linha 8, a função `pinMode()`, configura o pino especificado para funcionar como uma entrada ou saída e recebe dois parâmetros, o primeiro é o número do pino, o segundo é o modo do

pino, que pode ser *INPUT* (entrada) e *OUTPUT* (saída). Na linha 12, a função `loop()`, é onde os códigos serão inseridos para serem executados repetidamente, enquanto a placa Arduino estiver ligada.

Na linha 15, a função `digitalWrite()`, é responsável por acionar um valor *HIGH* ou *LOW* em um pino digital. Neste exemplo, a função liga o Led, que está no pino digital 10, com o parâmetro *HIGH*. Na linha 17, a função `delay()`, pausa o programa por uma quantidade específica de tempo (em milissegundos), cada segundo equivale a 1000 milissegundos. Neste exemplo da linha 17, a função deixa o Led aceso por 1 segundo. Na linha 19, a função `digitalWrite()` desliga o led, que está no pino digital 10, com o parâmetro *LOW*.

O algoritmo finaliza com novamente a função `delay()`, na linha 21, pausa o programa por 1000 milissegundos, deixando o Led apagado por 1 segundo. O ciclo se repete a partir da função `loop()` até que a placa Arduino esteja sendo alimentada por uma fonte de energia ou enquanto a simulação estiver sendo executada.

Passo 3: Executando o projeto no Tinkercad

Para ver o projeto em execução no simulador Tinkercad, clique no botão ‘Iniciar simulação’, na parte superior direita da tela, como mostra a Figura 5.9.

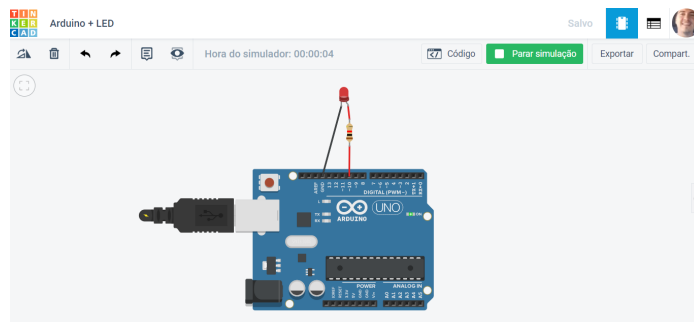


Figura 5.9: exemplo de projeto em execução.



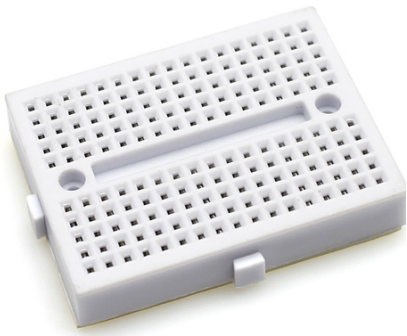
Hora



de

praticar!

Prática 2: Monte um circuito de forma que simule um semáforo de trânsito. O circuito é composto de 3 (três) LEDs, verde, amarelo e vermelho. Faça as ligações apropriadas e programe o Arduino de forma que os LEDs sejam acesos na seguinte sequência: primeiro acende o Led vermelho, depois o Led verde e por último o Led amarelo, retornando para o vermelho.



Usando Protoboard

Objetivos de aprendizagem

- Montagem de circuitos com vários componentes interligados através de uma matriz de contato;
- Uso do tipo de dado inteiro para declaração de variáveis numéricas;
- Controlar o tempo em milissegundos utilizando a função *delay*.

5.3 – Usando Protoboard

Este projeto demonstra o uso de uma Protoboard, também chamado de matriz de contato. A matriz de contato é muito utilizada para fazer ligações entre diversos componentes em um circuito. Uma matriz de contato permite conectar componentes de forma mais organizada que com cabos de teste que usamos até agora, e mais facilmente (e de modo reversível) que se usássemos solda para uni-los (PLATT, 2016).

A Figura 5.10, mostra um exemplo do funcionamento de uma matriz de contato, ela é composta de vários furos para encaixar os componentes e os fios. Como se observa na Figura 5.10, existe um barramento, longa linha de furos, superior e inferior (como mostram as setas verdes), que estão todos conectados na horizontal. Por outro lado, existe os pinos que estão todos conectados na vertical (como mostram as setas azul).

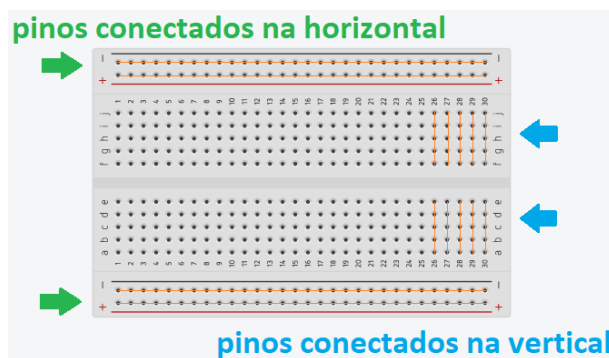


Figura 5.10: exemplo de matriz de contato.

A linha de furos conectados na horizontal, com o sinal – (negativo), é para ligação na porta *GND* do Arduino. A linha de furos conectados na horizontal, com o sinal + (positivo), é para ligação na porta de 5V ou 3.3V da placa Arduino.

Material necessário

- 1 Arduino.
- 2 Led.
- 2 Resistores de 1 k Ω .
- 1 Matriz de contato.
- Jumper cable.

Passo 1: montagem do circuito

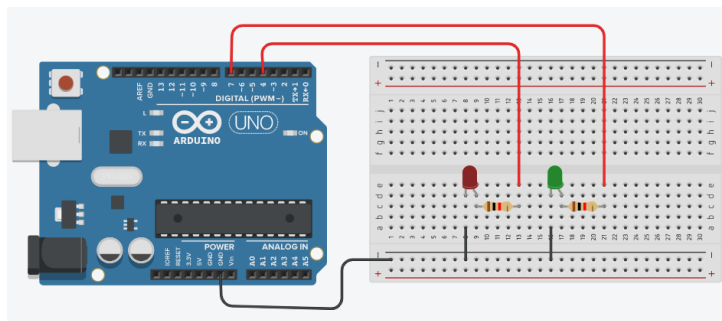


Figura 5.11: usando matriz de contatos com dois LEDs.

- Conecte o terminal positivo do led vermelho, na mesma coluna de furos que um dos terminais do resistor e o outro terminal do mesmo resistor é conectado na mesma coluna de furos que o fio vermelho que vai conectado na porta digital 4;
- Conecte o terminal negativo do led vermelho, na mesma coluna de furos do fio preto, que vai conectado no barramento negativo da protoboard, e esse barramento (negativo) está conectado direto na porta *GND* (*Ground*) da placa Arduino;
- Conecte o terminal positivo do led verde, na mesma coluna de furos que um dos terminais do resistor e o outro terminal do mesmo resistor é conectado na mesma coluna de furos que o fio vermelho que vai conectado na porta digital 7;
- Conecte o terminal negativo do led verde, na mesma coluna de furos do fio preto, que vai conectado no barramento negativo da protoboard.

Passo 2: Código-fonte

```
1 int led_vermelho = 4; // pino digital 4
2 int led_verde = 7;   // pino digital 7
3
4 void setup()
5 {
6     pinMode(led_vermelho, OUTPUT); // pino 4 como OUTPUT
7     pinMode(led_verde, OUTPUT);   // pino 7 como OUTPUT
8 }
9
10 void loop()
11 {
12     digitalWrite(led_vermelho, HIGH); // acende led vermelho
13     delay(1000);                       // espera por 1000 millisecond(s)
14     digitalWrite(led_vermelho, LOW);  // apaga led vermelho
15     delay(1000);                       // espera por 1000 millisecond(s)
16     digitalWrite(led_verde, HIGH);    // acende led verde
17     delay(1000);                       // espera por 1000 millisecond(s)
18     digitalWrite(led_verde, LOW);     // apaga led verde
19     delay(1000);                       // espera por 1000 millisecond(s)
20 }
```

Figura 5.12: código-fonte do projeto.

Para inserir o código no ambiente Tinkercad, clique no botão ‘Código’, parte superior direita, e mude a opção de ‘Blocos’ para a opção ‘Texto’. Será aberta uma janela, perguntando se tem certeza que deseja fechar o editor de blocos. Clique em continuar.

No código da Figura 5.12, linha 1, é declarada uma variável do tipo de dado *int*, com o identificador chamado *led_vermelho*, que recebe o valor 4. Essa variável, é responsável por armazenar o valor do pino digital 4, da placa Arduino. Na linha 2, é declarada uma variável do tipo de dado *int*, com o identificador chamado *led_verde*, que recebe o valor 7. Essa variável, é responsável por armazenar o valor do pino digital 7. Na linha 4, a função *setup()*, será executada apenas uma vez, e é responsável por inicializar variáveis, configurar o modo dos pinos (*INPUT* ou *OUTPUT*), inicializar bibliotecas, etc.

Na linha 6, a função *pinMode()*, configura o pino especificado para funcionar como uma entrada ou saída e recebe dois parâmetros, o primeiro é o número do pino, o segundo é o modo do pino, que pode ser *INPUT* (entrada) e *OUTPUT* (saída). Neste exemplo da linha 6, o pino do *led_vermelho* é configurado como *OUTPUT*, da mesma forma, na linha 7, o pino do *led_verde* é configurado como *OUTPUT*. Na linha 10, a função *loop()*, é onde os códigos serão inseridos para serem executados repetidamente, enquanto a placa Arduino estiver ligada.

Na linha 12, a função *digitalWrite()*, é responsável por acionar um valor *HIGH* ou *LOW* em um pino digital. Neste exemplo, a função liga o *led_vermelho*, que está no pino digital 4, com o parâmetro *HIGH*. Na linha 13, a função *delay()*, pausa o programa por uma quantidade específica de tempo (em milissegundos), cada segundo equivale a 1000 milissegundos. Neste exemplo da linha 13, a função *delay()* deixa o *led_vermelho* aceso por 1 segundo. Na linha 14, a função *digitalWrite()* desliga o *led_vermelho*, com o parâmetro *LOW*. Na linha 15, a função *delay()* mantém o *led_vermelho* desconectado por 1 segundo.

A partir da linha 16 em diante, o processo ocorrido com o *led_vermelho*, se repete para o *led_verde*. O ciclo se repete a partir da função *loop()* até que a placa Arduino esteja sendo alimentada por uma fonte de energia ou enquanto a simulação estiver sendo executada.

Passo 3: Executando o projeto no Tinkercad

Para ver o projeto em execução no simulador Tinkercad, clique no botão ‘Iniciar simulação’, na parte superior direita da tela, como mostra a Figura 5.13.

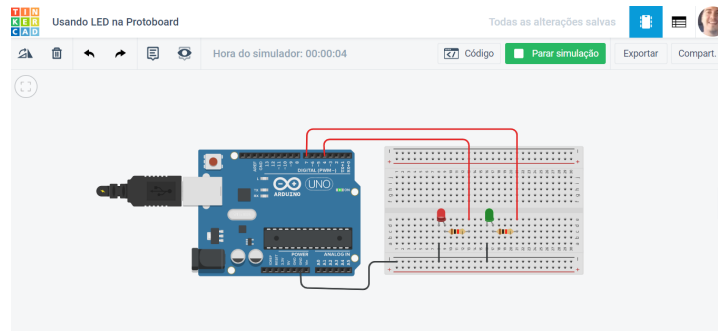
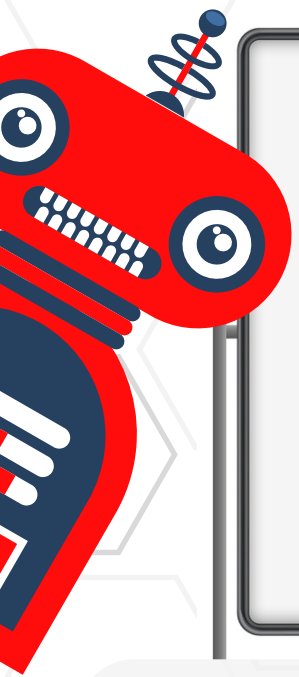


Figura 5.13: exemplo de projeto em execução.



Hora



de

praticar!

Prática 3: Usando uma Protoboard, monte um circuito de forma que simule um semáforo de trânsito para veículos e pedestres. O circuito é composto de 3 (três) LEDs, verde, amarelo e vermelho, representando o semáforo para veículos e 2 (dois) LEDs, verde e vermelho, representando o semáforo para pedestres. Faça as ligações apropriadas e programe o Arduino de forma que os LEDs sejam acesos e sincronizados na seguinte sequência: quando acender o Led vermelho para os veículos, o Led verde para os pedestres deve acender. Por outro lado, quando acender o Led verde para os veículos o Led vermelho para os pedestres deve acender.



Usando Fotoresistor

Objetivos de aprendizagem

- Uso das portas analógicas;
- Controlar eventos com o uso de estruturas de seleção compostas;
- Criar expressões lógicas com operadores relacionais;
- Uso da comunicação serial;
- Uso de funções internas (*analogRead*, *println*).

5.4 – Usando Fotoresistor

Este projeto demonstra o uso de um Fotoresistor, também chamado de LDR (*Light Dependent Resistor*). O LDR é um componente analógico, por isso, deve-se ligá-lo em uma porta analógica da placa Arduino. O LDR é um componente que varia a sua resistência conforme o nível de luminosidade que incide sobre ele. A resistência do LDR varia de forma inversamente proporcional à quantidade de luz incidente sobre ele.

O projeto apresenta o funcionamento das estruturas de seleção compostas, *if-else*, operador relacional, função para leitura analógica e a função de comunicação Serial. Segundo o site oficial do Arduino, a comunicação Serial é usada para comunicação entre uma placa Arduino e um computador ou outros dispositivos.

Material necessário

- 1 Arduino.
- 1 Led.
- 2 Resistores de 1 k Ω .
- 1 LDR.
- 1 Matriz de contato.
- Jumper cable.

Passo 1: montagem do circuito

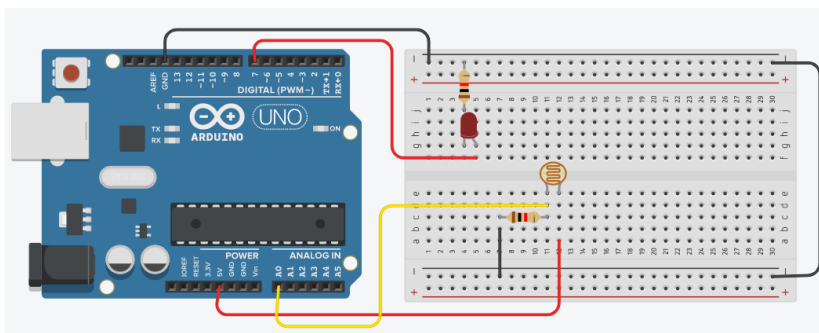


Figura 5.14: exemplo de circuito com LDR.

- Conecte o terminal positivo do led vermelho, na mesma coluna de furos que o fio vermelho que vai conectado na porta digital 7;
- Conecte o terminal negativo do led vermelho, na mesma coluna de furos de um dos terminais do resistor, e o outro terminal deste conectado direto no barramento (negativo) da

protoboard, através do fio preto, que vai conectado direto na porta *GND* (*Ground*) da placa Arduino;

- Conecte um dos terminais do LDR na mesma coluna de furos que o fio vermelho, que está conectado na porta de 5V;
- Conecte o outro terminal do LDR na mesma coluna de furos que o fio amarelo, que está conectado na porta analógica A0, e um dos terminais do resistor.
- Conecte o outro terminal do resistor que está conectado na mesma coluna de furos do fio preto, este conectado no barramento (negativo) da matriz de contato, como mostra a Figura 5.14.

Passo 2: Código-fonte

Para inserir o código no ambiente Tinkercad, clique no botão ‘Código’, parte superior direita, e mude a opção de ‘Blocos’ para a opção ‘Texto’. Será aberta uma janela, perguntando se tem certeza que deseja fechar o editor de blocos. Clique em continuar.

```
1 int led = 7;           // define pino digital 7
2 int ldr = A0;          // define pino analogico A0
3 int valor_ldr = 0;     // guarda o valor lido do ldr
4
5 void setup()
6 {
7   pinMode(led, OUTPUT); // define pino 7 como OUTPUT
8   Serial.begin(9600);   // inicializa comunicação serial
9 }
10
11 void loop()
12 {
13   // ler o valor do ldr e armazena na variável valor_ldr
14   valor_ldr = analogRead(ldr);
15
16   Serial.println(valor_ldr); // mostra o valor no Monitor serial
17   if(valor_ldr < 500)       // verifica se o valor do ldr é menor que 500
18   {
19     digitalWrite(led, HIGH); // liga o led
20   }else{
21     digitalWrite(led, LOW);  // desliga o led
22   }
23 }
```

Figura 5.15: código-fonte do projeto.

No código da Figura 5.15, linha 1, é declarada uma variável do tipo de dado *int*, com o identificador chamado *led*, que recebe o valor 7. Essa variável, é responsável por armazenar o valor do pino digital 7, da placa Arduino. Na linha 2, é declarada uma variável do tipo de dado *int*, com o

identificador chamado *ldr*, que recebe o valor A0. Essa variável, é responsável por armazenar o valor do pino analógico A0.

Na linha 3, é declarada uma variável do tipo de dado *int*, com o identificador chamado *valor_ldr*. Essa variável, é responsável por guardar o valor lido do componente *LDR*. Na linha 5, a função *setup()*, será executada apenas uma vez, e é responsável por inicializar variáveis, configurar o modo dos pinos (*INPUT* ou *OUTPUT*), inicializar bibliotecas, etc.

Na linha 7, a função *pinMode()*, configura o pino especificado para funcionar como uma entrada ou saída e recebe dois parâmetros, o primeiro é o número do pino, o segundo é o modo do pino, que pode ser *INPUT* (entrada) e *OUTPUT* (saída). Neste exemplo da linha 7, o pino do *led* é configurado como *OUTPUT*.

Na linha 8, a função *begin()* é responsável por inicializar a comunicação Serial, recebendo o valor 9600 como parâmetro de entrada, esse valor corresponde a taxa de transferência em bits por segundo para transmissão serial. Na linha 11, a função *loop()*, é onde os códigos serão inseridos para serem executados repetidamente, enquanto a placa Arduino estiver ligada.

Na linha 14, a variável *valor_ldr* recebe o valor da leitura do *ldr*, feita pela função *analogRead()*. Esta função lê o valor de um pino analógico especificado, A0. Na linha 16, a função *println()*, mostra o valor do LDR, através da comunicação Serial, no monitor serial.

Na linha 17, se tem uma estrutura de seleção *if*. As estruturas de seleção ou decisão são utilizadas quando existe a necessidade de verificar condições para determinar quais instruções serão, ou não, executadas. Os testes de seleção também podem ser utilizados para verificar opções de escolha (PUGA e RISSETTI, 2016). Portanto, é verificado se a expressão '*valor_ldr < 500*' é verdadeira, se for verdadeira, a linha 19 é executada, ligando o *led*. Caso contrário, se a expressão for falsa, a linha 21 é executada, desligando o *led*.

O ciclo se repete a partir da função *loop()* até que a placa Arduino esteja sendo alimentada por uma fonte de energia ou enquanto a simulação estiver sendo executada.

Passo 3: Executando o projeto no Tinkercad

Para ver o projeto em execução no simulador Tinkercad, clique no botão ‘Iniciar simulação’, na parte superior direita da tela, como mostra a Figura 5.16. Ao abrir a janela do Monitor serial, é mostrado o valor atual do LDR.

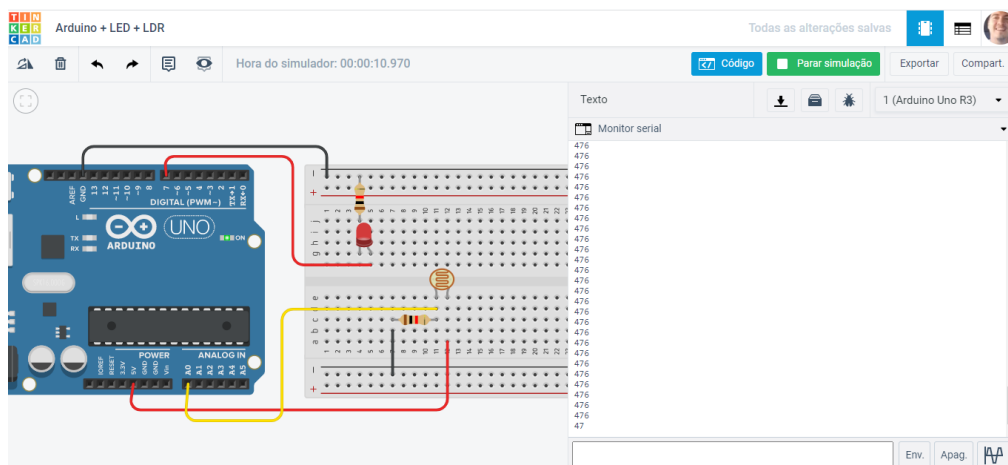


Figura 5.16: exemplo de projeto em execução.



Hora de praticar!



Prática 4: Monte um circuito com 1 (um) LDR e 2 (dois) LEDs, vermelho e verde. Quando o valor do LDR estiver menor do que 500, o Led verde irá acender e o Led vermelho irá apagar, caso contrário, se o valor do LDR estiver maior que 500, o Led vermelho irá acender e o Led verde irá apagar. Programe o Arduino e faça as conexões apropriadas dos componentes.



Usando Potenciômetro

Objetivos de aprendizagem

- Montagem do circuito e compreender o funcionamento de um potenciômetro;
- Controle de frequência de liga-desliga de um led através de um potenciômetro;
- Atribuir valor à uma variável através da leitura de um potenciômetro.

5.5 – Usando Potenciômetro

Este projeto demonstra o uso de um Potenciômetro, muito parecido com os controles de volumes de aparelhos de som antigo. O Potenciômetro é um componente analógico, por isso, deve-se ligá-lo em uma porta analógica da placa Arduino. O potenciômetro é um resistor variável no formato de um botão giratório que fornece um valor analógico. Se girarmos o potenciômetro, alteramos a resistência em cada lado do contato elétrico que vai conectado ao terminal central do botão (OLIVEIRA *et al*, 2018). O objetivo deste projeto é controlar a frequência para ligar e desligar um LED, através do potenciômetro e assim demonstrar seu funcionamento.

Material necessário

- 1 Arduino.
- 1 Led.
- 1 Resistor de 220 Ω .
- 1 Potenciômetro.
- 1 Matriz de contato.
- Jumper cable.

Passo 1: montagem do circuito

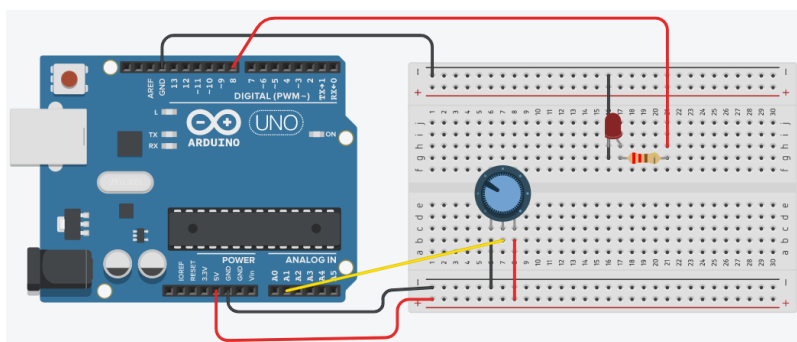


Figura 5.17: exemplo de circuito com Potenciômetro.

- Conecte o terminal positivo do led vermelho, na mesma coluna de furos que um dos terminais do resistor, o outro terminal do resistor é conectado na mesma coluna de furos que o fio vermelho que vai conectado na porta digital 8;
- Conecte o outro terminal negativo do led vermelho, na mesma coluna de furos que o fio preto, e este está conectado direto no barramento (negativo), que vai conectado direto na porta *GND* (*Ground*) da placa Arduino;

- O Potenciômetro tem três terminais, o terminal da direita é conectado na mesma coluna de furos que o fio vermelho, que está conectado na porta de 5V. O terminal do centro está conectado na mesma coluna de furos que o fio amarelo, e este conectado na porta analógica A1. O terminal da esquerda está conectado na mesma coluna de furos do fio preto, que está conectado no barramento (negativo) da matriz de contato, como mostra a Figura 5.17.

Passo 2: Código-fonte

Para inserir o código no ambiente Tinkercad, clique no botão ‘Código’, parte superior direita, e mude a opção de ‘Blocos’ para a opção ‘Texto’. Será aberta uma janela, perguntando se tem certeza que deseja fechar o editor de blocos. Clique em continuar.

```

1  int led_vermelho = 8;           // define pino digital 8
2  int potenciometro = A1;        // define pino analógico A1
3  int valor_potenciometro = 0;   // guarda o valor lido do potenciometro
4
5  void setup()
6  {
7      pinMode(led_vermelho, OUTPUT); // define pino 8 como OUTPUT
8      Serial.begin(9600);           // inicializa comunicação serial
9  }
10
11 void loop()
12 {
13     // ler o valor do potenciometro e armazena na variavel
14     valor_potenciometro = analogRead(potenciometro);
15
16     Serial.println(valor_potenciometro); // mostra o valor no monitor serial
17     digitalWrite(led_vermelho, HIGH);   // liga o led
18     delay(valor_potenciometro);          // espera o tempo em milissegundos
19     digitalWrite(led_vermelho, LOW);    // desliga o led
20     delay(valor_potenciometro);          // espera o tempo em milissegundos |
21 }

```

Figura 5.18: código-fonte do projeto.

No código da Figura 5.18, linha 1, é declarada uma variável do tipo de dado *int*, com o identificador chamado *led_vermelho*, que recebe o valor 8. Essa variável, é responsável por armazenar o valor do pino digital 8, da placa Arduino. Na linha 2, é declarada uma variável do tipo de dado *int*, com o identificador chamado *potenciometro*, que recebe o valor A1. Essa variável, é responsável por armazenar o valor do pino analógico A1.

Na linha 3, é declarada uma variável do tipo de dado *int*, com o identificador chamado *valor_potenciometro*. Essa variável, é responsável por guardar o valor lido do componente *Potenciômetro*. Na linha 5, a função *setup()*, será executada apenas uma vez, e é responsável por inicializar variáveis, configurar o modo dos pinos (*INPUT* ou *OUTPUT*), inicializar bibliotecas, etc.

Na linha 7, a função `pinMode()`, configura o pino especificado para funcionar como uma entrada ou saída e recebe dois parâmetros, o primeiro é o número do pino, o segundo é o modo do pino, que pode ser *INPUT* (entrada) e *OUTPUT* (saída). Neste exemplo da linha 7, o pino do *led_vermelho* é configurado como *OUTPUT*.

Na linha 8, a função `begin()` é responsável por inicializar a comunicação Serial, recebendo o valor 9600 como parâmetro de entrada, esse valor corresponde a taxa de transferência em bits por segundo para transmissão serial. Na linha 11, a função `loop()`, é onde os códigos serão inseridos para serem executados repetidamente, enquanto a placa Arduino estiver ligada.

Na linha 14, a variável *valor_potenciometro* recebe o valor da leitura do potenciômetro, feita pela função `analogRead()`. Esta função lê o valor de um pino analógico especificado, A1. Na linha 16, a função `println()`, mostra o valor do potenciômetro, no monitor serial.

Na linha 17, a função `digitalWrite()`, liga o led com o parâmetro *HIGH*. Na linha 18, a função `delay()` pausa o tempo em milissegundos, correspondente ao valor do potenciômetro. Na linha 19, a função `digitalWrite()`, desliga o led com o parâmetro *LOW*. Na linha 20, a função `delay()` pausa o tempo em milissegundos, correspondente ao valor do potenciômetro.

O ciclo se repete a partir da função `loop()` até que a placa Arduino esteja sendo alimentada por uma fonte de energia ou enquanto a simulação estiver sendo executada.

Passo 3: Executando o projeto no Tinkercad

Para ver o projeto em execução no simulador Tinkercad, clique no botão ‘Iniciar simulação’, na parte superior direita da tela, como mostra a Figura 5.19. Ao mover o giro do potenciômetro para direita ou para esquerda, percebe-se uma variação na frequência de liga e desliga do LED.

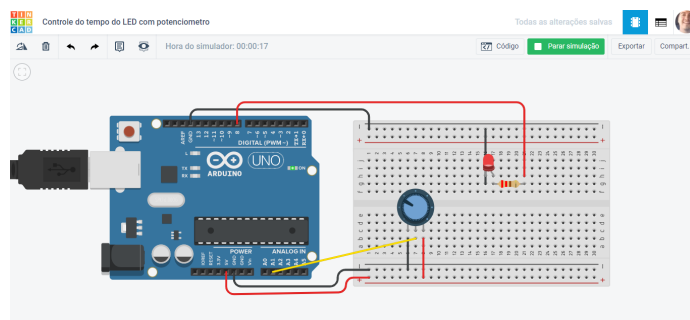


Figura 5.19: exemplo de projeto em execução.



Hora



de

praticar!

Prática 5: Monte um circuito com 1 (um) Potenciômetro e 2 (dois) LEDs, amarelo e verde. Quando o valor do Potenciômetro estiver menor do que 550, o Led verde irá acender e o Led amarelo irá apagar, caso contrário, se o valor do LDR estiver maior que 550, o Led amarelo irá acender e o Led verde irá apagar. Programe o Arduino e faça as conexões apropriadas dos componentes.



Usando LCD

Objetivos de aprendizagem

- Montagem do circuito e compreender o funcionamento de um LCD;
- Compreender o uso de bibliotecas;
- Uso de funções internas (*begin*, *setcursor*, *print*, *clear*) do LCD;
- Uso da função interna *analogWrite* da linguagem padrão;
- Fazer a leitura da comunicação serial utilizando a função interna *readString*;
- Uso do tipo de dado *String* para o tratamento de textos em variáveis;

5.6 – Usando LCD

Este projeto demonstra o uso de um *Liquid Crystal Display* (LCD), com 16 colunas e 2 linhas. Para o uso desse componente será utilizado a biblioteca *LiquidCrystal.h*, disponível no endereço <https://www.arduino.cc/en/Reference/LiquidCrystal>. Bibliotecas são conjunto de códigos pré-definidos para serem reutilizados com alguma finalidade, nesse caso, utilizaremos códigos para manipular dados do LCD. A montagem do display deve ser de acordo com sua especificação (*datasheet*), onde cada um dos pinos possui uma função específica (OLIVEIRA *et al*, 2018). O objetivo deste projeto é utilizar o LCD para mostrar os valores de um LDR no display.

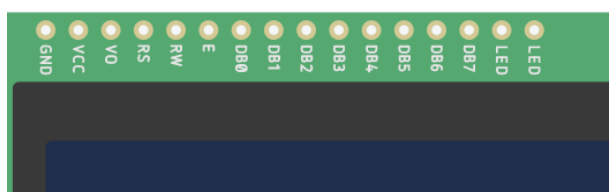


Figura 5.20: pinagem do LCD.

A Figura 5.20 mostra os 16 pinos de um LCD e a tabela 5.1 mostra o símbolo e a função para cada pino. No passo 1, demonstraremos como fazer a montagem e as ligações para cada pino do LCD.

Tabela 5.1: Função dos pinos de um LCD.

Pino	Símbolo	Função
1	VSS	GND(Alimentação)
2	VDD	5V(Alimentação)
3	VO	Ajuste de contraste
4	RS	Habilita / Desabilita seletor de registrador
5	R/W	Leitura / Escrita
6	E	Habilita escrita no LCD
7	DB0	Dado
8	DB1	Dado
9	DB2	Dado
10	DB3	Dado
11	DB4	Dado
12	DB5	Dado
13	DB6	Dado
14	DB7	Dado
15	A	5V (Backlight)
16	K	GND (Backlight)

Fonte: OLIVEIRA *et al* (2018).

Material necessário

- 1 Arduino.
- 1 LCD 16x2.
- 1 LDR.
- 1 Resistor 10 k Ω .
- 1 Resistor 1 k Ω .
- 1 Matriz de contato.
- Jumper cable.

Passo 1: montagem do circuito

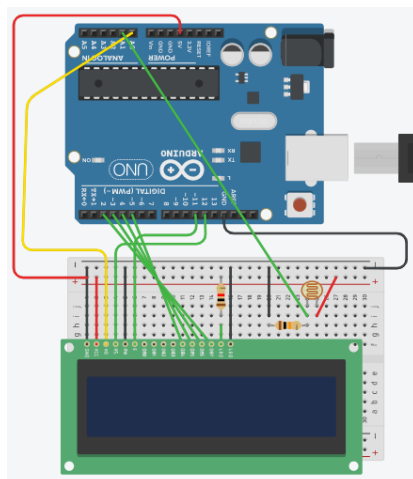


Figura 5.21: exemplo de circuito com LCD.

- A ligação dos pinos do LCD segue a seguinte sequência de montagem: pino 1 conectado ao GND; pino 2 conectado ao 5V; pino 3 conectado na porta analógica A0; pino 4 conectado na porta digital 12; pino 5 conectado ao GND; pino 6 conectado na porta digital 11; os pinos 7, 8, 9 e 10 não serão utilizados; pino 11 conectado na porta digital 5; pino 12 conectado na porta digital 4; pino 13 conectado na porta digital 3; pino 14 conectado na porta digital 2; pino 15 conectado ao 5V com um resistor de 1k Ω ; pino 16 conectado ao GND;
- Conecte um dos terminais do LDR na mesma coluna de furos que o fio vermelho, que está conectado na porta de 5V. Conecte o outro terminal do LDR na mesma coluna de furos que o fio verde, que está conectado na porta analógica A1, e um dos terminais do resistor de 10 k Ω ;
- Conecte o outro terminal do resistor de 10k Ω na mesma coluna de furos do fio preto, que está conectado no barramento (negativo) da matriz de contato, como mostra a Figura 5.21.

Passo 2: Código-fonte

Para inserir o código no ambiente Tinkercad, clique no botão ‘Código’, parte superior direita, e mude a opção de ‘Blocos’ para a opção ‘Texto’. Será aberta uma janela, perguntando se tem certeza que deseja fechar o editor de blocos. Clique em continuar.

```
1 #include <LiquidCrystal.h>
2
3 int ldr = A1;
4 int valor_ldr;
5 String mensagem = "";
6 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
7
8 void setup()
9 {
10   Serial.begin(9600);
11   lcd.begin(16, 2); // inicializa o lcd com 16 colunas e 2 linhas
12   analogWrite(A0, 127); // valor analógico de 0 - 127 para ajuste de contraste no lcd
13   lcd.setCursor(0, 0); // coluna 0 e linha 0
14   lcd.print("Usando LCD"); // mostra os dados no lcd
15   lcd.setCursor(0, 1); // coluna 0 e linha 1
16   lcd.print(":");
17   delay(2000);
18 }
19
20 void loop()
21 {
22   if(Serial.available() > 0)
23   {
24     mensagem = Serial.readString();
25     lcd.clear(); // limpa a tela do lcd
26     lcd.setCursor(0, 0); // coluna 0 e linha 0
27     lcd.print(mensagem);
28     delay(5000);
29   }else{
30     valor_ldr = analogRead(ldr);
31     lcd.clear(); // limpa a tela do lcd
32     lcd.setCursor(0, 0); // coluna 0 e linha 0
33     lcd.print("Valor LDR");
34     lcd.setCursor(0, 1); // coluna 0 e linha 1
35     lcd.print(valor_ldr);
36     delay(1000);
37   }
38 }
```

Figura 5.22: código-fonte do projeto.

No código da Figura 5.22, linha 1, é declarada a inclusão da biblioteca *LiquidCrystal.h*. A diretiva *#include* é usada para incluir bibliotecas externas. As bibliotecas são um conjunto de códigos pré-definidos, com funções definidas para serem reutilizadas. A biblioteca *LiquidCrystal*, possui funções para manipulações de dados com o componente *LCD*.

Na linha 3, é declarada uma variável do tipo de dado *int*, com o identificador chamado *ldr*, que recebe o valor A1. Essa variável, é responsável por armazenar o valor do pino analógico A1, da placa Arduino. Na linha 4, é declarada uma variável do tipo de dado *int*, com o identificador chamado *valor_ldr*, responsável por armazenar o valor da leitura do *ldr*.

Na linha 5, é declarada uma variável do tipo de dado *String*, com o identificador chamado *mensagem*, responsável por armazenar a mensagem recebida pela comunicação serial, para ser mostrada no display *LCD*. Na linha 6, é criado o objeto do tipo *LiquidCrystal*, chamado *lcd*, que recebe como parâmetro os pinos digitais 12, 11, 5, 4, 3, 2.

Na linha 8, a função *setup()*, será executada apenas uma vez, e é responsável por inicializar variáveis, configurar o modo dos pinos (*INPUT* ou *OUTPUT*), inicializar bibliotecas, etc. Na linha 10, é inicializado a comunicação serial. Na linha 11, a função *begin()* inicializa a interface para a tela *LCD* e especifica as dimensões (largura e altura) da tela. Na linha 12, a função *analogWrite()* ajusta o contraste do pino A0, para o valor 127.

Na linha 13, a função *setCursor()*, posiciona o cursor do *LCD*, ou seja, define o local em que o texto subsequente escrito no *LCD* será exibido, coluna 0 e linha 0. Na linha 14, a função *print()*, mostra o texto “Usando *LCD*” na tela. Na linha 15, o cursor é posicionado na coluna 0 e linha 1 e novamente na linha 16, é mostrado o texto “:)” na tela. Na linha 17, a função *delay()* espera 2 segundos, exibindo os textos na tela do *LCD*.

Na linha 20, a função *loop()*, é onde os códigos serão inseridos para serem executados repetidamente, enquanto a placa *Arduino* estiver ligada. Na linha 22, se tem uma estrutura de seleção *if*. As estruturas de seleção ou decisão são utilizadas quando existe a necessidade de verificar condições para determinar quais instruções serão, ou não, executadas. Os testes de seleção também podem ser utilizados para verificar opções de escolha (PUGA e RISSETTI, 2016). Portanto, é verificado se a expressão ‘*Serial.available() > 0*’ é verdadeira, se for verdadeira, a linha 24 até 28 é executada. Caso contrário, se a expressão for falsa, a linha 30 até 36 é executada.

A função *available()*, retorna o número de bytes (caracteres) disponíveis para leitura da porta serial. Portanto, caso a comunicação serial tenha caracteres disponíveis, a função *readString()*, da linha 24, lê caracteres do *buffer* serial e os armazena na variável *mensagem*. Na linha 25, a função *clear()* limpa a tela do *lcd*. Na linha 26, é posicionado o cursor na coluna 0 e linha 0. Na linha 27, é mostrado a mensagem na tela e em seguida, a função *delay()* espera 5 segundos, exibindo os textos na tela do *LCD*.

Na linha 30, a função *analogRead()* lê o valor do *LDR* e armazena-o na variável *valor_ldr*. Na linha 31, a função *clear()* limpa a tela, em seguida é posicionado o cursor para a coluna 0 e linha 0. O texto “Valor *LDR*” é mostrado na tela e novamente o cursor é posicionado na coluna 0 e linha 1, mostrando o valor do componente *LDR*.

O ciclo se repete a partir da função *loop()* até que a placa *Arduino* esteja sendo alimentada por uma fonte de energia ou enquanto a simulação estiver sendo executada.

Passo 3: Executando o projeto no Tinkercad

Para ver o projeto em execução no simulador Tinkercad, clique no botão ‘Iniciar simulação’, na parte superior direita da tela, como mostra a Figura 5.23. Ao mover a barra do componente LDR, para direita ou para esquerda, percebe-se o valor sendo exibido na tela do LCD.

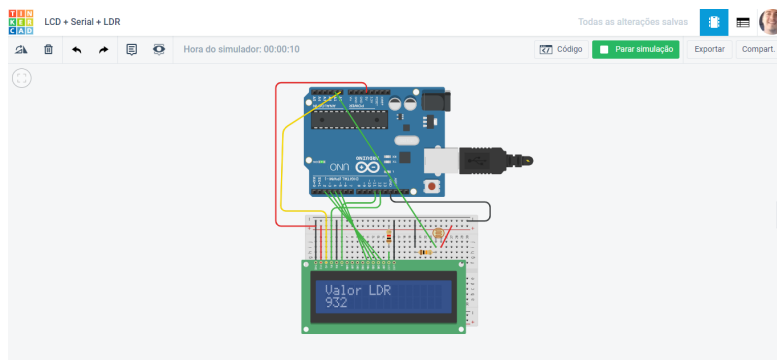


Figura 5.23: exemplo de projeto em execução.



Hora de praticar!



Prática 6: Monte um circuito com 1 (um) LCD e 2 (dois) LEDs, amarelo e verde. O circuito irá fazer uma comunicação serial e através do monitor serial, receberá a mensagem digitada pelo usuário. Se a mensagem for igual a "liga led verde", o LED verde irá ligar e mostrará no LCD a mensagem "Led verde ligado!", caso contrário, se a mensagem digitada pelo usuário for "desliga led verde", o LED verde irá desligar e mostrará no LCD a mensagem "Led verde desligado!". Da mesma forma, Se a mensagem for igual a "liga led amarelo", o LED amarelo irá ligar e mostrará no LCD a mensagem "Led amarelo ligado!", caso contrário, se a mensagem digitada pelo usuário for "desliga led amarelo", o LED amarelo irá desligar e mostrará no LCD a mensagem "Led amarelo desligado!". Programe o Arduino e faça as conexões apropriadas dos componentes.



Usando sensor de presença

Objetivos de aprendizagem

- Montagem do circuito e compreender o funcionamento de um sensor de presença;
- Atribuir valor à uma variável através da leitura de um sensor de presença;
- Uso de um operador de comparação em uma expressão lógica.

5.7 – Usando sensor de presença

Este projeto demonstra o uso de um sensor de presença, chamado PIR. Este sensor utiliza infravermelho para detectar movimentos. No módulo pode-se ajustar a sensibilidade e o tempo que o sinal será enviado ao Arduino, ao se detectar algum movimento o sensor envia o sinal 1 (*HIGH*) para o Arduino (OLIVEIRA *et al*, 2018). O objetivo deste projeto é demonstrar o funcionamento do sensor PIR, caso detecte algum movimento, acende o LED vermelho, caso contrário, não detecte movimento, acende o LED verde.

Material necessário

- 1 Arduino.
- 2 Led.
- 2 Resistores de 1 k Ω .
- 1 sensor PIR.
- 1 Matriz de contato.
- Jumper cable.

Passo 1: montagem do circuito

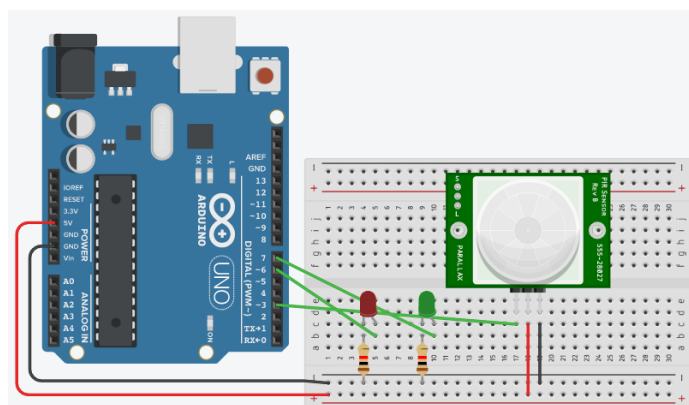


Figura 5.24: exemplo de circuito com sensor PIR.

- Conecte o terminal negativo do led vermelho, na mesma coluna de furos que um dos terminais do resistor e o outro terminal do mesmo resistor vai conectado no barramento negativo da protoboard, e esse barramento (negativo) está conectado, através do fio preto, direto na porta *GND* (*Ground*) da placa Arduino;
- Conecte o terminal positivo do led vermelho, na mesma coluna de furos do fio verde, que está conectado na porta digital 6;

- Conecte o terminal negativo do led verde, na mesma coluna de furos que um dos terminais do resistor e o outro terminal do mesmo resistor é conectado no barramento negativo da protoboard, que através do fio preto vai conectado na porta GND;
- Conecte o terminal positivo do led verde, na mesma coluna de furos do fio verde, que vai conectado na porta digital 7;
- O sensor PIR possui três terminais, o terminal da direita vai conectado no barramento negativo da protoboard, o terminal do centro vai conectado no barramento positivo da protoboard, o terminal da esquerda vai conectado direto na porta digital 3, como mostra a Figura 5.24.

Passo 2: Código-fonte

Para inserir o código no ambiente Tinkercad, clique no botão ‘Código’, parte superior direita, e mude a opção de ‘Blocos’ para a opção ‘Texto’. Será aberta uma janela, perguntando se tem certeza que deseja fechar o editor de blocos. Clique em continuar.

```

1 int led_vermelho = 6; // define pino digital 6
2 int led_verde = 7; // define pino digital 7
3 int sensorPIR = 3; // define pino digital 3
4 int acionamento; // guarda o valor lido do sensor PIR
5
6 void setup()
7 {
8   pinMode(led_vermelho, OUTPUT); // define pino 6 como OUTPUT
9   pinMode(led_verde, OUTPUT); // define pino 7 como OUTPUT
10  pinMode(sensorPIR, INPUT); // define pino 3 como INPUT
11  Serial.begin(9600); // inicializa comunicação serial
12 }
13
14 void loop()
15 {
16   acionamento = digitalRead(sensorPIR); // ler o valor do sensor PIR e armazena na variável
17   if(acionamento == LOW){ // verifica se houve detecção de algum objeto
18     digitalWrite(led_verde, HIGH); // liga led verde
19     digitalWrite(led_vermelho, LOW); // desliga led vermelho
20     Serial.println("Sem movimento"); // mostra texto no monitor serial
21   }else{
22     digitalWrite(led_verde, LOW); // desliga led verde
23     digitalWrite(led_vermelho, HIGH); // liga led vermelho
24     Serial.println("Movimento detectado"); // mostra texto no monitor serial
25   }
26   delay(1000); // espera 1 segundo
27 }

```

Figura 5.25: código-fonte do projeto.

No código da Figura 5.25, linha 1, é declarada uma variável do tipo de dado *int*, com o identificador chamado *led_vermelho*, que recebe o valor 6. Essa variável, é responsável por armazenar o valor do pino digital 6, da placa Arduino. Na linha 2, é declarada uma variável do tipo

de dado *int*, com o identificador chamado *led_verde*, que recebe o valor 7. Essa variável, é responsável por armazenar o valor do pino digital 7.

Na linha 3, é declarada uma variável do tipo de dado *int*, com o identificador chamado *sensorPIR*, que recebe o valor 3. Essa variável, é responsável por armazenar o valor do pino digital 3. Na linha 4, é declarada uma variável do tipo de dado *int*, com o identificador chamado *acionamento*. Essa variável, é responsável por guardar o valor lido do sensor PIR. Na linha 6, a função *setup()*, será executada apenas uma vez, é responsável por inicializar variáveis, configurar o modo dos pinos (*INPUT* ou *OUTPUT*), inicializar bibliotecas, etc.

Na linha 8, a função *pinMode()*, configura o pino especificado para funcionar como uma entrada ou saída e recebe dois parâmetros, o primeiro é o número do pino, o segundo é o modo do pino, que pode ser *INPUT* (entrada) e *OUTPUT* (saída). Neste exemplo da linha 8, o pino do *led_vermelho* é configurado como *OUTPUT*. Na linha 9, o pino do *led_verde* é configurado como *OUTPUT*. Na linha 10, o pino do *sensorPIR* é configurado como *INPUT*.

Na linha 11, a função *begin()* é responsável por inicializar a comunicação serial, recebendo o valor 9600 como parâmetro de entrada, esse valor corresponde a taxa de transferência em bits por segundo para transmissão serial. Na linha 14, a função *loop()*, é onde os códigos serão inseridos para serem executados repetidamente, enquanto a placa Arduino estiver ligada.

Na linha 16, a variável *acionamento* recebe o valor da leitura do sensor PIR, feita pela função *digitalRead()*. Esta função faz a leitura de um pino digital especificado, pino 3. Na linha 17, é verificado se houve detecção de algum objeto, caso seja verdadeira a expressão, as linhas 18 a 20 são executadas, caso contrário, as linhas 22 a 24 são executadas.

Na linha 18, a função *digitalWrite()*, liga o *led_verde* com o parâmetro *HIGH*. Na linha 19, a função *digitalWrite()*, desliga o *led_vermelho* com o parâmetro *LOW*. Na linha 20, a função *println()* envia o texto “*sem movimento*” para o monitor serial. Na linha 22, a função *digitalWrite()*, desliga o *led_verde* com o parâmetro *LOW*. Na linha 23, a função *digitalWrite()*, liga o *led_vermelho* com o parâmetro *HIGH*. Na linha 24, a função *println()* envia o texto “*movimento detectado*” para o monitor serial.

Na linha 26, a função *delay()* aguarda por 1 segundo e o ciclo se repete a partir da função *loop()* até que a placa Arduino esteja sendo alimentada por uma fonte de energia ou enquanto a simulação estiver sendo executada.

Passo 3: Executando o projeto no Tinkercad

Para ver o projeto em execução no simulador Tinkercad, clique no botão ‘*Iniciar simulação*’, na parte superior direita da tela, como mostra a Figura 5.26. Ao clicar no sensor de presença, percebe-se uma região destacada com um objeto que pode ser movido. Movendo o objeto na região de detecção, é acionado o sensor e o led vermelho é conectado, caso não detecte movimento, o led verde é conectado.

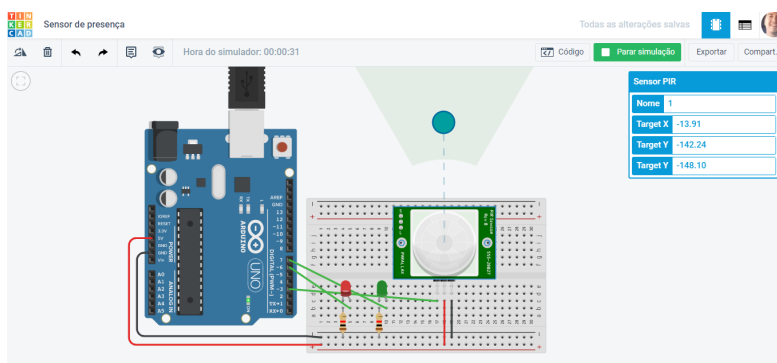


Figura 5.26: exemplo de projeto em execução.



Hora de praticar!



Prática 7: Monte um circuito com 1 (um) sensor PIR e 1 (um) LCD. Se um objeto for detectado pelo sensor de presença, a seguinte mensagem "Objeto detectado!" irá ser mostrada no display do LCD, caso contrário, se nenhum objeto for detectado pelo sensor PIR, o LCD mostrará a seguinte mensagem "Nenhum objeto detectado!". Programe o Arduino e faça as conexões apropriadas dos componentes.



Usando Buzzer

Objetivos de aprendizagem

- Montagem do circuito e compreender o funcionamento de um Buzzer;
- Acionar o Buzzer através da leitura de um potenciômetro;
- Uso da função interna *tone*.

5.8 – Usando Buzzer

Este projeto demonstra o uso do componente chamado Buzzer. Este componente é um dispositivo que gera sinais sonoros. Para a emissão do som, o buzzer vibra através de um oscilador. Essa oscilação é determinada por uma frequência, que por sua vez define um som específico (OLIVEIRA *et al*, 2018). O objetivo deste projeto é demonstrar o funcionamento do Buzzer. Neste projeto será utilizado um potenciômetro para controlar a frequência daquele componente. Para isso, a função *tone(pino, frequência, duração)* é responsável por receber como parâmetro, o *pino* que irá gerar a frequência, a *frequência* é definida em hertz e a *duração* é definida em milissegundos.

Material necessário

- 1 Arduino.
- 1 Potenciômetro.
- 1 Resistor de 220 Ω .
- 1 Buzzer.
- 1 Matriz de contato.
- Jumper cable.

Passo 1: montagem do circuito

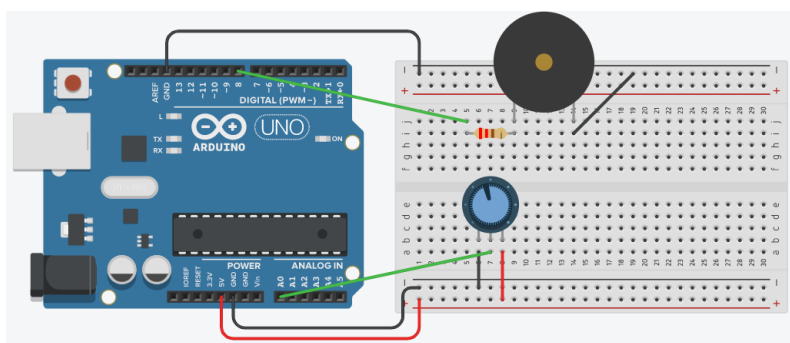


Figura 5.27: exemplo de circuito com Buzzer.

- Conecte o terminal direito do Buzzer na mesma coluna de furos do fio preto que vai conectado no barramento negativo da protoboard, e este barramento (negativo) está conectado, através do fio preto, direto na porta *GND* (*Ground*) da placa Arduino;
- Conecte o terminal positivo do Buzzer, na mesma coluna de furos que um dos terminais do resistor e o outro terminal do mesmo resistor é conectado na mesma coluna de furos que o fio verde, que está conectado na porta digital 8;

- O Potenciômetro tem três terminais, o terminal da direita é conectado na mesma coluna de furos que o fio vermelho, que está conectado na porta de 5V. O terminal do centro está conectado na mesma coluna de furos que o fio verde, e este está conectado na porta analógica A0. O terminal da esquerda está conectado na mesma coluna de furos do fio preto, que está conectado no barramento (negativo) da matriz de contato, que por sua vez está conectado pelo fio preto na porta *GND*, como mostra a Figura 5.27.

Passo 2: Código-fonte

Para inserir o código no ambiente Tinkercad, clique no botão ‘Código’, parte superior direita, e mude a opção de ‘Blocos’ para a opção ‘Texto’. Será aberta uma janela, perguntando se tem certeza que deseja fechar o editor de blocos. Clique em continuar.

```
1  int buzzer = 8;           // Pino ao qual o buzzer está conectado
2  int potenciometro = A0;   // Pino ao qual o pontenciômetro está conectado
3  int valor_pot = 0;        // guarda o valor lido do potenciômetro
4
5  void setup()
6  {
7      pinMode(buzzer, OUTPUT); // pino 8 como modo OUTPUT
8  }
9
10 void loop()
11 {
12     // ler e armazena o valor do potenciometro
13     valor_pot = analogRead(potenciometro);
14     // ajusta a frequência do buzzer
15     tone(buzzer, valor_pot, 500);
16     // espera 1 segundo
17     delay(1000);
18 }
```

Figura 5.28: código-fonte do projeto.

No código da Figura 5.28, linha 1, é declarada uma variável do tipo de dado *int*, com o identificador chamado *buzzer*, que recebe o valor 8. Essa variável, é responsável por armazenar o valor do pino digital 8, da placa Arduino. Na linha 2, é declarada uma variável do tipo de dado *int*, com o identificador chamado *potenciometro*, que recebe o valor A0. Essa variável, é responsável por armazenar o valor do pino analógico A0.

Na linha 3, é declarada uma variável do tipo de dado *int*, com o identificador chamado *valor_pot*. Essa variável, é responsável por guardar o valor lido do componente *Potenciômetro*. Na linha 5, a função *setup()*, será executada apenas uma vez, e é responsável por inicializar variáveis, configurar o modo dos pinos (*INPUT* ou *OUTPUT*), inicializar bibliotecas, etc.

Na linha 7, a função `pinMode()`, configura o pino especificado para funcionar como uma entrada ou saída e recebe dois parâmetros, o primeiro é o número do pino, o segundo é o modo do pino, que pode ser `INPUT` (entrada) e `OUTPUT` (saída). Neste exemplo da linha 7, o pino do *buzzer* é configurado como `OUTPUT`.

Na linha 10, a função `loop()`, é onde os códigos serão inseridos para serem executados repetidamente, enquanto a placa Arduino estiver ligada. Na linha 13, a variável `valor_pot` recebe o valor da leitura do potenciômetro, feita pela função `analogRead()`. Esta função lê o valor de um pino analógico especificado, pino A0. Na linha 15, a função `tone()`, ajusta o pino 8 com a frequência igual ao valor do potenciômetro, pelo tempo de 500 milissegundos.

Na linha 17, a função `delay()` pausa o tempo em 1000 milissegundos, equivalente a 1 segundo. O ciclo se repete a partir da função `loop()` até que a placa Arduino esteja sendo alimentada por uma fonte de energia ou enquanto a simulação estiver sendo executada.

Passo 3: Executando o projeto no Tinkercad

Para ver o projeto em execução no simulador Tinkercad, clique no botão ‘Iniciar simulação’, na parte superior direita da tela, como mostra a Figura 5.29. Ao mover o giro do potenciômetro para direita ou para esquerda, percebe-se uma variação na frequência do Buzzer.

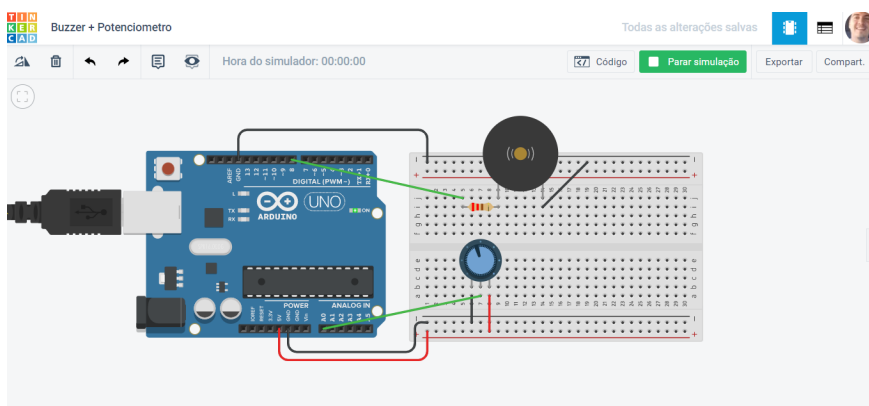


Figura 5.29: exemplo de projeto em execução.



Hora de praticar!



Prática 8: Monte um circuito com 1 (um) Buzzer e 1 (um) sensor de presença. Se um objeto for detectado pelo sensor de presença, o Buzzer irá emitir um som com frequência igual a 570 Hz e duração 1000 milissegundos, caso contrário, se nenhum objeto for detectado pelo sensor PIR, o Buzzer irá emitir um som com frequência igual a 20 Hz e duração 500 milissegundos. Programe o Arduino e faça as conexões apropriadas dos componentes.



Usando sensor de temperatura

Objetivos de aprendizagem

- Montagem do circuito e compreender o funcionamento de um sensor de temperatura;
- Atribuir valor à uma variável através da leitura de um sensor de temperatura;
- Usar estruturas de seleção encadeadas.

5.9 – Usando sensor de temperatura

Este projeto demonstra o uso do componente sensor de temperatura. Este componente faz a verificação da temperatura de um ambiente. O objetivo deste projeto é a criação de um circuito em que a partir de uma temperatura x , um led a acenda e a partir de uma temperatura y , um led b acenda.

Material necessário

- 1 Arduino.
- 1 sensor de temperatura.
- 2 Resistores de 220 Ω .
- 2 LED.
- 1 Matriz de contato.
- Jumper cable.

Passo 1: montagem do circuito

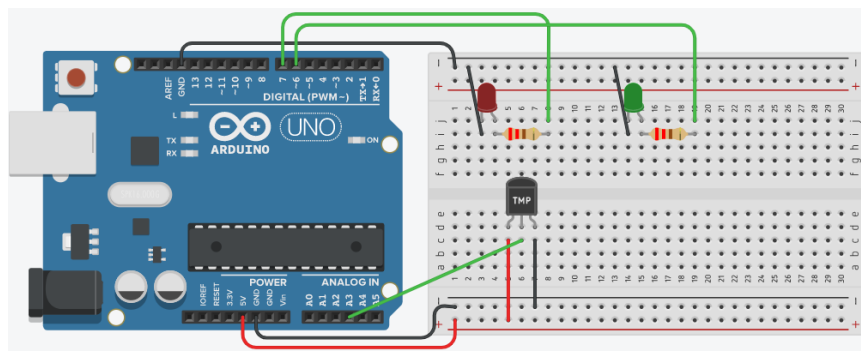


Figura 5.30: exemplo de circuito com sensor de temperatura.

- Conecte o terminal positivo do led vermelho, na mesma coluna de furos que um dos terminais do resistor e o outro terminal do mesmo resistor é conectado na mesma coluna de furos que o fio verde que vai conectado na porta digital 7;
- Conecte o terminal negativo do led vermelho, na mesma coluna de furos do fio preto, que vai conectado no barramento negativo da protoboard, e esse barramento (negativo) está conectado direto na porta *GND* (*Ground*) da placa Arduino;
- Conecte o terminal positivo do led verde, na mesma coluna de furos que um dos terminais do resistor e o outro terminal do mesmo resistor é conectado na mesma coluna de furos que o fio verde que vai conectado na porta digital 6;

- Conecte o terminal negativo do led verde, na mesma coluna de furos do fio preto, que vai conectado no barramento negativo da protoboard;
- O sensor de temperatura tem três terminais, o terminal da esquerda é conectado na mesma coluna de furos que o fio vermelho, que está conectado na porta de 5V. O terminal do centro está conectado na mesma coluna de furos que o fio verde, e este está conectado na porta analógica A3. O terminal da direita está conectado na mesma coluna de furos do fio preto, que está conectado no barramento (negativo) da matriz de contato, que por sua vez está conectado pelo fio preto na porta *GND*, como mostra a Figura 5.30.

Passo 2: Código-fonte

Para inserir o código no ambiente Tinkercad, clique no botão ‘Código’, parte superior direita, e mude a opção de ‘Blocos’ para a opção ‘Texto’. Será aberta uma janela, perguntando se tem certeza que deseja fechar o editor de blocos. Clique em continuar.

```

1  int tmp = A3;           // define pino analógico A3
2  int led_vermelho = 7;   // define pino digital 7
3  int led_verde = 6;      // define pino digital 6
4  int leitura_tmp;        // guarda o valor lido da temperatura
5
6  void setup()
7  {
8      pinMode(led_vermelho, OUTPUT); // pino 7 OUTPUT
9      pinMode(led_verde, OUTPUT);    // pino 6 OUTPUT
10     Serial.begin(9600);             // inicializa comunicação serial
11 }
12
13 void loop()
14 {
15     leitura_tmp = analogRead(tmp);   // armazena o valor lido da temperatura
16     Serial.println(leitura_tmp);     // mostra no monitor serial a temperatura
17
18     if(leitura_tmp >= 205)           // verifica se temperatura é maior que 205
19     {
20         digitalWrite(led_vermelho, HIGH); // liga led vermelho
21         digitalWrite(led_verde, LOW);     // desliga led verde
22     } else if(leitura_tmp < 205){     // verifica se temperatura é menor que 205
23         digitalWrite(led_verde, HIGH);   // liga led verde
24         digitalWrite(led_vermelho, LOW); // desliga led vermelho
25     }
26     delay(1000); // espera 1 segundo
27 }

```

Figura 5.31: código-fonte do projeto.

No código da Figura 5.31, linha 1, é declarada uma variável do tipo de dado *int*, com o identificador chamado *tmp*, que recebe o valor A3. Essa variável, é responsável por armazenar o valor do pino analógico A3, da placa Arduino. Na linha 2, é declarada uma variável do tipo de dado

int, com o identificador chamado *led_vermelho*, que recebe o valor 7. Essa variável, é responsável por armazenar o valor do pino digital 7.

Na linha 3, é declarada uma variável do tipo de dado *int*, com o identificador chamado *led_verde*, que recebe o valor 6. Essa variável, é responsável por armazenar o valor do pino digital 6. Na linha 4, é declarada uma variável do tipo de dado *int*, com o identificador chamado *leitura_tmp*. Essa variável, é responsável por guardar o valor lido do sensor de temperatura. Na linha 6, a função *setup()*, será executada apenas uma vez, é responsável por inicializar variáveis, configurar o modo dos pinos (*INPUT* ou *OUTPUT*), inicializar bibliotecas, etc.

Na linha 8, a função *pinMode()*, configura o pino especificado para funcionar como uma entrada ou saída e recebe dois parâmetros, o primeiro é o número do pino, o segundo é o modo do pino, que pode ser *INPUT* (entrada) e *OUTPUT* (saída). Neste exemplo da linha 8, o pino do *led_vermelho* é configurado como *OUTPUT*. Na linha 9, o pino do *led_verde* é configurado como *OUTPUT*.

Na linha 10, a função *begin()* é responsável por inicializar a comunicação serial, recebendo o valor 9600 como parâmetro de entrada, esse valor corresponde a taxa de transferência em bits por segundo para transmissão serial. Na linha 13, a função *loop()*, é onde os códigos serão inseridos para serem executados repetidamente, enquanto a placa Arduino estiver ligada.

Na linha 15, a variável *leitura_tmp* recebe o valor da leitura do sensor de temperatura, feita pela função *analogRead()*. Esta função faz a leitura de um pino analógico especificado, pino A3. Na linha 16 é mostrado no monitor serial o valor da temperatura. Na linha 18, é verificado se a temperatura é maior ou igual a 205, caso seja verdadeira a expressão, as linhas 20 e 21 serão executadas, caso contrário, as linhas 23 e 24 serão executadas.

Na linha 20, a função *digitalWrite()*, liga o *led_vermelho* com o parâmetro *HIGH*. Na linha 21, a função *digitalWrite()*, desliga o *led_verde* com o parâmetro *LOW*. Na linha 23, a função *digitalWrite()*, liga o *led_verde* com o parâmetro *HIGH*. Na linha 24, a função *digitalWrite()*, desliga o *led_vermelho* com o parâmetro *LOW*.

Na linha 26, a função *delay()* aguarda por 1 segundo e o ciclo se repete a partir da função *loop()* até que a placa Arduino esteja sendo alimentada por uma fonte de energia ou enquanto a simulação estiver sendo executada.

Passo 3: Executando o projeto no Tinkercad

Para ver o projeto em execução no simulador Tinkercad, clique no botão ‘*Iniciar simulação*’, na parte superior direita da tela, como mostra a Figura 5.32.

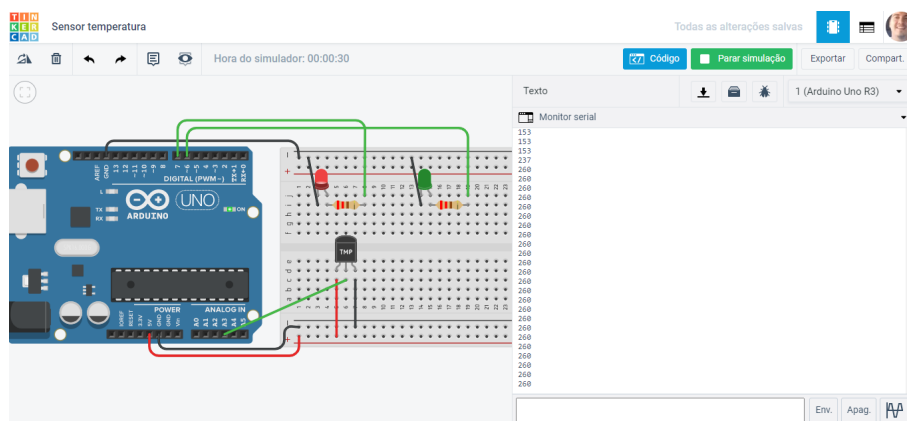
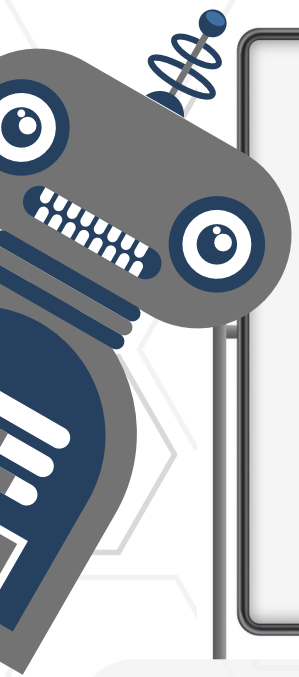


Figura 5.32: exemplo de projeto em execução.



Hora de praticar!



Prática 9: Monte um circuito com 1 (um) sensor de temperatura e 1 (um) Buzzer. Se a temperatura for maior que 200, o Buzzer irá emitir um som com frequência igual a 800 Hz e duração 50 milissegundos, caso contrário, se a temperatura for menor que 200, o Buzzer irá emitir um som com frequência igual a 10 Hz e duração 1000 milissegundos. Programe o Arduino e faça as conexões apropriadas dos componentes.

REFERÊNCIAS

- AQUINO, L.M.; FARIAS, M.P.; CRISPIM, I.V.; PINTO, V.P.; VASCONCELOS, F.H.L. Proposta de um curso semipresencial de robótica educacional utilizando a plataforma Arduino. *Revista Principia*, João Pessoa, maio. 2017
- BANZI, Massimo; SHILOH, Michael. *Primeiros passos com o Arduino: a plataforma de prototipagem eletrônica open source*. 2. ed. São Paulo: Novatec, 2015. 235 p. ISBN 978-85-7522-435-9.
- CAMARGO, Fausto; DAROS, Thuinie. *A sala de aula inovadora: estratégias pedagógicas para fomentar o aprendizado ativo*. Porto Alegre: Penso, 2018. 123p.
- CAMPOS, F. R. A robótica para uso educacional. São Paulo: Senac, 2019.
- CHITOLINA, Renati Fronza; NORONHA, Fabrícia Py Tortelli; BACKES, Luciana. A Robótica Educacional como tecnologia potencializadora da aprendizagem: das ciências da natureza às ciências da computação. *Educação, Formação & Tecnologias*, 9 (2), 56-65, 2016.
- COSTA, M.A.F.; COSTA M.F.B. *Projeto de pesquisa: entenda e faça*. Petrópolis: Vozes, 2011. 136p.
- FILIFELOP. *O que é Arduino*. Disponível em <https://www.filipeflop.com/blog/o-que-e-arduino/>. Acesso em 01 nov. de 2019.
- FRIEDRICH, Ronaldo Vaz; SANTOS, Daniela Scherer; KELLER, Rodrigo dos Santos; PUNTEL, Márcio Daniel; BIASOLI, Daniel. Proposta Metodológica para a Inserção ao Ensino de Lógica de Programação com Logo e Lego Mindstorms. *Anais do 23o Simpósio Brasileiro de Informática na Educação (SBIE 2012)*, ISSN 2316-6533 Rio de Janeiro, 2012.
- JUNIOR, Luiz Alberto Silva; LEÃO, Marcelo Brito Carneiro. O software Atlas.ti como recurso para a análise de conteúdo: analisando a robótica no Ensino de Ciências em teses brasileiras. *Ciênc. Educ.*, Bauru, v. 24, n. 3, p. 715-728, 2018.
- JUNIOR, Nacim Miguel Francisco; VASQUES, Carla K. Diálogos entre a robótica educacional e a sala de aula: um estudo de caso. *Directory of Open Access Journals (DOAJ)*, Vol.6(1), pp.3-20, 2010.
- MARCONI, M. A.; LAKATOS, E. M. Fundamentos de metodologia científica. 8. ed. São Paulo: Atlas, 2020.
- MCROBERTS, Michael. *Arduino Básico*. 2. ed. São Paulo: Novatec, 2015. 506 p. ISBN 978-85-7522-404-5.
- MILL, Daniel; CÉSAR, Danilo. Robótica pedagógica livre: sobre inclusão sócio-digital e democratização do conhecimento. *PERSPECTIVA*, Florianópolis, v. 27, n. 1, 217-248, 2009.

MORALES, Andréa Cantarelli; GIACOMELLI, Patricia; COSTA, Gabriela Martins da. Relações entre a Robótica Educacional e a Física do Ensino Médio. *SCIENTIA CUM INDUSTRIA*, V. 5, N. 2, PP. 121 —128, 2017.

NUNES, Edinéia G.; ARAUJO, Alex F.; PILASTRI, Andre L.; GULO, Carlos A. S. J. Ensino de Programação utilizando Robótica Educacional. *DataCite* 10.13140/RG.2.1.3283.1208, 2013.

OLIVEIRA, C. L. V.; ZANETTI, H. A. P.; NABARRO, C. B. M; GONÇALVES, J. A. V.; Aprenda Arduino: uma abordagem prática. Duque de Caxias: Katzen Editora, 2018.

PINTO, M.C. *Aplicação de Arquitetura Pedagógica em curso de robótica educacional com Hardware Livre*. 2011. 158f. Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro. 2011.

PLATT, Charles. Eletrônica para Makers: um manual prático para o novo entusiasta de eletrônica. São Paulo: Novatec, 2016.

PUGA, Sandra; RISSETTI, Gerson. *Lógica de programação e estruturas de dados*. São Paulo: Pearson Education do Brasil, 2016. 274p.

SENRA, Clarice; BRAGA, Marco; MEDINA, Márcio. A formação de uma comunidade de prática a partir da robótica educacional. *X Congreso Internacional sobre Investigación en didáctica de las Ciencias*, ISSN (DIGITAL): 2174-6486, 2017.

TORRES, V. P.; AROCA, R. V.; BURLAMAQUI A. F. Ambiente de programação baseado na web para robótica educacional de baixo custo. *HOLOS*, Ano 30, Vol. 5, 2014.

WING, J. Computational thinking. *Communications of ACM*, v. 49, n. 3, p. 33-36, 2006.

Robótica educacional uma abordagem prática no ensino de lógica de programação tem como proposta um ensino contextualizado e articulado na Educação Profissional e Tecnológica. O material traz uma sequência didática, para o ensino de algoritmos com uso de componentes eletrônicos, com nove projetos para serem trabalhados em sala de aula, podendo ser utilizado uma ferramenta de prototipagem eletrônica de baixo custo ou também um simulador para a prática de tais projetos.

