2024

Guia do TinkerCad

AUTODESK®
TINKERCAD®

Antonio Fernando Traina

Fatec Bebedouro

2024

1. Introdução ao Tinkercad

O Tinkercad é uma plataforma online gratuita criada pela Autodesk, projetada para facilitar o aprendizado e a prática de design 3D, simulação de circuitos eletrônicos e programação de micro controladores. Popular entre educadores, estudantes e entusiastas de tecnologia, o Tinkercad oferece uma interface intuitiva e acessível que permite aos usuários criar e simular projetos diretamente no navegador, sem necessidade de instalação de software.

Quadro 1 Principais Funcionalidades do Tinkercad

Funcionalidade	Descrição
Design 3D	Ferramentas para modelagem tridimensional simples, ideal para iniciantes.
	Permite a criação de modelos para impressão 3D, design de produto e prototipagem rápida.
Simulação de Circuitos	Criação e simulação de circuitos eletrônicos utilizando uma ampla gama de componentes como resistores, **LEDs**, capacitores, transistores, micro controladores (Arduino, NodeMCU), sensores, motores e mais.
	Ferramentas para analisar o funcionamento do circuito em tempo real e identificar erros ou falhas de projeto.
Programação de Microcontroladores	Suporte para programação de micro controladores, especialmente Arduino, com um editor de código integrado.
	Opções para programação visual usando blocos de código (similar ao *Scratch*) ou escrita de código diretamente em C++ (Arduino IDE).
Codeblocks (Blocos de Código)	Um ambiente de programação visual baseado em blocos que permite a criação de scripts para controlar objetos 3D e simular movimentos ou interações.

O principal uso do Tinkercad é na educação e na prototipagem. Em ambientes educacionais, ele serve como um laboratório virtual onde estudantes podem aprender os princípios da eletrônica e da programação de micro controladores, como o Arduino, sem a necessidade de equipamentos físicos caros. Por exemplo, uma turma de engenharia pode usar o Tinkercad para montar circuitos simples, como o acionamento de **LEDs**, até sistemas mais complexos, como sensores de temperatura integrados a micro controladores. Essa simulação virtual permite que os alunos experimentem e aprendam com erros em um ambiente seguro e controlado.

Além disso, o Tinkercad é amplamente utilizado para prototipagem rápida e desenvolvimento de produtos. Designers e engenheiros podem rapidamente criar modelos 3D para impressão ou testar conceitos de produtos sem precisar passar pelo processo completo de fabricação. Isso é particularmente útil para criar protótipos funcionais de dispositivos, ajustar o design antes da produção em larga escala, ou simplesmente visualizar ideias de maneira mais tangível.

No campo da Internet das Coisas (IoT), o Tinkercad se destaca por permitir a simulação de redes IoT, incluindo o uso de sensores, atuadores e dispositivos de comunicação como o **NodeMCU**. Usuários podem criar projetos que simulam sistemas automatizados, como uma estufa inteligente que ajusta a irrigação com base em dados de umidade do solo, ou sistemas de monitoramento remoto que usam sensores de temperatura e umidade conectados a uma rede Wi-Fi.

Além dos usos educacionais e de prototipagem, o Tinkercad é uma ferramenta valiosa para entusiastas e *makers* que desejam explorar eletrônica e design 3D por conta própria. A plataforma permite que esses usuários criem desde circuitos simples, como dispositivos de alarme caseiros, até projetos mais sofisticados, como robôs controlados por Arduino. A facilidade de uso do Tinkercad e sua vasta biblioteca de componentes eletrônicos incentivam o aprendizado autodidata, permitindo que

qualquer pessoa, independentemente de sua experiência anterior, possa experimentar e desenvolver suas habilidades.

O Tinkercad também é conhecido por sua comunidade ativa e seus recursos educacionais. Oferece uma ampla gama de tutoriais e lições interativas que guiam os usuários passo a passo através de conceitos fundamentais e projetos práticos. Isso o torna uma ferramenta de aprendizado versátil, capaz de suportar uma ampla gama de disciplinas e níveis de habilidade, desde iniciantes até usuários mais avançados.

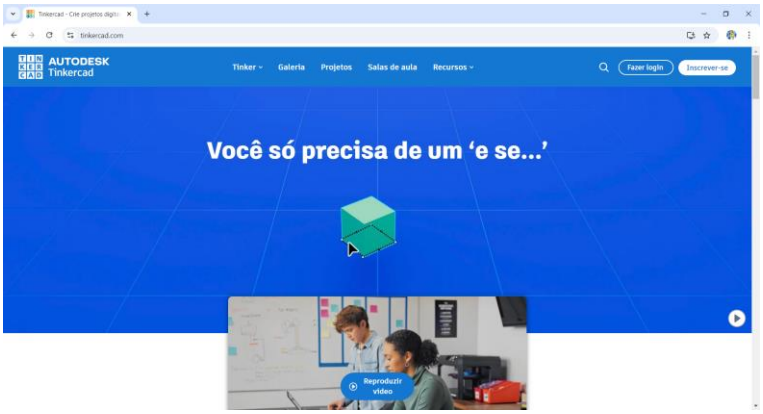
Em resumo, o Tinkercad é uma plataforma poderosa e versátil que facilita o aprendizado e a prática em várias disciplinas técnicas. Sua acessibilidade, interface amigável e suporte a uma ampla variedade de projetos o tornam uma ferramenta ideal para estudantes, educadores, profissionais e entusiastas interessados em eletrônica, design 3D e programação de micro controladores.

Quadro 2 Exemplos de Uso do Tinkercad

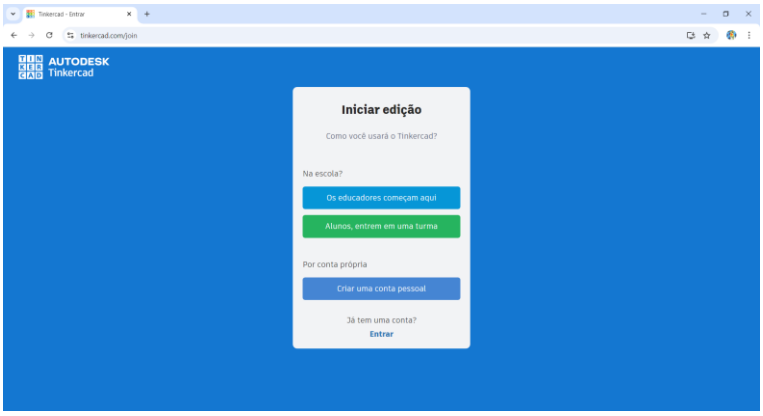
Categoria de Uso	Exemplo de Uso	Descrição
Educação em Engenharia e Eletrônica	**Laboratórios Virtuais**	Professores e estudantes utilizam o Tinkercad para realizar laboratórios virtuais de eletrônica, simulando circuitos que normalmente exigiriam equipamentos físicos caros ou de difícil acesso.
	Introdução ao Arduino	O Tinkercad é amplamente utilizado para ensinar conceitos básicos de programação e eletrônica utilizando o Arduino, permitindo que os alunos pratiquem o que aprenderam sem a necessidade de hardware físico.
Prototipagem e Desenvolvimento de Produtos	**Prototipagem Rápida**	Designers e engenheiros usam o Tinkercad para criar protótipos de novos produtos rapidamente, visualizando e ajustando o design em um ambiente digital antes de construir versões físicas.
	Design para Impressão 3D	Criadores de todo o mundo utilizam o Tinkercad para projetar modelos que podem ser impressos em 3D, desde peças funcionais até arte e objetos decorativos.
Desenvolvimento de Projetos IoT (Internet das Coisas)	**Simulação de Redes IoT**	O Tinkercad permite a construção de projetos de IoT simulando sensores, atuadores e dispositivos de comunicação que podem ser integrados a redes Wi-Fi usando plataformas como NodeMCU.
	Desenvolvimento de Sistemas Automatizados	Criadores utilizam o Tinkercad para simular sistemas automatizados, como controle de iluminação, irrigação inteligente e monitoramento ambiental, integrando hardware e software em um único ambiente de simulação.
Projetos Maker e Hobby	**Circuitos e Dispositivos DIY (*Do It Yourself*)**	Entusiastas de eletrônica e *makers* utilizam o Tinkercad para planejar e simular circuitos para projetos DIY, como robôs, sistemas de alarme e *gadgets* personalizados.
	Aprendizado Autodidata	O Tinkercad oferece uma plataforma acessível para aprendizes autodidatas explorarem eletrônica e design 3D em um ambiente de baixo risco e alta flexibilidade.

2. Como Criar uma Conta no Tinkercad

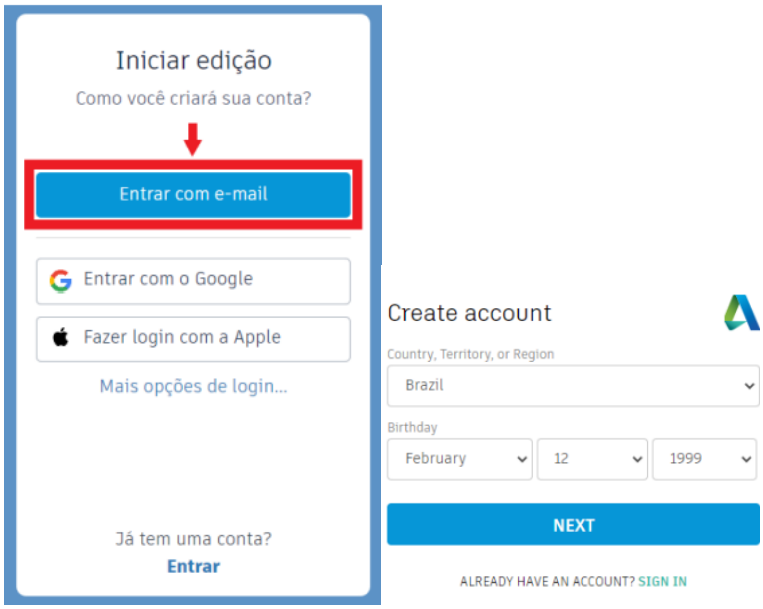
1. **Acesse o Site**: Vá para o site do Tinkercad (tinkercad.com).



2. **Clique em "Fazer *Login*"**: No canto superior direito da tela, clique em "Fazer *Login*" ou "Inscrever-se" para criar uma conta.
3. **Escolha uma Opção de Cadastro**: você pode se inscrever usando um e-mail ou uma conta de rede social (Google, Apple, etc.).



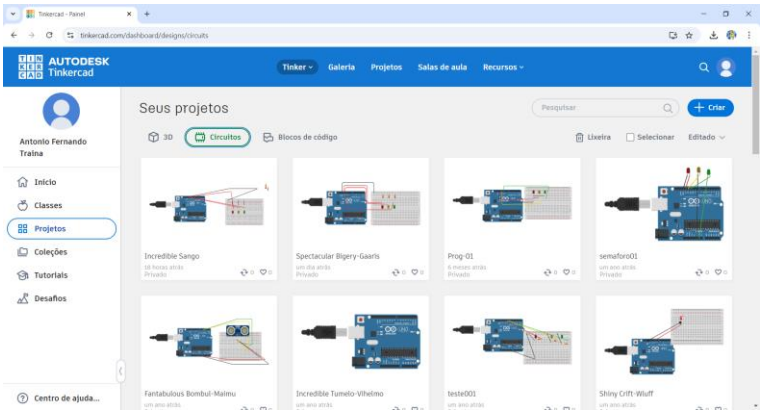
4. **Preencha Seus Dados**: Insira seu nome, e-mail e crie uma senha.



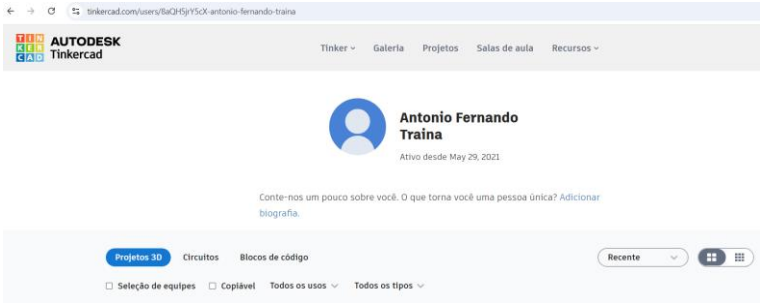
5. **Confirme Seu E-mail**: Você receberá um e-mail de confirmação; clique no link para ativar sua conta.
6. **Complete Seu Perfil**: Após a ativação, você pode completar seu perfil adicionando informações adicionais, se desejar.

3. Navegando pela Interface do Tinkercad

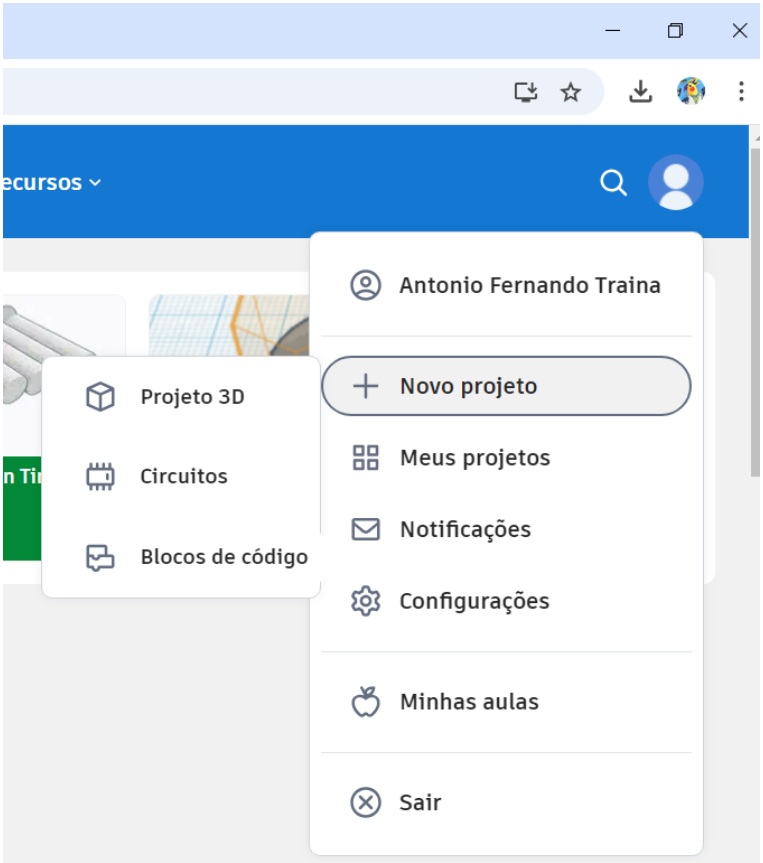
- **Página Inicial (*Dashboard*)**: Ao fazer *login*, você verá o painel principal, onde estão listados seus projetos (chamados de "Designs").



- **Menu Principal**: No topo da página, você encontrará o menu com opções para "3D Designs", "Circuitos", "Blocos de código", e "Lições".

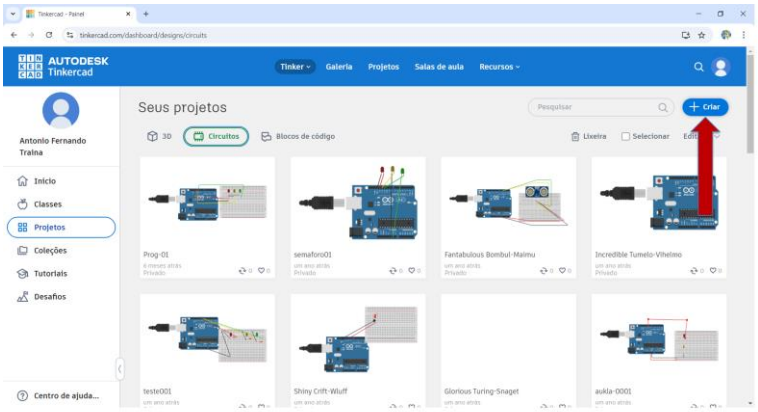


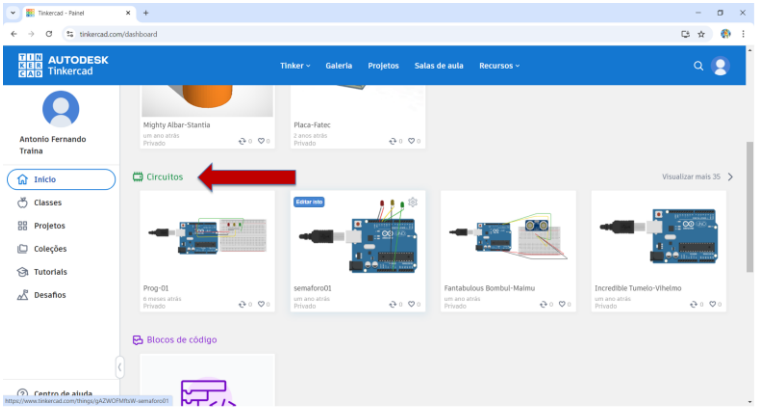
- **Criação de Novos Projetos**: Use o botão "Novo Projeto" para iniciar um novo projeto em 3D, um novo circuito ou um novo bloco de código.



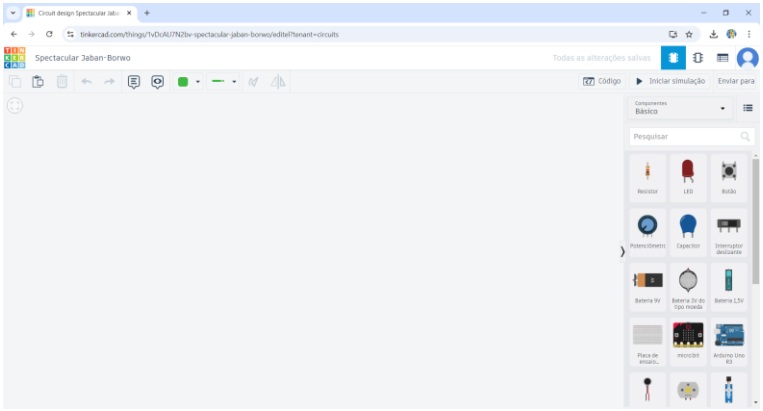
4. Criando e Simulando Circuitos no Tinkercad

1. **Iniciando um Novo Projeto de Circuito**: No painel principal, clique em "Circuitos" e depois em "Criar".

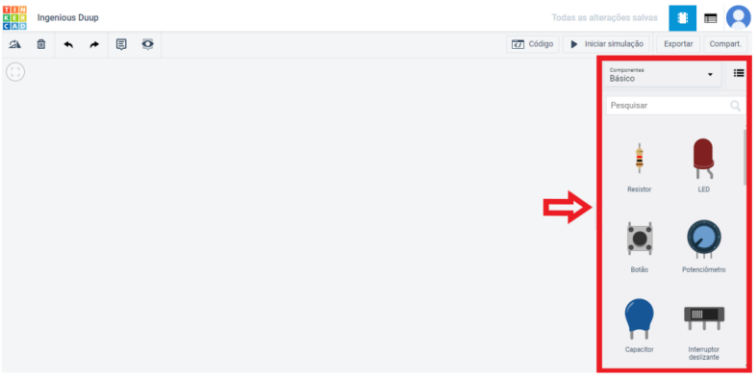




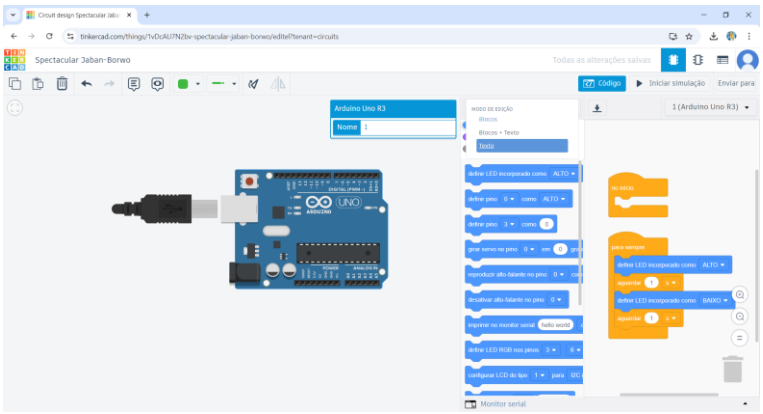
2. **Explorando o Ambiente de Circuitos**:
    - **Área de Trabalho**: O espaço em branco onde você monta seus componentes.



    - **Biblioteca de Componentes**: Localizada à direita, onde você encontra todos os componentes eletrônicos disponíveis, como resistores, **LEDs**, capacitores, Arduino, NodeMCU, e muito mais.

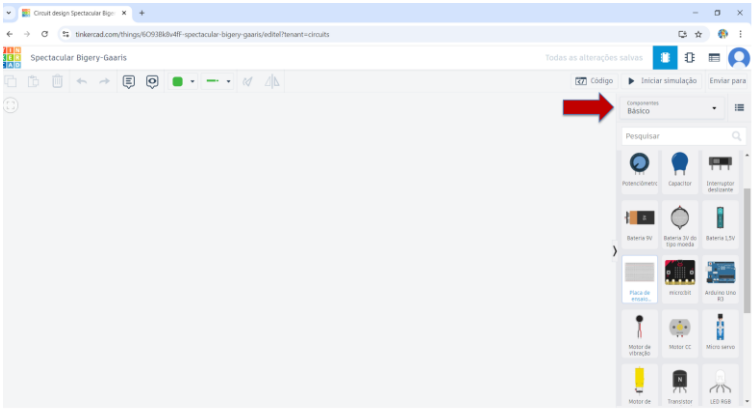


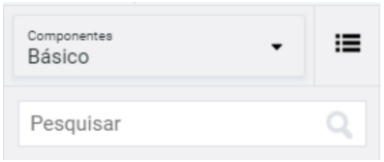
- **Painel de Código**: Parte inferior da tela, usada para programar micro controladores, como o Arduino. Você pode escrever código em C++ (Arduino IDE) ou usar blocos de código visual (similar ao *Scratch*).



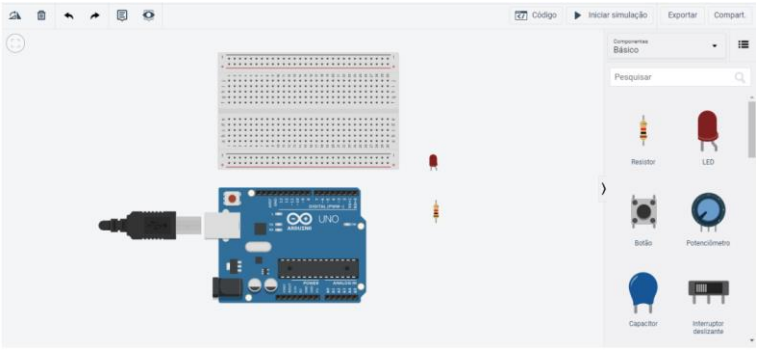
3. Montando um Circuito Básico:

- **Adicionar Componentes**: Clique e arraste componentes da biblioteca para a área de trabalho.

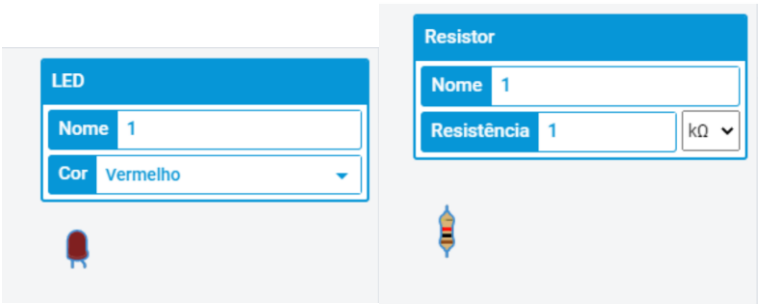


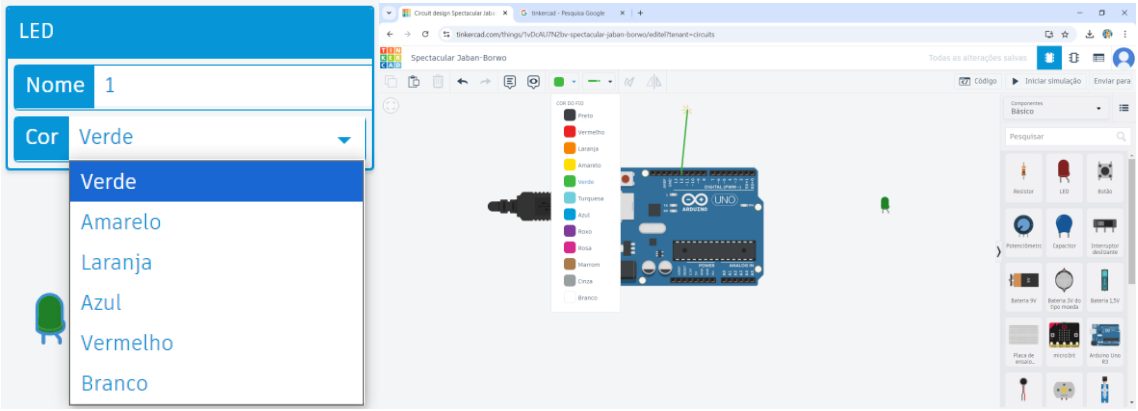


- **Conectar Componentes**: Use o cursor para conectar os terminais dos componentes com fios virtuais. Clique em um terminal, arraste para o próximo terminal e solte para criar a conexão.



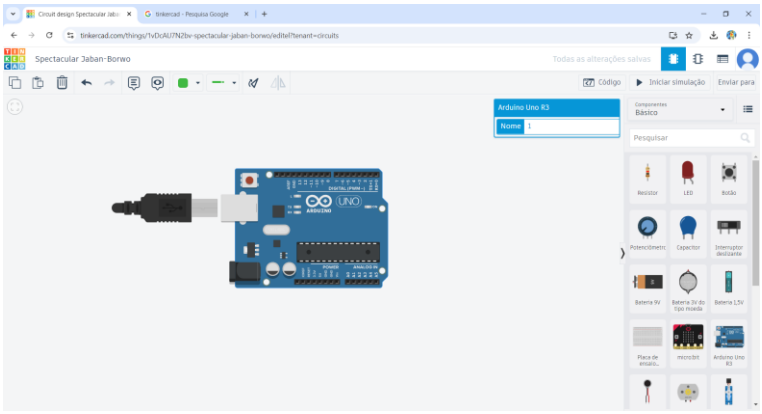
- **Configuração dos Componentes**: Clique em qualquer componente para configurar seus parâmetros (por exemplo, valor da resistência, cor do **LED**, etc.).



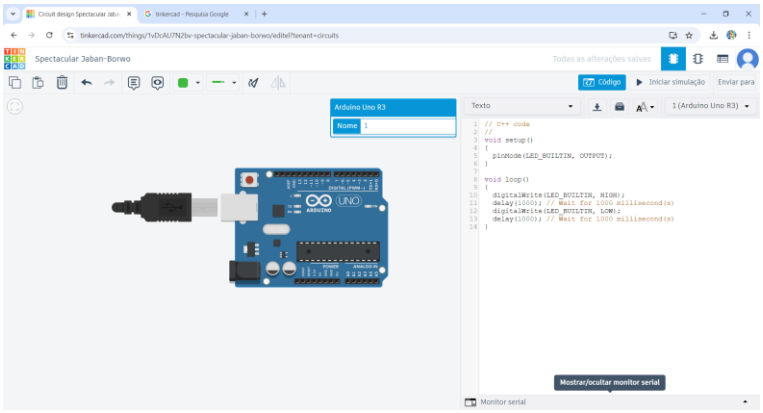


Programação de Micro controladores:

- **Selecionar Arduino Uno**: Arraste um Arduino Uno da biblioteca para a área de trabalho.



- **Programação com Blocos de Código**: Selecione a opção “Código” no painel inferior e escolha “Blocos” para programar usando blocos visuais.

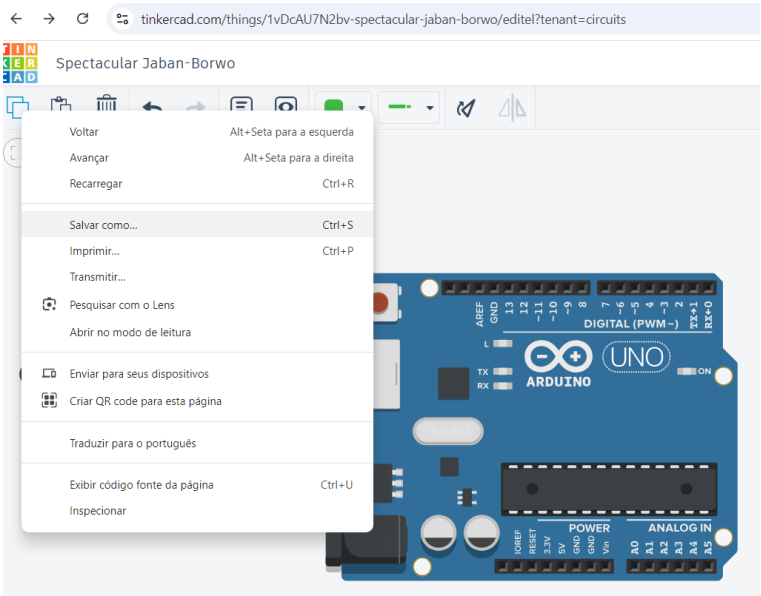


- **Programação em Texto**: Altere a aba para "Texto" para escrever código C++ diretamente.

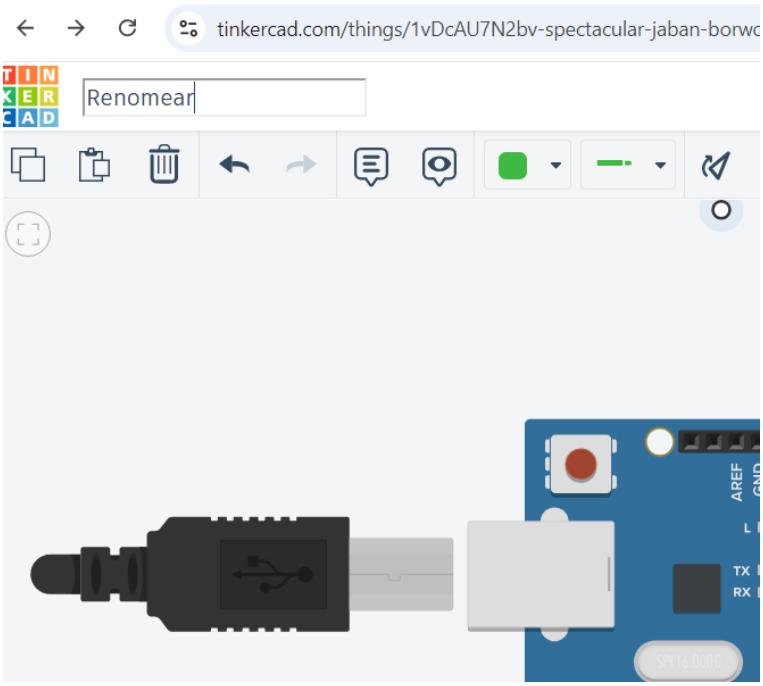
Executando o Código: Clique em "Iniciar simulação" no canto superior direito para iniciar a simulação e ver o circuito em ação.

5. Salvando e Compartilhando Projetos

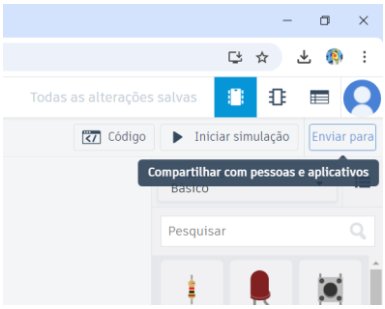
1. **Salvar Projeto**: O Tinkercad salva automaticamente seus projetos, mas é uma boa prática clicar em "Salvar" periodicamente.



2. **Renomear Projeto**: Clique no nome do projeto no canto superior esquerdo para renomeá-lo.

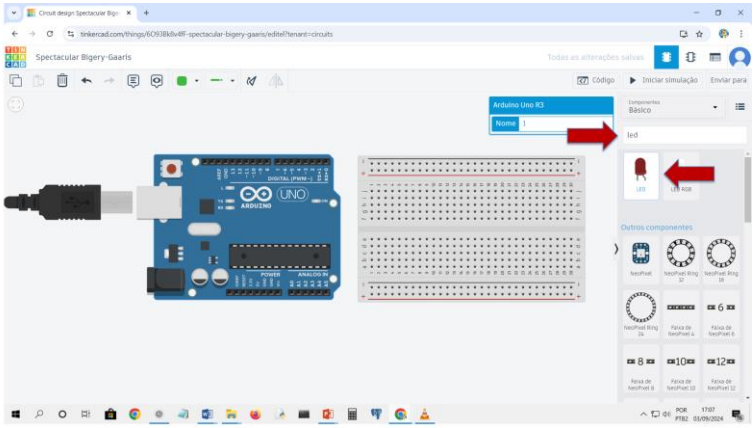


3. **Compartilhar Projeto**: Use a opção “Share” para gerar um link compartilhável ou para baixar o projeto como um arquivo .BRD (para PCBs) ou código .INO (para Arduino).

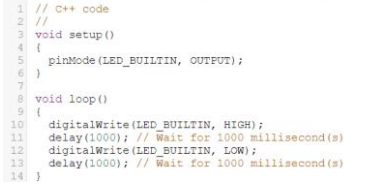


6. Demonstração Prática no Tinkercad (Blink)

Adicione à área de projeto um Arduino, uma placa de ensaio e um **LED** vermelho



Automaticamente o código do *blink* irá surgir.



```
// C++ code
//
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Vamos explicar esse código a seguir:

```
//Código Arduino Documentado e Comentado
// Função setup() é executada uma vez quando o Arduino é ligado ou //resetado

void setup() {
  // Configura o pino digital 1 (ou porta número 13) como saída (OUTPUT)
  pinMode(13, OUTPUT);
}

// Função loop() é executada repetidamente após a função setup()
// O código dentro de loop() será executado continuamente em um ciclo //infinito

void loop() {

// Envia um sinal de alta tensão (HIGH) para o pino 13
  // Isso faz com que o LED conectado ao pino 13 acenda
  digitalWrite(13, HIGH);

  // Pausa a execução do código por 1000 milissegundos (1 segundo)
  delay(1000);

  // Envia um sinal de baixa tensão (LOW) para o pino 13
  // Isso faz com que o LED conectado ao pino 13 apague

digitalWrite(13, LOW);

  // Pausa a execução do código por 1000 milissegundos (1 segundo)
  delay(1000);
}
```

Explicação dos Comandos:

Função setup():

Esta função é executada uma única vez quando o Arduino é inicializado ou quando é pressionado o botão de reset.

Dentro de setup(), usamos o comando pinMode() para definir o pino 13 do Arduino como uma saída (OUTPUT). Isso é necessário porque queremos controlar um LED (ou outro componente) que está conectado a este pino.

Função loop():

A função loop() contém o código que será executado repetidamente em um ciclo infinito. No contexto de Arduino, essa função é onde a maior parte do código de controle é colocada.

Dentro de loop(), utilizamos digitalWrite(13, HIGH) para enviar um sinal de alta tensão (5V) ao pino 13, o que faz com que o LED conectado a este pino acenda.

Em seguida, delay(1000) faz uma pausa na execução do código por 1000 milissegundos (ou 1 segundo). Durante este tempo, o LED permanece aceso.

Após a pausa, digitalWrite(13, LOW) envia um sinal de baixa tensão (0V) ao pino 13, apagando o LED.

Outra pausa de 1 segundo é introduzida com delay(1000) antes do loop se repetir.

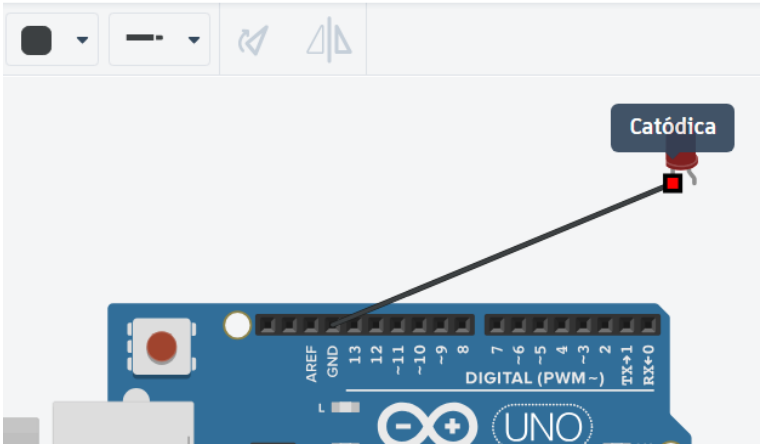
Resumo do Comportamento:

O código faz com que um LED conectado ao pino 13 do Arduino pisque continuamente. O LED permanece aceso por 1 segundo, depois apaga por 1 segundo, e esse ciclo se repete indefinidamente. Este é um exemplo clássico

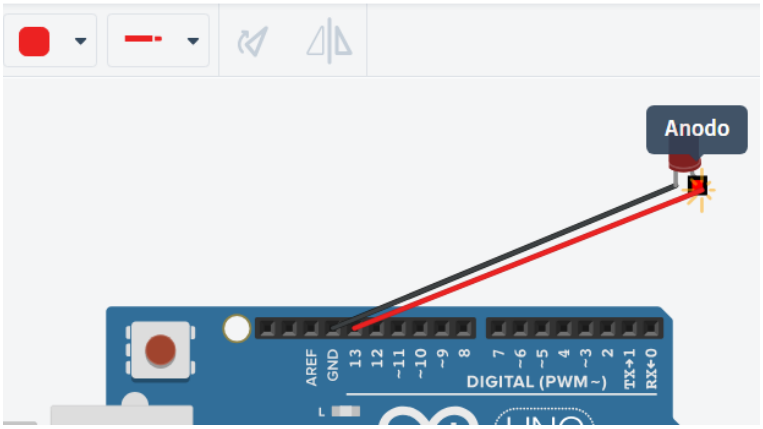
conhecido como "Blink", que é frequentemente usado como o primeiro programa para iniciantes aprenderem a controlar componentes básicos com um Arduino.

Agora monte o seguinte circuito:

a) Conecte um fio preto na base catódica do LED:



b) Conecte um fio vermelho na base anódica do LED:



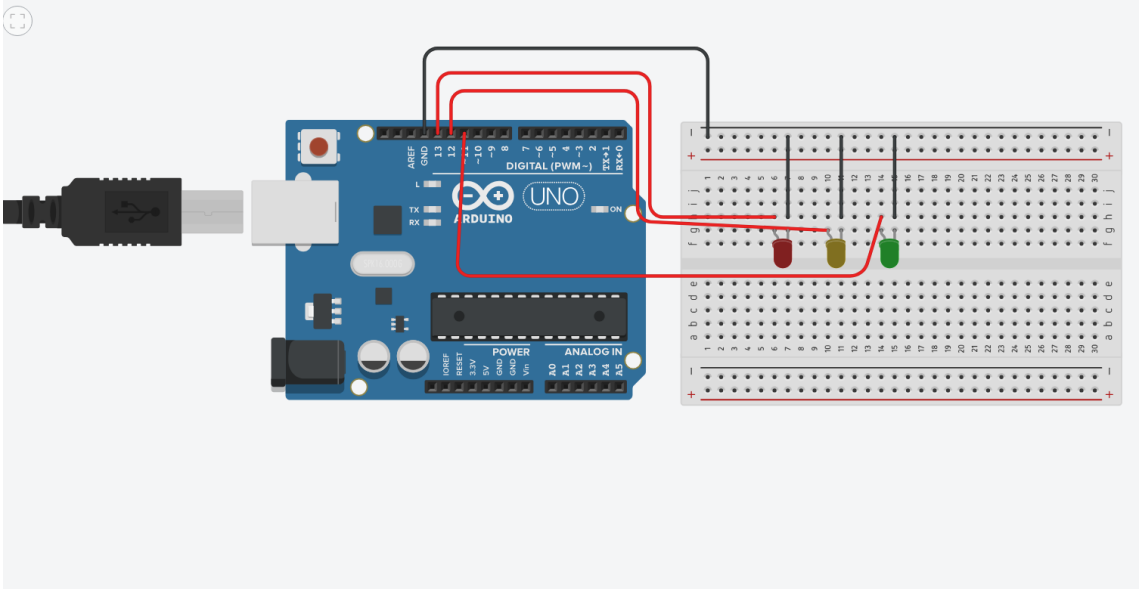
c) E faça a simulação:

Note que o LED irá acender. Mas uma informação (i) irá surgir. Isso significa que o LED está com uma tensão superior a sua carga limite. E que o LED poderá queimar. Nãose preocupe, trata-se de uma simulação. Mas para prever que isso não aconteça

no mundo real um resistor deverá ser conectado para abaixar a tensão que passa pelo LED. Veremos isso mais adiante.

7. Demonstração Prática no Tinkercad (Semáforo)

1. Começe montado o seguinte hardware:



2. Usando com crlt+C e o crlt V escreva o seguinte código Arduino:

```
// C++ code
//
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
}

void loop()
{
  //led vermelho
  digitalWrite(13, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(13, LOW);
  delay(1000); // Wait for 1000 millisecond(s)

  //led amarelo
  digitalWrite(12, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(12, LOW);
  delay(1000); // Wait for 1000 millisecond(s)

  //led verde
   digitalWrite(11, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(11, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Vamos explicar esse código a seguir:

```
// Código C++
// Código simples para simular um semáforo usando três LEDs conectados aos pinos
digitais 13, 12 e 11
void setup()
{
// Configura os pinos 13, 12 e 11 como saídas para controlar os LEDs
  pinMode(13, OUTPUT);  // Configura o pino 13 como saída para o LED vermelho
  pinMode(12, OUTPUT);  // Configura o pino 12 como saída para o LED amarelo
  pinMode(11, OUTPUT);  // Configura o pino 11 como saída para o LED verde
}

void loop()
```

```
{
  // Simulação da luz vermelha do semáforo
  digitalWrite(13, HIGH);  // Acende o LED (luz vermelha)
  delay(1000);                      // Espera por 1000 milissegundos (1 segundo)
  digitalWrite(13, LOW);   // Apaga o LED (luz vermelha apagada)
  delay(1000);                      // Espera por 1000 milissegundos (1 segundo)

  // Simulação da luz amarela do semáforo
  digitalWrite(12, HIGH);  // Acende o LED (luz amarela)
  delay(1000);                      // Espera por 1000 milissegundos (1 segundo)
  digitalWrite(12, LOW);   // Apaga o LED (luz amarela apagada)
  delay(1000);                      // Espera por 1000 milissegundos (1 segundo)

  // Simulação da luz verde do semáforo
  digitalWrite(11, HIGH);  // Acende o LED (luz verde)
  delay(1000);                      // Espera por 1000 milissegundos (1 segundo)
  digitalWrite(11, LOW);   // Apaga o LED (luz verde apagada)
  delay(1000);                      // Espera por 1000 milissegundos (1 segundo)
}
```

Explicação dos Comandos:

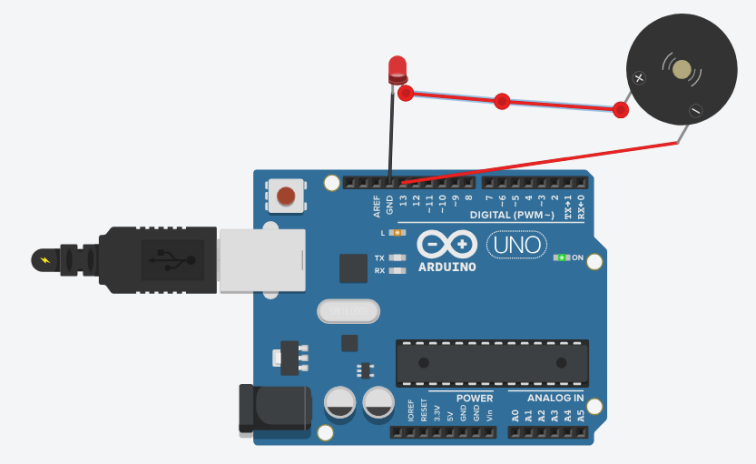
1. **Função setup():**
   - A função setup() é executada uma vez quando o Arduino é inicializado ou quando o botão de reset é pressionado.
   - Dentro da função setup(), utilizamos o comando pinMode() para configurar os pinos 13, 12 e 11 como saídas. Esses pinos serão usados para controlar os LEDs vermelho, amarelo e verde, respectivamente.
2. **Função loop():**
   - A função loop() contém o código que será executado repetidamente em um ciclo infinito. É aqui que o comportamento do semáforo é simulado.
   - **Simulação da Luz Vermelha:**
     - O comando digitalWrite(13, HIGH) envia um sinal de alta tensão (5V) ao pino 13, acendendo o LED vermelho.
     - O comando delay(1000) introduz uma pausa de 1 segundo, mantendo o LED vermelho aceso.
     - digitalWrite(13, LOW) apaga o LED vermelho, e outra pausa de 1 segundo (delay(1000)) é introduzida antes de passar para a luz amarela.
   - **Simulação da Luz Amarela:**
     - De maneira similar, digitalWrite(12, HIGH) acende o LED amarelo no pino 12.
     - O LED amarelo é mantido aceso por 1 segundo e depois apagado com digitalWrite(12, LOW).
     - Outra pausa de 1 segundo é introduzida antes de passar para a luz verde.
   - **Simulação da Luz Verde:**
     - digitalWrite(11, HIGH) acende o LED verde no pino 11.
     - O LED verde é mantido aceso por 1 segundo e depois apagado com digitalWrite(11, LOW).

- Após uma pausa de 1 segundo, o ciclo do semáforo se reinicia, voltando para a luz vermelha.

Resumo do Comportamento:

Este código simula o funcionamento de um semáforo usando três LEDs: vermelho, amarelo e verde. Cada LED é conectado a um pino digital diferente do Arduino (pinos 13, 12 e 11, respectivamente). O semáforo acende e apaga cada LED por 1 segundo, simulando as transições de luz em um semáforo real. O ciclo é repetido indefinidamente, proporcionando uma simulação contínua do semáforo.

8. Desafio: Usando o circuito anterior, acrescente um Buzzer (Piezo) monte o seguinte circuito:



Perguntas:

1) Deve haver alguma alteração no código?
2) O que o dispositivo Buzzer faz?

9. Exercícios extras para quem quer ir além:

1. **Aumentar e Diminuir o Brilho de um LED (Fade)**

   Descrição: Use um LED para criar um efeito de "respiração", aumentando e diminuindo seu brilho gradualmente.

   Objetivo: Aprender a usar PWM (Modulação por Largura de Pulso) com o comando analogWrite().

   Dica: Conecte o LED a um pino PWM (como o pino 9) e use um resistor de 220Ω.

2. **Sequência de Pisca com Três LEDs**

   Descrição: Crie uma sequência onde três LEDs piscam um após o outro.

   Objetivo: Aprender a controlar a sequência de eventos usando delay().

   Dica: Conecte os LEDs aos pinos 11, 12 e 13.

3. **LED Reagindo a um Botão**

   Descrição: Faça um LED acender apenas quando um botão for pressionado.

Objetivo: Aprender a ler entradas digitais usando digitalRead() e usar condicionais (if).

Dica: Use um resistor pull-down com o botão conectado ao pino 2 e o LED ao pino 13.

4. **Piscar um LED com Intervalo Controlado por Potenciômetro**

   Descrição: Controle a velocidade de pisca de um LED usando um potenciômetro.

   Objetivo: Aprender a usar entradas analógicas com analogRead() para controlar saídas digitais.

   Dica: Conecte o potenciômetro ao pino A0 e o LED ao pino 13.

5. **Correr de LEDs (Chase Effect)**

   Descrição: Crie um efeito de "corrida" com uma série de LEDs que acendem em sequência e apagam após um intervalo curto.

   Objetivo: Aprender a criar loops de controle e manipular múltiplos LEDs.

   Dica: Use pelo menos quatro LEDs conectados aos pinos 8, 9, 10, e 11.

6. **LED Acende de Acordo com a Intensidade de Luz**

   Descrição: Use um sensor de luz (LDR) para controlar o brilho de um LED: quanto mais escuro, mais brilhante o LED.

   Objetivo: Aprender a usar sensores analógicos e mapear seus valores para controlar saídas.

   Dica: Conecte o LDR ao pino A0 e o LED a um pino PWM como o 9.

7. **Piscar LEDs em Código Morse**

   Descrição: Faça três LEDs piscarem em código Morse para representar uma palavra ou letra.

   Objetivo: Aprender a manipular tempos e criar padrões complexos com base em código.

   Dica: Escolha três letras simples e use os pinos 11, 12 e 13 para conectar os LEDs.