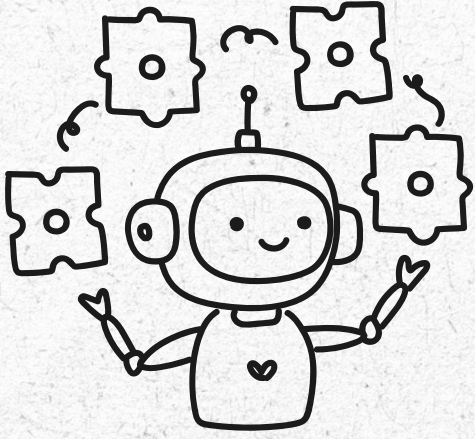
TINKERCAD

10 PROJETOS BÁSICOS

WADSON BENFICA

TINKERCAD

10 PROJETOS BÁSICOS



PREFÁCIO

O mundo da tecnologia está em constante evolução, e, a cada dia, novas ferramentas e conhecimentos tornam-se mais acessíveis, permitindo que pessoas de todas as idades e formações explorem, criem e inovem. Este livro, "10 Projetos Básicos com Arduino no Tinkercad", é uma ponte para esse universo, trazendo conceitos fundamentais de eletrônica e programação de forma prática, didática e envolvente.

POR QUE ESTE LIVRO É ESPECIAL?

O diferencial desta obra está na sua abordagem prática e amigável. Cada projeto foi cuidadosamente pensado para ensinar um conceito importante de forma gradual e clara. Desde o clássico "Piscar um LED" até sistemas automatizados de irrigação, cada capítulo é um convite para aprender enquanto se cria algo tangível. Utilizando o Tinkercad, uma ferramenta de simulação gratuita, o leitor pode experimentar e desenvolver seus projetos sem a necessidade de equipamentos físicos, eliminando barreiras para o aprendizado.

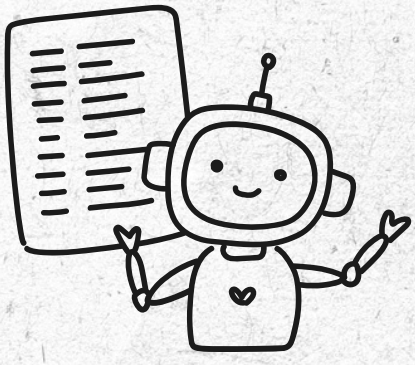
A QUEM ESTE LIVRO SE DESTINA?

Se você é um iniciante curioso, um estudante em busca de reforço em conceitos práticos, ou até mesmo um entusiasta querendo expandir suas habilidades, este livro é para você. Com explicações detalhadas, diagramas claros e dicas preciosas, o Professor Wadson Benfica conduz o leitor com segurança por cada etapa, transformando dificuldades em oportunidades para aprender e crescer.

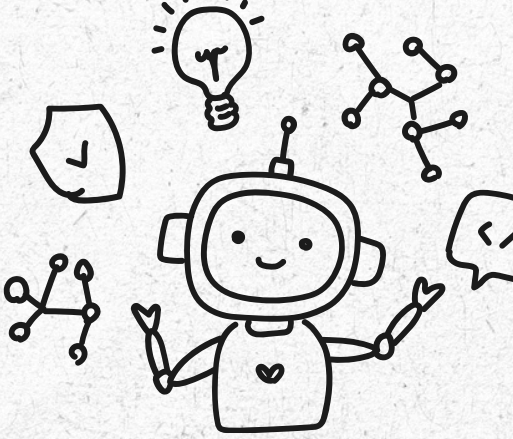
O IMPACTO DESTE APRENDIZADO

Mais do que apenas projetos, este livro é uma porta de entrada para um futuro onde tecnologia e criatividade caminham lado a lado. A experiência e dedicação do autor garantem que o leitor não apenas aprenda a usar o Arduino, mas também compreenda os princípios que tornam possíveis tantas inovações no mundo moderno.

TABELA DE CONTEÚDOS



INTRODUÇÃO



O Arduino revolucionou o mundo da eletrônica ao trazer simplicidade e acessibilidade para projetos de prototipagem. Se você sempre sonhou em criar algo com suas próprias mãos — como um robô, um sistema automatizado, ou apenas acender um LED de forma criativa — este livro é para você!

O QUE VOCÊ VAI APRENDER NESTE LIVRO?

Neste livro, exploraremos **10 projetos práticos e básicos usando o Arduino**, desde o clássico "piscar um LED" até a automação de um sistema de irrigação. Usaremos o **Tinkercad**, uma ferramenta online gratuita e intuitiva para prototipagem virtual. Não precisa de peças físicas — só um computador, internet e sua criatividade!

ESTRUTURA DO LIVRO

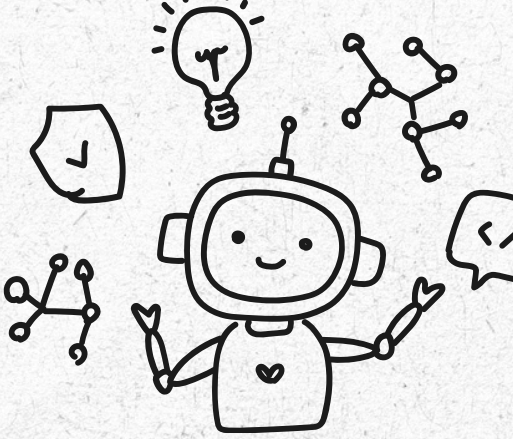
Cada capítulo aborda um projeto específico. Você verá:

1. Descrição do Projeto: O que você aprenderá.
2. Componentes Virtuais: Tudo que será usado no Tinkercad.
3. Esquema de Circuito: Passo a passo para montar no simulador.
4. Código em Arduino: Com explicações linha por linha.
5. Dicas e Melhorias: Como aprimorar o projeto para algo mais desafiador.

POR QUE USAR O TINKERCAD?

O Tinkercad é ideal para iniciantes, pois permite simular circuitos e testar códigos sem o risco de erros custosos. Além disso:

- Você pode experimentar e aprender conceitos de eletrônica com facilidade.
- É uma ferramenta gratuita e acessível.
- Ajuda a reduzir o tempo e o custo de prototipagem.



O QUE É O ARDUINO?

O Arduino é uma plataforma de código aberto que combina hardware e software. Seu microcontrolador permite controlar sensores, LEDs, motores, e muito mais. Ele é amplamente utilizado em projetos que envolvem:

- Automação doméstica: Controle de luzes e aparelhos.
- Robótica: Sensores de movimento e motores.
- Educação: Experimentos científicos e ensino de tecnologia.

Com o Arduino, suas ideias podem se transformar em realidade.

ANTES DE COMEÇAR

Certifique-se de criar uma conta no **Tinkercad**. É simples:

1. Acesse https://www.tinkercad.com.
2. Registre-se com seu e-mail.
3. Explore a seção de "Circuitos".

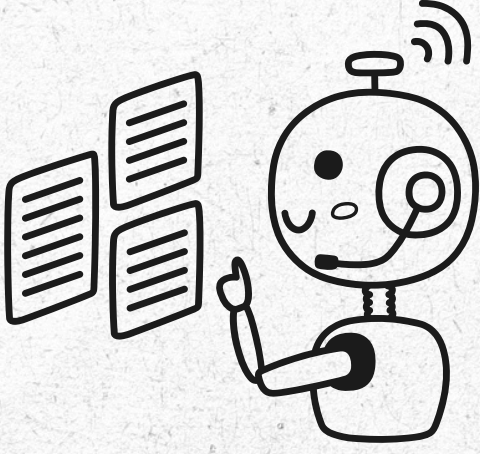
CAlém disso, baixe o software Arduino IDE (se quiser experimentar os códigos fisicamente no futuro) no site oficial: https://www.arduino.cc.

PRONTO PARA COMEÇAR?

Nos próximos capítulos, você será guiado passo a passo pelos projetos. Seja você iniciante ou curioso por eletrônica, este livro vai abrir as portas para um novo mundo de criação.

Agora, vamos para o primeiro projeto: Piscar um LED!

PROJETO 01



PISCAR UM LED (O CLÁSSICO BLINK)

Nosso primeiro projeto é o clássico "piscar um LED". Este projeto é uma ótima introdução ao mundo do Arduino, pois ensina conceitos básicos de controle de hardware e programação.

OBJETIVOS DO PROJETO

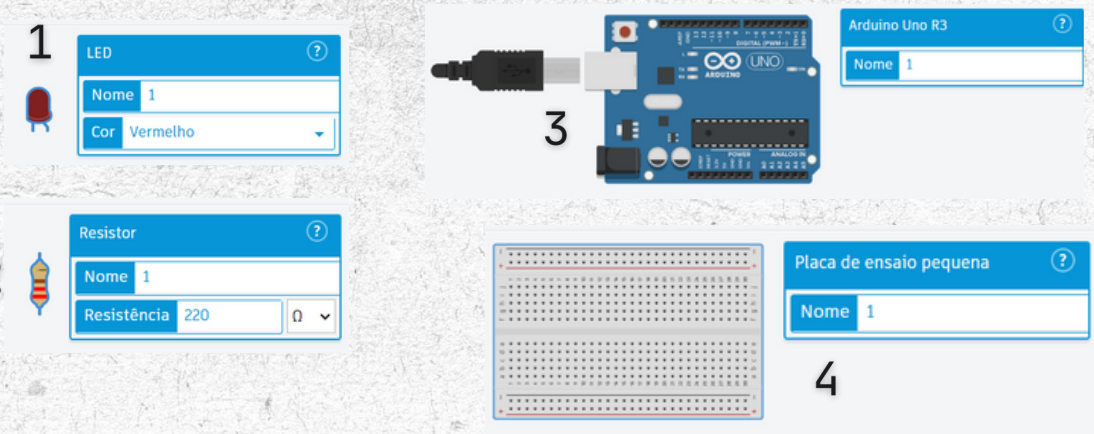
Controlar um LED para que ele pisque em intervalos regulares. Isso envolve a compreensão de:

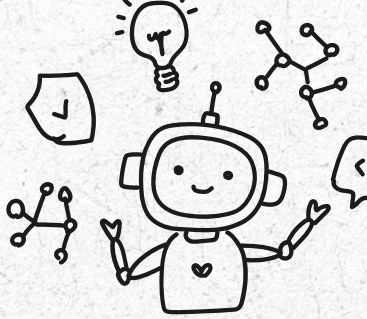
- Como conectar e controlar um LED.
- Uso do Tinkercad para montar o circuito e simular o código.
- Funções básicas do Arduino, como pinMode(), digitalWrite(), e delay().

COMPONENTES VIRTUAIS NECESSÁRIOS

1. 1 LED (vermelho, por exemplo).
2. 1 Resistor de 220 ohms.
3. Placa Arduino Uno.
4. Protoboard (placa de ensaio).
5. Fios de conexão.

Esses componentes estão disponíveis na biblioteca do Tinkercad.





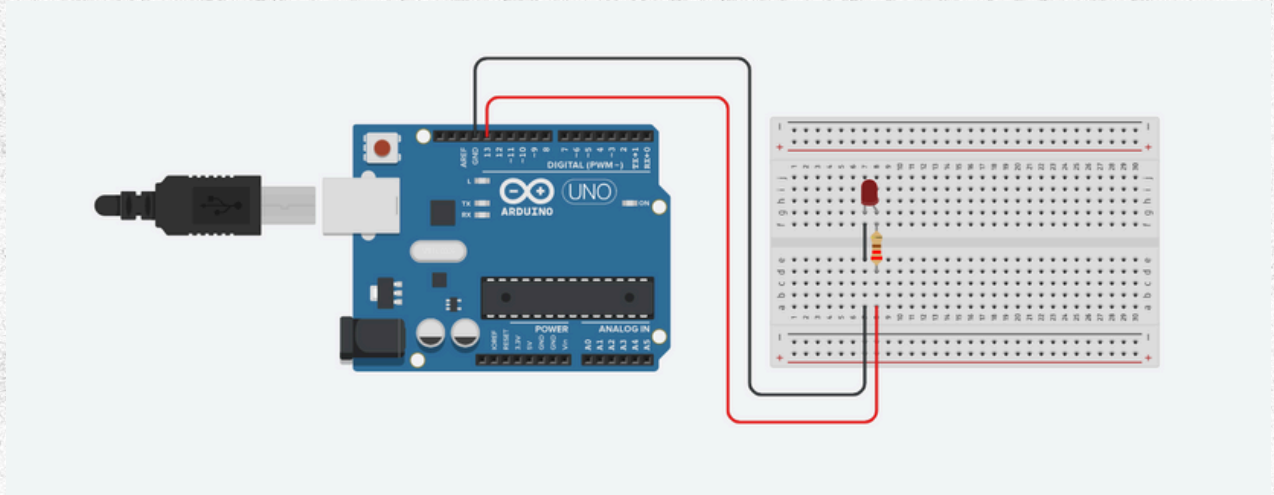
PASSO 1: MONTAGEM DO CIRCUITO

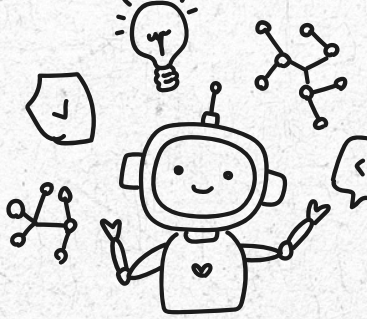
1. **Abra o Tinkercad:** Acesse a aba de circuitos e crie um novo projeto.
2. **Adicione os componentes:**
   - Arraste a placa Arduino Uno para a área de trabalho.
   - Insira o LED na protoboard (o terminal longo é o positivo/anodo e o curto é o negativo/catodo).
   - Conecte um resistor de 220 ohms ao terminal positivo do LED para limitar a corrente.
3. **Conexões no Arduino:**
   - Ligue o terminal positivo do LED (através do resistor) ao pino 13 do Arduino.
   - Conecte o terminal negativo do LED ao GND do Arduino.

Esquema do Circuito

No Tinkercad, o circuito deve se parecer com isso:

- Pino 13 → Resistor → LED positivo (anodo).
- GND → LED negativo (catodo).





PASSO 2: O CÓDIGO EM ARDUINO

Abaixo está o código para fazer o LED piscar:

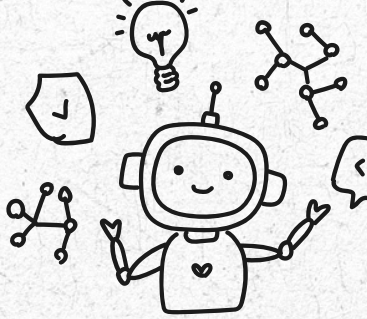
```
void setup() {
  pinMode(13, OUTPUT); // Configura o pino 13 como saída
}

void loop() {
  digitalWrite(13, HIGH); // Liga o LED
  delay(1000); // Espera 1 segundo
  digitalWrite(13, LOW); // Desliga o LED
  delay(1000); // Espera 1 segundo
}
```

Explicação do Código

1. setup(): Configuramos o pino 13 como saída com a função pinMode(). Este pino controlará o LED.
2. loop(): Este bloco executa repetidamente:
   - digitalWrite(13, HIGH): Liga o LED aplicando 5V no pino 13.
   - delay(1000): Aguarda 1000 milissegundos (1 segundo).
   - digitalWrite(13, LOW): Desliga o LED.
   - delay(1000): Aguarda mais 1 segundo antes de repetir.

PASSO 3: TESTANDO NO TINKERCAD

1. **Após montar o circuito e adicionar o código, clique no botão Iniciar Simulação.**
2. **Observe o LED piscando com intervalos de 1 segundo.**
3. **Experimente alterar os valores no delay() para mudar o tempo de piscar.**

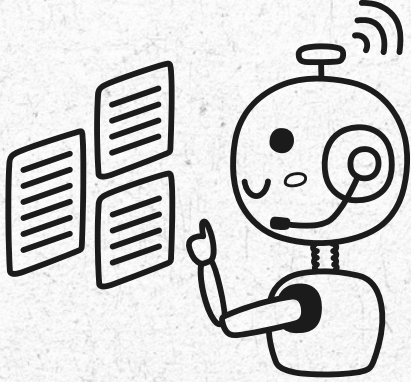


DICAS PARA MELHORAR O PROJETO

- Adicione mais LEDs: Tente adicionar LEDs em outros pinos, como 12 ou 11, e controle-os individualmente.
- Troque a cor do LED: Use LEDs de diferentes cores para criar um efeito de iluminação.
- Modifique os tempos: Altere o tempo de delay() para criar padrões diferentes de piscar.

CONCLUSÃO

Parabéns! Você criou e simulou seu primeiro projeto Arduino no Tinkercad. Este projeto apresentou conceitos básicos que serão usados em projetos futuros, como controle de saída digital e uso de temporizadores.
No próximo capítulo, vamos construir um semáforo simples, onde controlaremos múltiplos LEDs para simular um semáforo de trânsito.



PROJETO 02

SEMÁFORO SIMPLES

Neste capítulo, construiremos um semáforo simples usando LEDs. Este projeto irá expandir os conceitos aprendidos no capítulo anterior, adicionando o controle de múltiplas saídas e lógica sequencial.

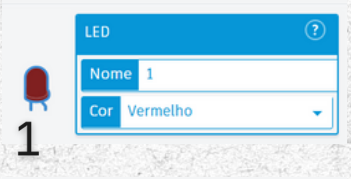
OBJETIVO DO PROJETO

Criar um semáforo com três LEDs (vermelho, amarelo e verde) que acendem em sequência, simulando o funcionamento de um semáforo real.

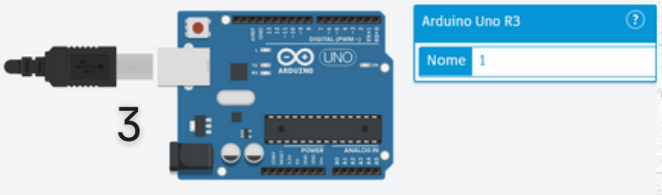
COMPONENTES VIRTUAIS NECESSÁRIOS

1. 3 LEDs (vermelho, amarelo, verde).
2. 3 Resistores de 220 ohms.
3. Placa Arduino Uno.
4. Protoboard (placa de ensaio).

Fios de conexão.

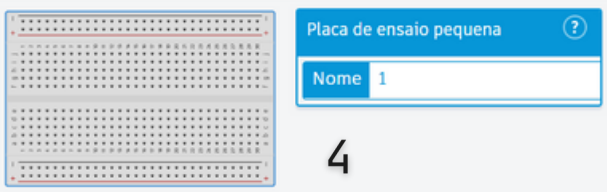


1

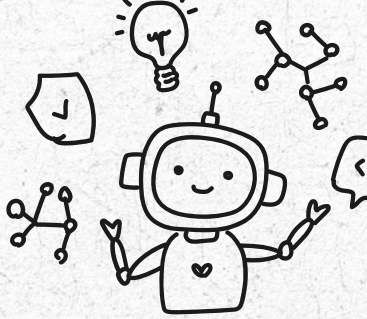


3

2



4



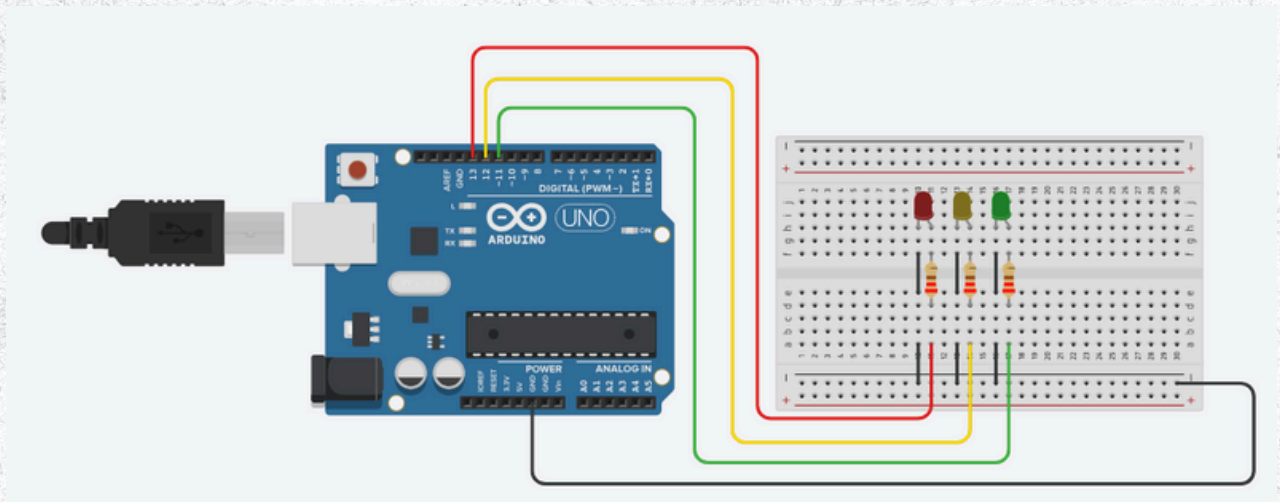
PASSO 1: MONTAGEM DO CIRCUITO

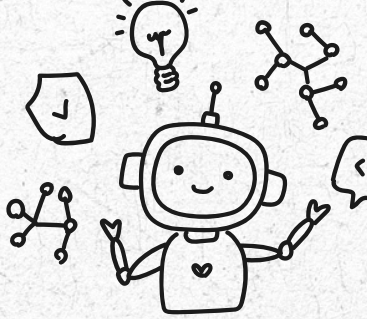
1. **Abra o Tinkercad e crie um novo projeto na aba de circuitos.**
2. **Adicione os componentes:**
   - Coloque três LEDs na protoboard, alinhados em sequência (vermelho, amarelo e verde).
   - Conecte um resistor de 220 ohms ao terminal positivo (anodo) de cada LED.
3. **conexões no Arduino:**
   - Conecte o terminal positivo do LED vermelho ao pino 13 do Arduino.
   - Conecte o terminal positivo do LED amarelo ao pino 12.
   - Conecte o terminal positivo do LED verde ao pino 11.
   - Ligue o terminal negativo (catodo) de todos os LEDs ao GND do Arduino.

Esquema do Circuito

No Tinkercad, o circuito deve se parecer com isso:

- Pino 13 → Resistor → LED vermelho (anodo).
- Pino 12 → Resistor → LED amarelo (anodo).
- Pino 11 → Resistor → LED verde (anodo).
- GND → Catodos dos três LEDs.

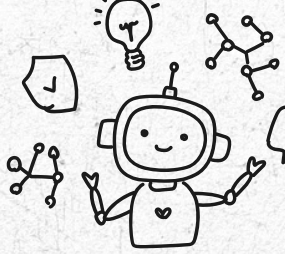




PASSO 2: O CÓDIGO EM ARDUINO

Abaixo está o código para controlar os LEDs em sequência, simulando um semáforo.

```
void setup() {
  pinMode(13, OUTPUT); // LED vermelho
  pinMode(12, OUTPUT); // LED amarelo
  pinMode(11, OUTPUT); // LED verde
}

void loop() {
  digitalWrite(13, HIGH); // Liga o LED vermelho
  delay(5000);            // Mantém por 5 segundos
  digitalWrite(13, LOW);  // Desliga o LED vermelho

  digitalWrite(12, HIGH); // Liga o LED amarelo
  delay(2000);            // Mantém por 2 segundos
  digitalWrite(12, LOW);  // Desliga o LED amarelo

  digitalWrite(11, HIGH); // Liga o LED verde
  delay(5000);            // Mantém por 5 segundos
  digitalWrite(11, LOW);  // Desliga o LED verde
}
```



Explicação do Código

1. **setup(): Configura os pinos 13, 12, e 11 como saídas para controlar os LEDs.**
2. **loop(): Cria a sequência do semáforo:**
   - Liga o LED vermelho por 5 segundos (delay(5000)).
   - Liga o LED amarelo por 2 segundos.
   - Liga o LED verde por 5 segundos.

Repete o ciclo.

PASSO 3: TESTANDO NO TINKERCAD

1. **Após montar o circuito e adicionar o código, clique em Iniciar Simulação.**
2. **Observe os LEDs acenderem na sequência correta:**
   - Vermelho → 5 segundos.
   - Amarelo → 2 segundos.
   - Verde → 5 segundos.

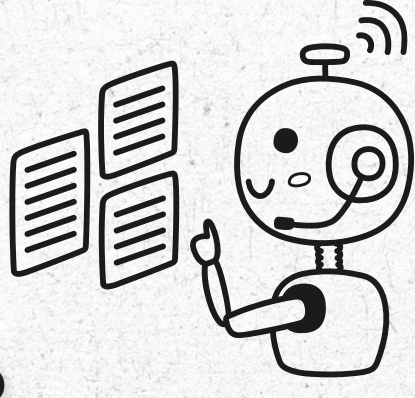
Dicas para Melhorar o Projeto

Adicione botões: Simule um botão de pedestre para alterar o ciclo do semáforo.

Mude os tempos: Ajuste os valores do delay() para representar diferentes cenários, como um semáforo de pedestres.
Crie um semáforo duplo: Adicione mais LEDs para simular um cruzamento.

CONCLUSÃO

Você agora entende como controlar múltiplas saídas digitais usando o Arduino. Este projeto introduziu conceitos importantes para projetos mais complexos, como lógica sequencial e controle de tempo.
No próximo capítulo, vamos explorar o uso de sensores analógicos, começando com um Sensor de Luminosidade com LDR.



PROJETO 03

SENSOR DE LUMINOSIDADE COM LDR

Neste capítulo, vamos explorar a leitura de um sensor analógico utilizando um **LDR (Light Dependent Resistor)**. Este projeto ensinará como o Arduino pode interpretar dados do ambiente e tomar decisões baseadas nesses dados.

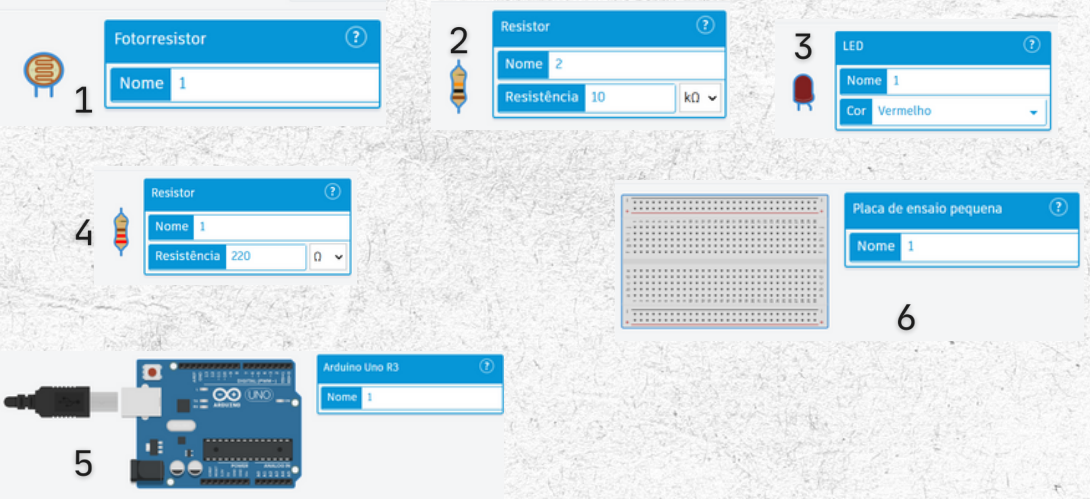
OBJETIVO DO PROJETO

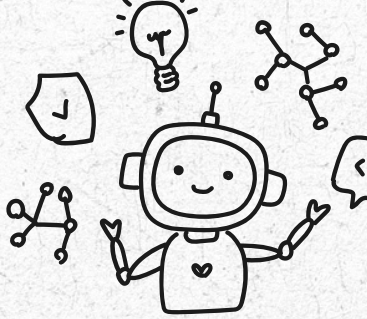
Criar um sistema que detecta a intensidade de luz usando um LDR e acende ou apaga um LED dependendo da luminosidade.

COMPONENTES VIRTUAIS NECESSÁRIOS

1. 1 LDR (Resistor Dependente de Luz).
2. 1 Resistor de 10 kΩ.
3. 1 LED (por exemplo, vermelho).
4. 1 Resistor de 220 ohms (para o LED).
5. Placa Arduino Uno.
6. Protoboard (placa de ensaio).

Fios de conexão.





PASSO 1: MONTAGEM DO CIRCUITO

1. **Abra o Tinkercad e crie um novo projeto.**

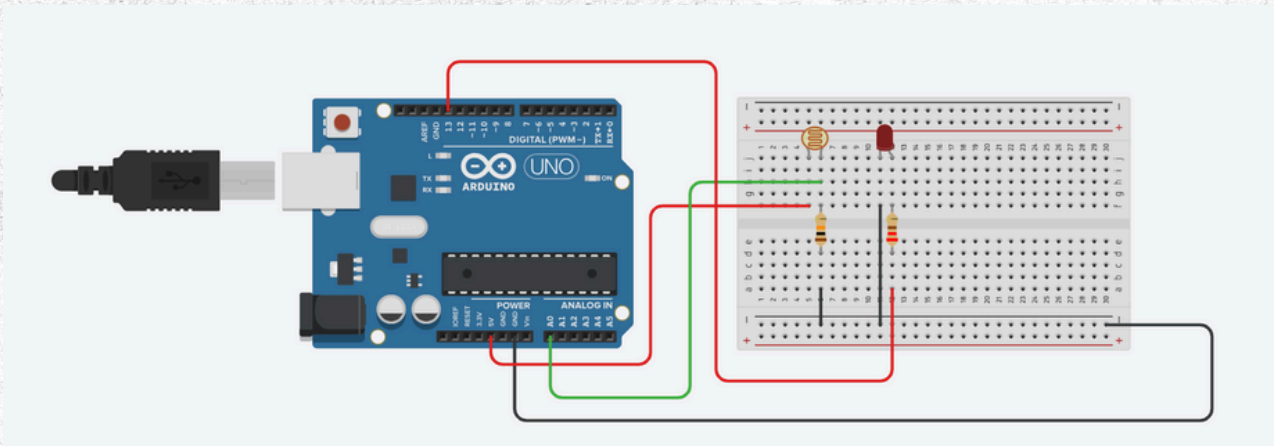
2. **Conecte o LDR:**
   - Conecte uma extremidade do LDR ao pino de 5V do Arduino.
   - Conecte a outra extremidade ao pino A0 (entrada analógica do Arduino).
   - Insira um resistor de 10 kΩ entre o pino A0 e o GND.

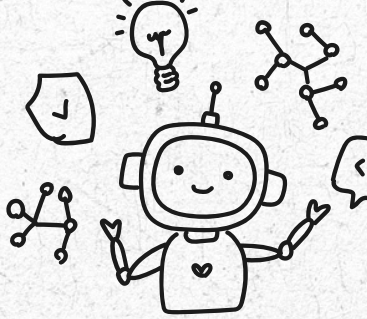
3. **Conexões Resumidas:**
   - LDR entre 5V e pino A0.
   - Resistor de 10 kΩ entre A0 e GND.
   - LED entre pino 13 e GND, com resistor.

Esquema do Circuito

O circuito no Tinkercad deve parecer algo assim:

- O LDR forma um divisor de tensão com o resistor de 10 kΩ, permitindo medir a luz ambiente.
- O LED é conectado ao pino digital 13.

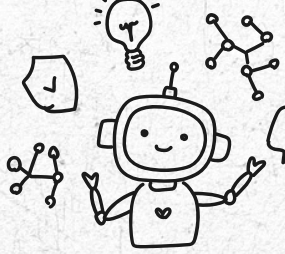




PASSO 2: O CÓDIGO EM ARDUINO

Abaixo está o código para ler o valor do LDR e acender/apagar o LED dependendo da luminosidade

```
int ldrPin = A0;  // Pino conectado ao LDR
int ledPin = 13;  // Pino conectado ao LED
int threshold = 500; // Limiar de luminosidade

void setup() {
  pinMode(ledPin, OUTPUT); // Configura o LED como saída
  Serial.begin(9600);      // Inicializa a comunicação serial para depuração
}

void loop() {
  int ldrValue = analogRead(ldrPin); // Lê o valor do LDR
  Serial.println(ldrValue);          // Exibe o valor no monitor serial

  if (ldrValue < threshold) { // Se o valor do LDR for menor que o limiar
    digitalWrite(ledPin, HIGH); // Liga o LED
  } else {
    digitalWrite(ledPin, LOW);  // Desliga o LED
  }

  delay(500); // Aguarda meio segundo
}
```



Explicação do Código

1. **Variáveis:**
   - ldrPin e ledPin armazenam os pinos conectados ao LDR e ao LED.
   - threshold define o limite de luz para acender o LED.
2. **setup():**
   - Configura o pino do LED como saída.
   - Inicializa a comunicação serial para monitorar os valores do LDR.
3. **loop():**
   - Lê o valor do LDR usando analogRead(). O valor varia de 0 (escuro) a 1023 (muito claro).
   - Compara o valor com o limite (threshold).
   - Liga o LED se estiver escuro; desliga caso contrário.

PASSO 3: TESTANDO NO TINKERCAD

1. Monte o circuito e insira o código no simulador.
2. Clique em Iniciar Simulação.
3. Abra o Monitor Serial no Tinkercad para visualizar os valores do LDR.

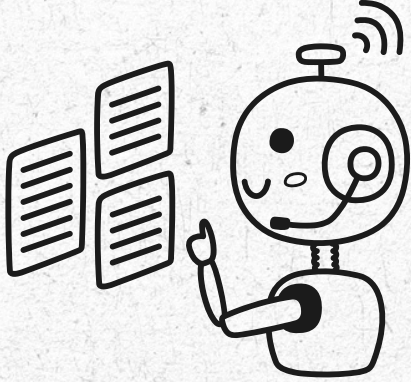
Simule mudanças de luz movendo o cursor sobre o LDR no circuito virtual. Você verá o LED acender e apagar conforme a luz detectada

Dicas para Melhorar o Projeto

- **Limiar Dinâmico: Adicione um potenciômetro para ajustar o valor de threshold em tempo real.**
- **LED RGB: Substitua o LED por um LED RGB e mude a cor dependendo da intensidade de luz.**
- **Mais Sensores: Use múltiplos LDRs para medir diferentes níveis de luz em locais distintos.**

CONCLUSÃO

Neste projeto, você aprendeu a usar entradas analógicas no Arduino para medir a luz ambiente. Isso abre portas para criar dispositivos como lâmpadas automáticas ou sistemas de monitoramento de iluminação.



PROJETO 04

CONTROLE DE SERVO MOTOR

Neste capítulo, vamos explorar como controlar um servo motor com o Arduino. Este projeto introduz o conceito de controle de ângulo usando sinais PWM (Modulação por Largura de Pulso), que é essencial para sistemas mecânicos e robóticos.

OBJETIVO DO PROJETO

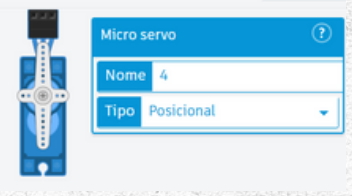
Controlar o ângulo de um servo motor por meio do Arduino, configurando diferentes posições (0°, 90° e 180°).

COMPONENTES VIRTUAIS NECESSÁRIOS

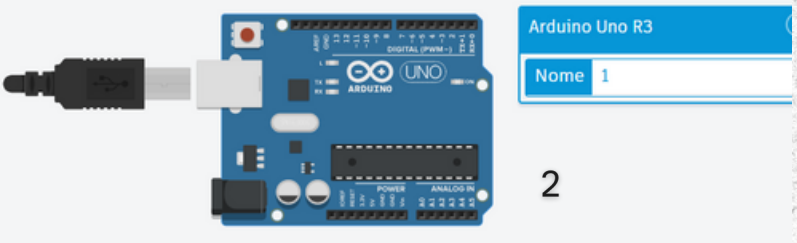
1. 1 Servo Motor (SG90 ou similar) ou Micro Servo.
2. Placa Arduino Uno.
3. Protoboard (opcional, para conexões organizadas).

Fios de conexão.

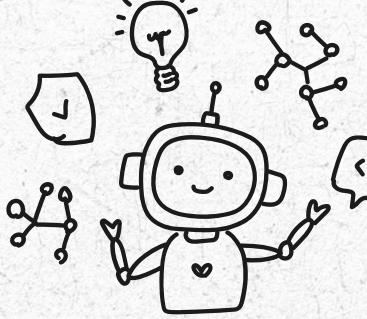
1



3



2



PASSO 1: MONTAGEM DO CIRCUITO

1. **Abra o Tinkercad e crie um novo projeto.**

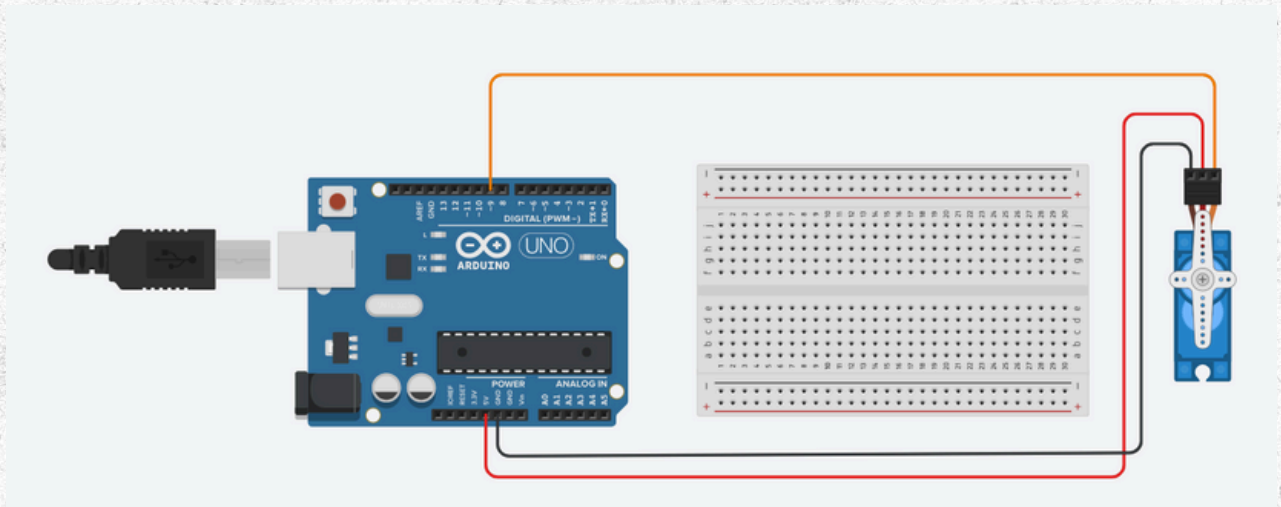
2. **Conecte o Servo Motor:**
   - Conecte o fio vermelho do servo ao pino de 5V do Arduino.
   - Conecte o fio preto/marrom ao pino GND.
   - Conecte o fio amarelo/laranja (sinal) ao pino digital 9 do Arduino.

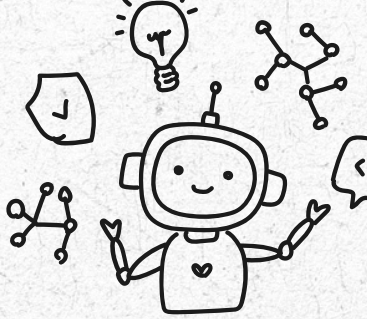
3. **Conexões Resumidas:**
   - 5V → Vermelho do servo.
   - GND → Preto/marrom do servo.
   - Pino 9 → Amarelo/laranja do servo.

Esquema do Circuito

O circuito no Tinkercad será simples:

- Servo conectado diretamente ao Arduino.
- Alimentação do servo pelo Arduino.

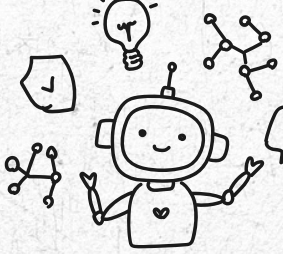




PASSO 2: O CÓDIGO EM ARDUINO

Abaixo está o código para controlar o servo motor, configurando-o em diferentes ângulos.

```
#include <Servo.h> // Inclui a biblioteca Servo.h

Servo myServo; // Cria um objeto Servo

void setup() {
  myServo.attach(9); // Conecta o servo ao pino 9
}

void loop() {
  myServo.write(0); // Move o servo para 0 graus
  delay(1000);      // Espera 1 segundo

  myServo.write(90); // Move o servo para 90 graus
  delay(1000);       // Espera 1 segundo

  myServo.write(180); // Move o servo para 180 graus
  delay(1000);        // Espera 1 segundo
}
```



Explicação do Código

1. #include <Servo.h>: Adiciona a biblioteca que facilita o controle de servo motores.
2. Servo myServo;: Cria um objeto para controlar o servo.
3. myServo.attach(9);: Conecta o servo ao pino digital 9.
4. myServo.write(ângulo);: Define o ângulo do servo (entre 0° e 180°).
5. delay(tempo);: Pausa o código para que o servo complete o movimento.

PASSO 3: TESTANDO NO TINKERCAD

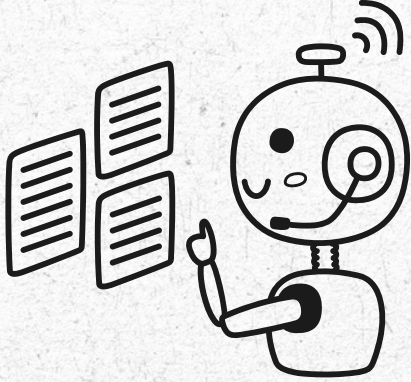
1. Monte o circuito e insira o código no simulador.
2. Clique em Iniciar Simulação.
3. Observe o servo motor se movendo para os ângulos 0°, 90° e 180°, com intervalos de 1 segundo.

Dicas para Melhorar o Projeto

- Controle por Potenciômetro: Conecte um potenciômetro ao Arduino para controlar o ângulo do servo manualmente.
- Vários Servos: Adicione mais servos conectados a diferentes pinos para criar sistemas mecânicos complexos.
- Automação: Integre sensores, como um ultrassônico, para mover o servo com base em entradas do ambiente.

CONCLUSÃO

Neste projeto, você aprendeu a controlar um servo motor com o Arduino, usando a biblioteca Servo.h. Este conhecimento é essencial para projetos de robótica e automação que exigem movimentos precisos.



PROJETO 05

TERMÔMETRO COM O SENSOR LM35

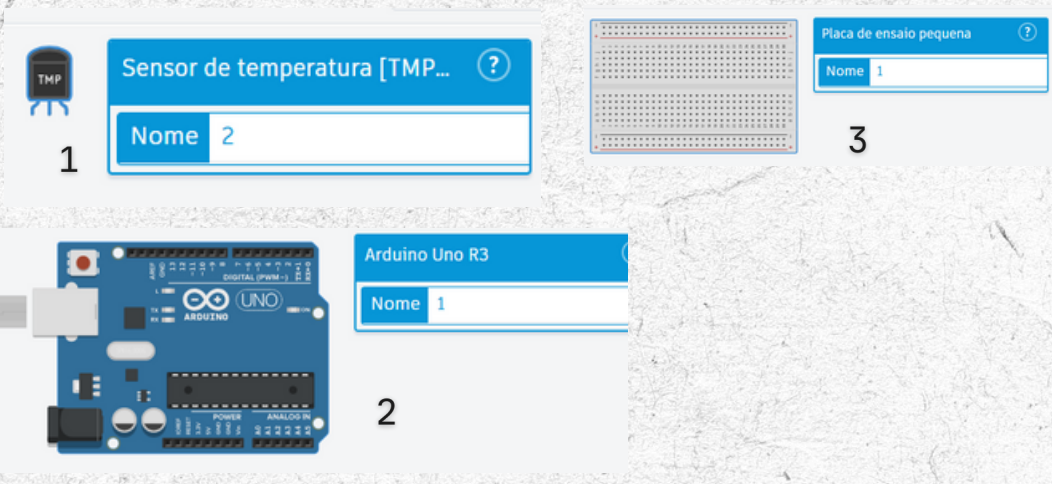
Neste capítulo, construiremos um termômetro básico usando o sensor de temperatura LM35. Você aprenderá como o Arduino pode medir temperatura e exibir os valores no Monitor Serial ou em um display.

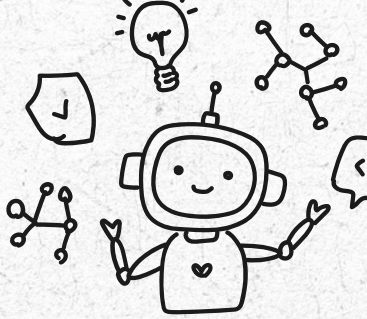
OBJETIVO DO PROJETO

Criar um termômetro simples que lê a temperatura ambiente em graus Celsius e exibe o valor no Monitor Serial.

COMPONENTES VIRTUAIS NECESSÁRIOS

1. 1 Sensor TMP.
2. Placa Arduino Uno.
3. Protoboard.
4. Fios de conexão.





PASSO 1: MONTAGEM DO CIRCUITO

1. **Abra o Tinkercad e crie um novo projeto.**

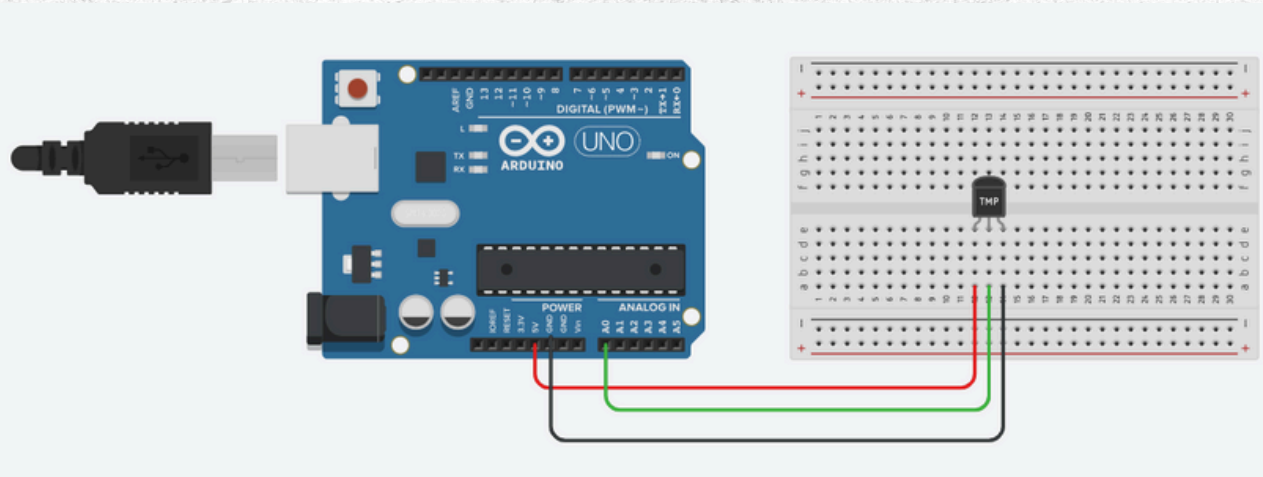
2. **Conecte o TMP:**
   - O pino 1 (VCC) do sensor deve ser conectado ao pino de 5V do Arduino.
   - O pino 2 (Saída) deve ser conectado ao pino A0 do Arduino.
   - O pino 3 (GND) deve ser conectado ao GND do Arduino.

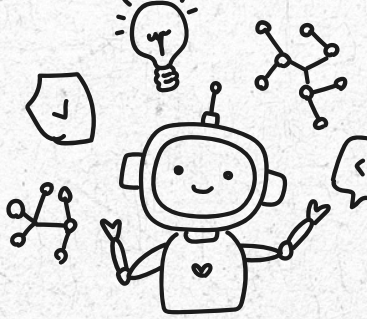
3. **Conexões Resumidas:**
   - VCC (pino 1) → 5V.
   - Saída (pino 2) → A0.
   - GND (pino 3) → GND.

Esquema do Circuito

O circuito no Tinkercad será simples:

- O TMP conectado diretamente ao Arduino.
- O sinal analógico gerado pelo LM35 será lido no pino A0.

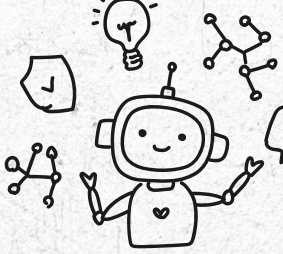




PASSO 2: O CÓDIGO EM ARDUINO

Abaixo está o código para medir a temperatura com o TMP e exibi-la no Monitor Serial.

```
const int sensorPin = A0; // Pino onde o LM35 está conectado

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

void loop() {
  int sensorValue = analogRead(sensorPin); // Lê o valor do sensor (0-1023)
  float voltage = sensorValue * (5.0 / 1023.0); // Converte o valor para tensão (em volts)
  float temperature = voltage * 100; // Converte a tensão para temperatura em Celsius

  Serial.print("Temperatura: ");
  Serial.print(temperature);
  Serial.println(" °C");

  delay(1000); // Atualiza a cada 1 segundo
}
```



Explicação do Código

1. Leitura do LM35: A saída do LM35 é proporcional à temperatura (10 mV por grau Celsius). O Arduino lê esse valor como um número entre 0 e 1023.
2. Conversão para Voltagem: sensorValue * (5.0 / 1023.0) converte o valor lido para tensão em volts.
3. Conversão para Temperatura: Multiplicamos a voltagem por 100 para obter a temperatura em graus Celsius, conforme especificação do LM35.
4. Monitor Serial: Os valores são exibidos no Monitor Serial do Arduino.

PASSO 3: TESTANDO NO TINKERCAD

1. Monte o circuito e insira o código no simulador.
2. Clique em Iniciar Simulação.
3. Abra o Monitor Serial no Tinkercad.
4. Observe a temperatura sendo exibida no Monitor Serial, atualizada a cada segundo.

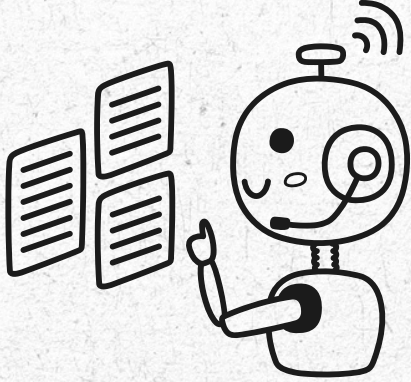
Dicas para Melhorar o Projeto

- Unidades Alternativas: Adicione uma conversão para Fahrenheit, caso necessário:
  `float temperatureF = (temperature * 9.0 / 5.0) + 32.0;`

- Display de Dados: Use um display LCD ou OLED para exibir a temperatura.
- Automação: Adicione um ventilador ou um LED que liga quando a temperatura ultrapassa um valor específico.

CONCLUSÃO

Neste projeto, você aprendeu a usar o sensor LM35 para medir a temperatura e exibi-la no Monitor Serial. Isso é útil para sistemas de monitoramento ambiental ou automação baseada em temperatura.



PROJETO 06

ALARME SONORO COM BUZZER PIEZOTRO COM O SENSOR LM35

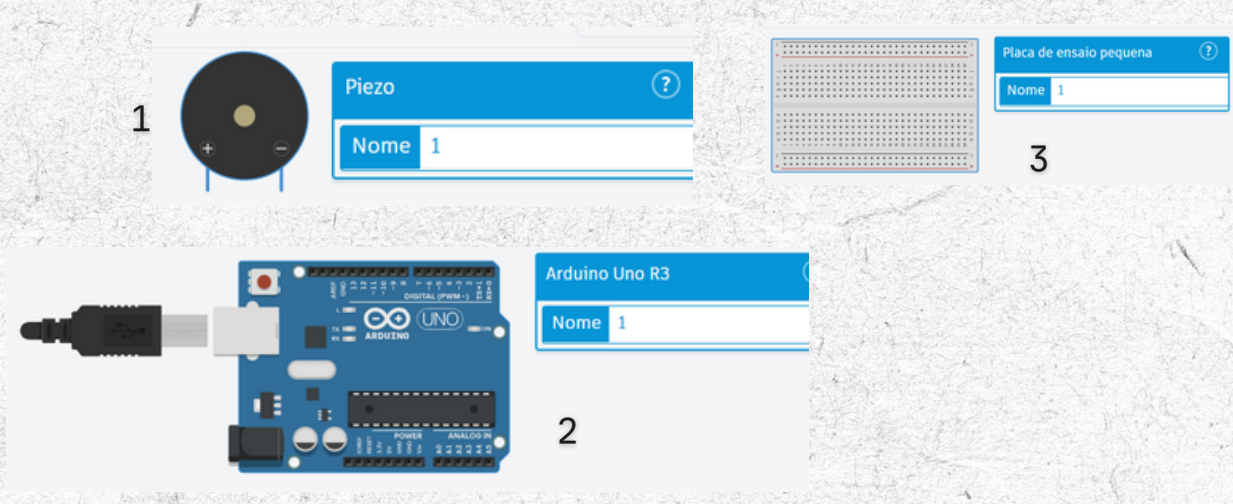
Neste capítulo, vamos criar um alarme sonoro simples usando um Buzzer Piezo. Este projeto introduz conceitos básicos de som em dispositivos eletrônicos e como gerar diferentes tons com o Arduino.

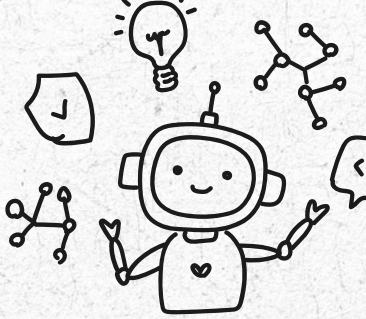
OBJETIVO DO PROJETO

Usar um buzzer piezoelétrico para emitir sons e criar um alarme simples com diferentes tons.

COMPONENTES VIRTUAIS NECESSÁRIOS

1. 1 Buzzer Piezo.
2. Placa Arduino Uno.
3. Protoboard (opcional, para organizar as conexões).
4. Fios de conexão.





PASSO 1: MONTAGEM DO CIRCUITO

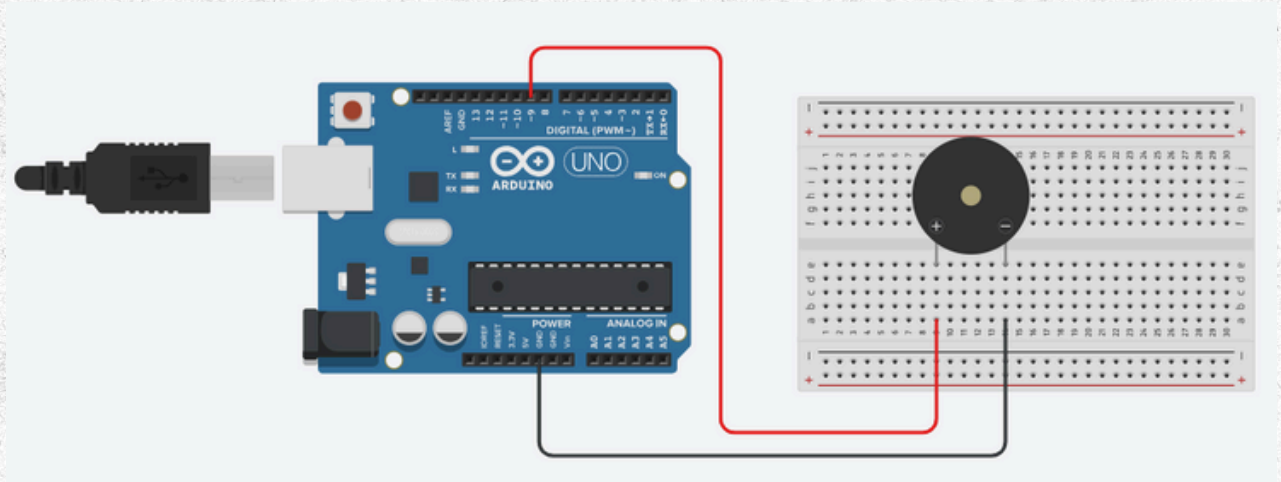
1. **Abra o Tinkercad e crie um novo projeto.**

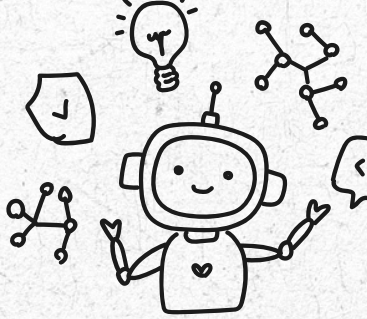
2. **Conecte o Buzzer Piezo:**
    - O terminal positivo do buzzer (geralmente marcado com um +) deve ser conectado ao pino 9 do Arduino.
    - O terminal negativo deve ser conectado ao GND do Arduino.

Esquema do Circuito

- Pino 9 → Terminal positivo do buzzer.
- GND → Terminal negativo do buzzer.
-

Este é um circuito extremamente simples e direto, ideal para iniciantes.

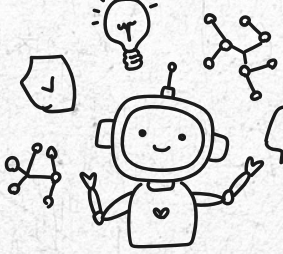




PASSO 2: O CÓDIGO EM ARDUINO

Abaixo está o código para gerar tons diferentes no buzzer piezo.

```
int buzzerPin = 9; // Pino onde o buzzer está conectado

void setup() {
  pinMode(buzzerPin, OUTPUT); // Configura o pino do buzzer como
saída
}

void loop() {
  tone(buzzerPin, 1000); // Emite um tom de 1000 Hz
  delay(1000);           // Mantém o tom por 1 segundo
  noTone(buzzerPin);      // Para o tom
  delay(500);            // Pausa de 0,5 segundo

  tone(buzzerPin, 1500); // Emite um tom de 1500 Hz
  delay(1000);           // Mantém o tom por 1 segundo
  noTone(buzzerPin);      // Para o tom
  delay(500);            // Pausa de 0,5 segundo
}
```



Explicação do Código

1. tone(buzzerPin, frequência): Gera um som na frequência especificada (em Hertz). Por exemplo:
   - 1000 Hz produz um som médio.
   - 1500 Hz produz um som mais agudo.
2. noTone(buzzerPin): Interrompe o som no pino especificado.
3. Alternância de Tons: O loop alterna entre dois tons (1000 Hz e 1500 Hz) com pausas entre eles.

PASSO 3: TESTANDO NO TINKERCAD

1. Monte o circuito e insira o código no simulador.
2. Clique em Iniciar Simulação.
3. Ouça os sons gerados pelo buzzer. Eles devem alternar entre tons médios e agudos.

Dicas para Melhorar o Projeto

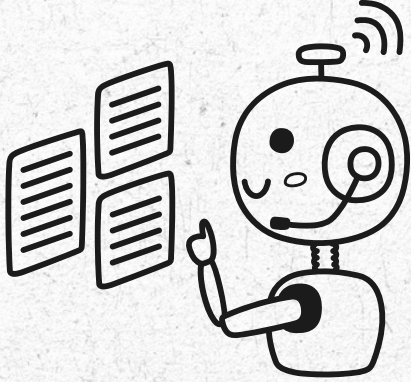
- Sons Personalizados: Crie um padrão de tons mais complexo, como este exemplo:

```
tone(buzzerPin, 500); delay(300); noTone(buzzerPin);
tone(buzzerPin, 700); delay(300); noTone(buzzerPin);
tone(buzzerPin, 900); delay(300); noTone(buzzerPin);
```

- Alarme Automático: Combine com sensores, como um sensor ultrassônico, para ativar o alarme quando algo for detectado.
- Melodias: Use bibliotecas como pitches.h para tocar músicas no buzzer.

CONCLUSÃO

Neste projeto, você aprendeu a usar o sensor LM35 para medir a temperatura e exibi-la no Monitor Serial. Isso é útil para sistemas de monitoramento ambiental ou automação baseada em temperatura.



PROJETO 07

DISPLAY DE 7 SEGMENTOS

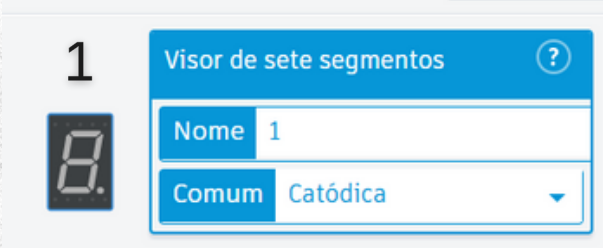
Neste capítulo, vamos aprender a controlar um Display de 7 Segmentos usando o Arduino. Este projeto é ideal para criar contadores simples, relógios ou qualquer sistema que precise exibir números.

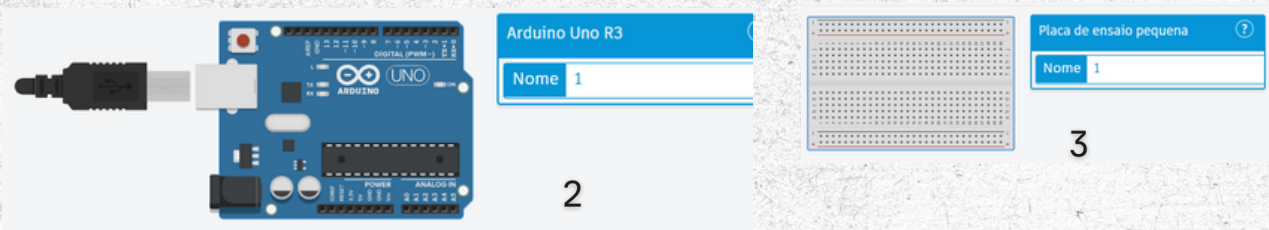
OBJETIVO DO PROJETO

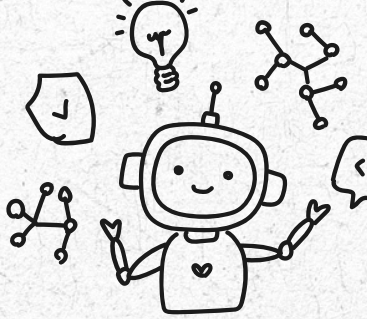
Configurar e controlar um Display de 7 Segmentos para exibir números de 0 a 9.

COMPONENTES VIRTUAIS NECESSÁRIOS

1. 1 Display (Visor) de 7 Segmentos (comum catódica).
2. 8 Resistores de 220 ohms (um para cada segmento).
3. Placa Arduino Uno.
4. Protoboard.
5. Fios de conexão.







PASSO 1: MONTAGEM DO CIRCUITO

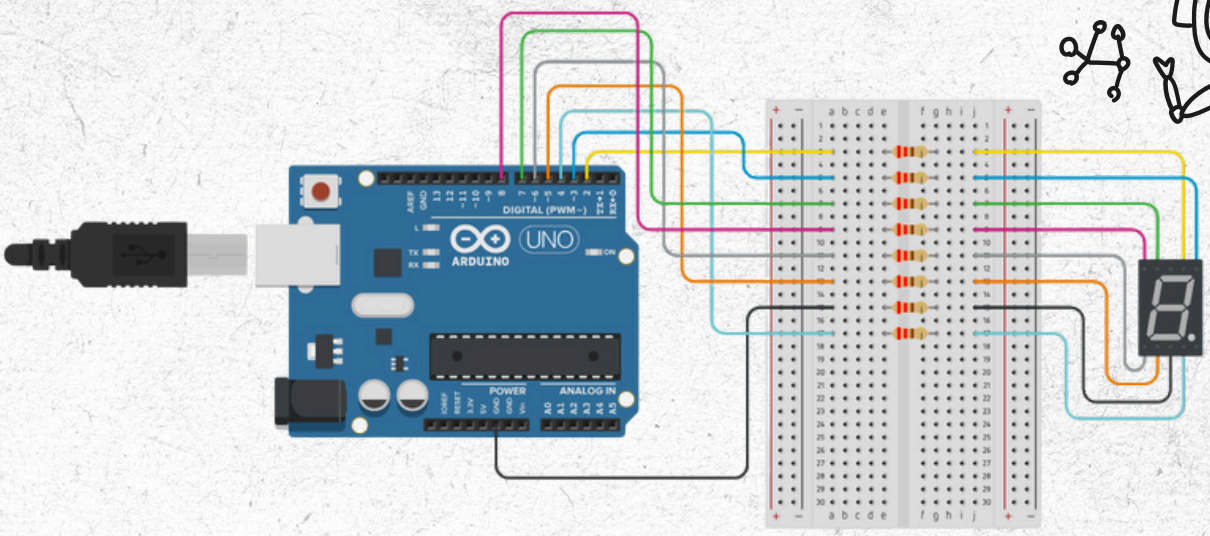
1. **Identifique os pinos do display:**
    - Os 7 segmentos do display são nomeados como a, b, c, d, e, f, g, e o ponto decimal é chamado de dp.
    - Cada segmento deve ser conectado a um pino do Arduino através de um resistor de 220 ohms para limitar a corrente.
    - O pino comum (cátodo) do display será conectado ao GND.

2. **Conecte os pinos do display ao Arduino:**
    - Conecte os segmentos:
        - a → Pino 2 do Arduino.
        - b → Pino 3.
        - c → Pino 4.
        - d → Pino 5.
        - e → Pino 6.
        - f → Pino 7.
        - g → Pino 8.
        - dp (opcional) → Deixe desconectado ou conecte ao pino 9 para ativar o ponto decimal.
    - Conecte o pino comum (cátodo) ao GND.

3. **Use uma protoboard para facilitar as conexões.**

Esquema do Circuito

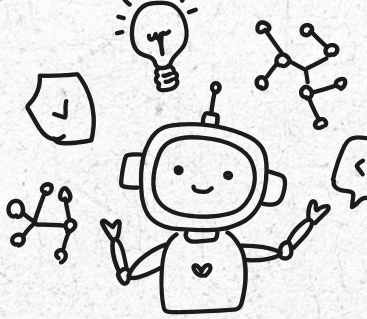
No Tinkercad, o circuito será montado com resistores em série entre os pinos do Arduino e os segmentos do display. Certifique-se de conectar o pino comum ao GND.



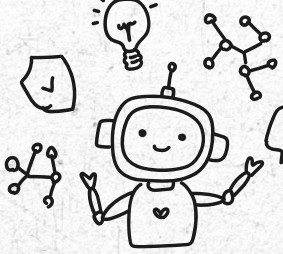
PASSO 2: O CÓDIGO EM ARDUINO

Abaixo está o código para exibir números de 0 a 9 no display.

```
// Defina os pinos conectados aos segmentos do display de 7
segmentos
const int segmentos[] = {2, 3, 4, 5, 6, 7, 8}; // A, B, C, D, E,
F, G

// Mapeamento de números para segmentos (0-9)
const byte numeros[10] = {
  0b00111111, // 0
  0b00000110, // 1
  0b01011011, // 2
  0b01001111, // 3
  0b01100110, // 4
  0b01101101, // 5
  0b01111101, // 6
  0b00000111, // 7
  0b01111111, // 8
  0b01101111  // 9
};
```



```
void setup() {
  // Configurar os pinos dos segmentos como saída
  for (int i = 0; i < 7; i++) {
    pinMode(segmentos[i], OUTPUT);
    digitalWrite(segmentos[i], LOW); // Inicializa desligado
  }
}

void loop() {
  // Exibe os números de 0 a 9
  for (int num = 0; num <= 9; num++) {
    exibirNumero(num);
    delay(1000); // Espera 1 segundo antes de mudar
  }
}

// Função para exibir um número no display de 7 segmentos
void exibirNumero(int numero) {
  for (int i = 0; i < 7; i++) {
    // Liga ou desliga cada segmento com base no mapeamento
    digitalWrite(segmentos[i], (numeros[numero] & (1 << i)) ? HIGH
: LOW);
  }
}
```



Explicação do Código

1. **const int segments[]**: Um array que mapeia os pinos do Arduino para os segmentos do display.
2. **const byte numeros[10]**: Um array de bytes que representa os estados dos segmentos para exibir cada número de 0 a 9.
3. **exibirNumero(num)**: Ativa ou desativa os segmentos necessários para formar o número especificado.

PASSO 3: TESTANDO NO TINKERCAD

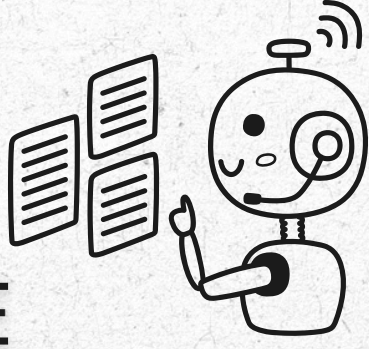
1. Monte o circuito e insira o código no simulador.
2. Clique em Iniciar Simulação.
3. Observe o display de 7 segmentos alternando entre os números de 0 a 9, um por segundo.

Dicas para Melhorar o Projeto

- Mais Dígitos: Use múltiplos displays para exibir números com mais de um dígito. Neste caso, será necessário usar drivers ou técnicas como multiplexação.
- Adicione um Botão: Controle manualmente os números exibidos adicionando um botão para avançar entre os números.
- Ative o Ponto Decimal: Use o ponto decimal para indicar estados adicionais, como erro ou unidade.

CONCLUSÃO

Você agora entende como controlar um display de 7 segmentos para exibir números. Este projeto é a base para criar contadores, relógios e outras ferramentas que exibem informações numéricas.



PROJETO 08

SENSOR DE PROXIMIDADE COM ULTRASSÔNICO HC-SR04

NNeste capítulo, vamos explorar o uso do sensor ultrassônico HC-SR04 para medir distâncias. Este projeto introduz conceitos importantes de sensores digitais e cálculos baseados em tempo e velocidade do som.

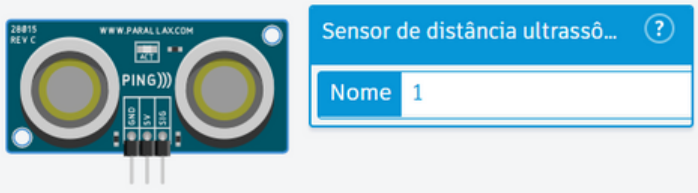
OBJETIVO DO PROJETO

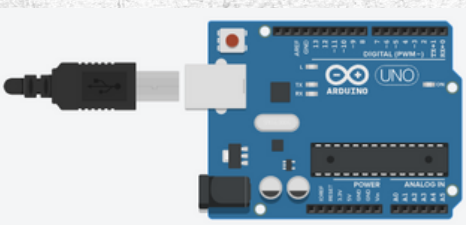
Medir a distância de objetos usando o sensor HC-SR04 e exibi-la no Monitor Serial do Arduino. Além disso, adicionaremos um LED de alerta para indicar quando a distância medida for menor que um valor específico.

COMPONENTES VIRTUAIS NECESSÁRIOS

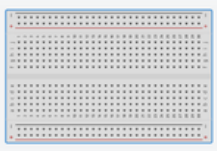
1. Sensor PING Ultrassônico
2. Placa Arduino Uno
3. Protoboard (opcional)
4. Fios de conexão
5. 1 LED e resistor de 220Ω (opcional, para indicação de proximidade)

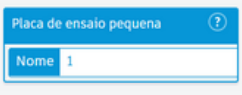
1



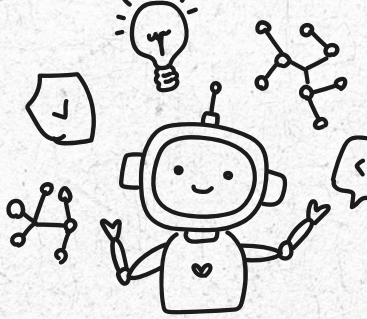


2





3



PASSO 1: MONTAGEM DO CIRCUITO

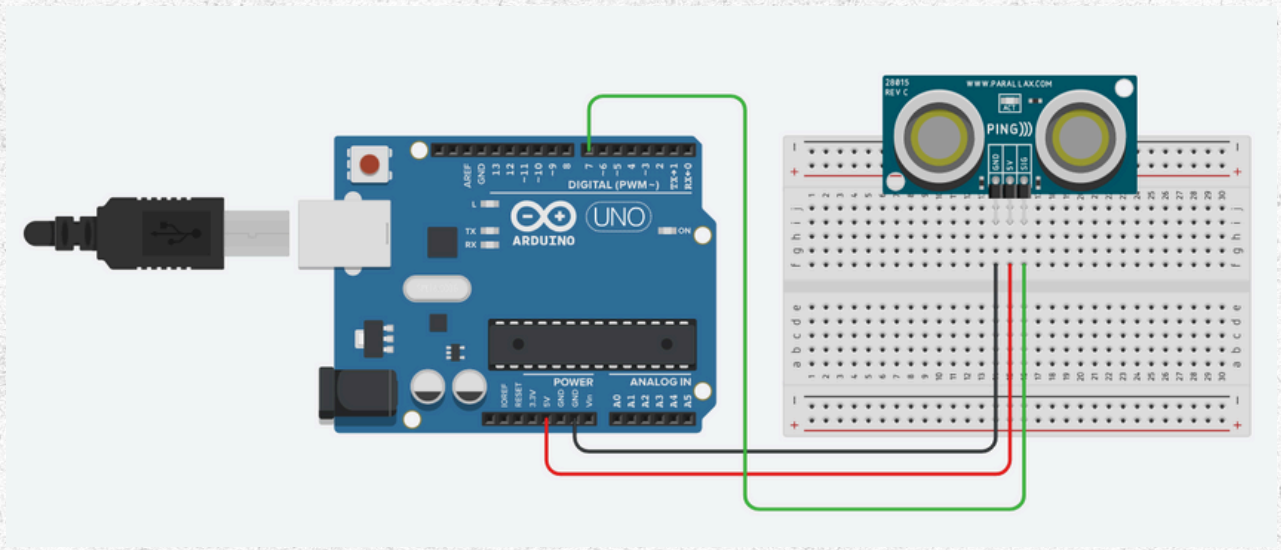
1. **Identifique os pinos do sensor HC-SR04:**

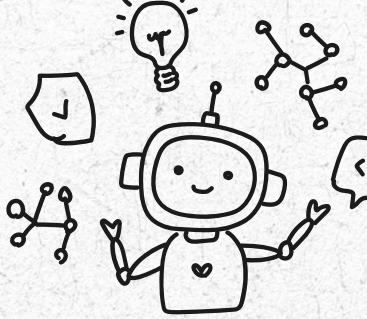
   - 5V: Alimentação (+5V).
   - GND: Terra.
   - SIG: Envia e Recebe o sinal ultrassônico.

2. **Conexões no Arduino:**
   - GND do sensor → GND do Arduino.
   - 5V do sensor → 5V do Arduino.
   - SIG do sensor → Pino Digital 7 no Arduino.
   - (Opcional) LED conectado ao pino digital 13 com um resistor de 220Ω.

Esquema do Circuito

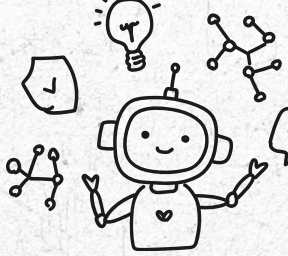
- 5V → 5V do sensor.
- GND → GND do sensor.
- Pino 7 → SIG do sensor.
- (Opcional) Pino 13 → LED em série com resistor de 220Ω.



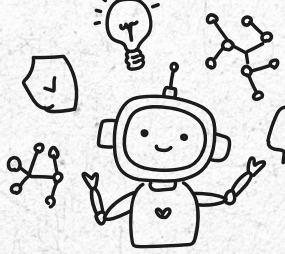


PASSO 2: O CÓDIGO EM ARDUINO

```
const int sigPin = 7;  // Pino SIG do sensor
const int ledPin = 13; // LED para alerta (opcional)

void setup() {
  pinMode(ledPin, OUTPUT); // Configura LED como saída (opcional)
  Serial.begin(9600);      // Inicializa a comunicação serial
}

void loop() {
  long duration;
  float distance;

  // Envia o pulso de trigger
  pinMode(sigPin, OUTPUT);// Configura SIG como saída
  digitalWrite(sigPin, LOW);// Garante que o pino começa em LOW
  delayMicroseconds(2);     // Atraso curto para estabilidade
  digitalWrite(sigPin, HIGH);// Pulso de 5 µs para início medição
  delayMicroseconds(5);
  digitalWrite(sigPin, LOW);
```



```
  // Muda SIG para entrada para ler o retorno do pulso
  pinMode(sigPin, INPUT);
  duration = pulseIn(sigPin, HIGH, 30000); // Mede o tempo do
pulso, limite de 30ms para evitar travamento

  // Verifica se houve um valor válido
  if (duration > 0) {
    // Calcula a distância em centímetros
    distance = duration * 0.034 / 2;

    // Exibe a distância no Monitor Serial
    Serial.print("Distância: ");
    Serial.print(distance);
    Serial.println(" cm");

    // Acende o LED se a distância for menor que 10 cm
    if (distance < 10) {
      digitalWrite(ledPin, HIGH);
    } else {
      digitalWrite(ledPin, LOW);
    }
  } else {
    // Caso nenhum pulso de retorno seja detectado, exibe mensagem
de erro
    Serial.println("Nenhum sinal detectado ou fora do alcance!");
    digitalWrite(ledPin, LOW); // Mantém o LED apagado
  }

  delay(500); // Aguarda 0,5 segundos antes de repetir
}
```

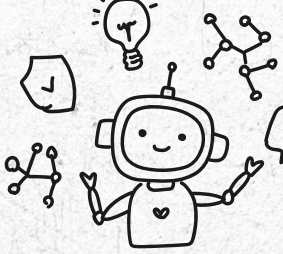


Explicação do Código

1. Pulso de Trigger:
   - Configura o pino SIG como saída.
   - Envia um pulso de 5 µs para iniciar a medição.
2. Leitura de Echo:
   - Configura o pino SIG como entrada.
   - Mede o tempo que o pulso levou para retornar utilizando pulseIn().
3. Cálculo da Distância:
   - Calcula a distância em centímetros: Distaˆncia=Durac¸a˜o×0.0342\text{Distância} = \frac{\text{Duração} \times 0.034}{2}Distaˆncia=2Durac¸a˜o×0.034
   - Onde 0.034 é a velocidade do som em cm/µs.
4. LED de Alerta (opcional):
   - Acende o LED se a distância medida for menor que 10 cm.
5. Monitor Serial:
   - Exibe a distância calculada no Monitor Serial.

PASSO 3: TESTANDO NO TINKERCAD

1. Monte o circuito conforme descrito.
2. Carregue o código no Arduino.
3. Abra o Monitor Serial no Arduino IDE para visualizar as distâncias medidas.
4. (Opcional) Verifique o comportamento do LED ao aproximar objetos do sensor.

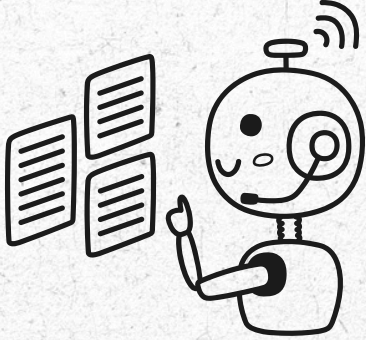


Dicas para Melhorar o Projeto

1. Display LCD ou OLED:
   - Adicione um display para exibir a distância sem depender do Monitor Serial.
2. Controle de Servo Motor:
   - Use a distância para controlar a posição de um servo motor, criando um sistema de barreira automatizada.
3. Buzzer de Alerta:
   - Inclua um buzzer que emita som quando um objeto estiver muito próximo.

CONCLUSÃO

Este projeto utiliza o sensor PING para medir distâncias e exibi-las no Monitor Serial. Com modificações simples, é possível expandir o projeto para aplicações práticas, como detectores de obstáculos, sistemas de segurança ou controle automático de dispositivos.



PROJETO 09

CONTROLE DE LEDS RGB COM POTENCIÔMETRO

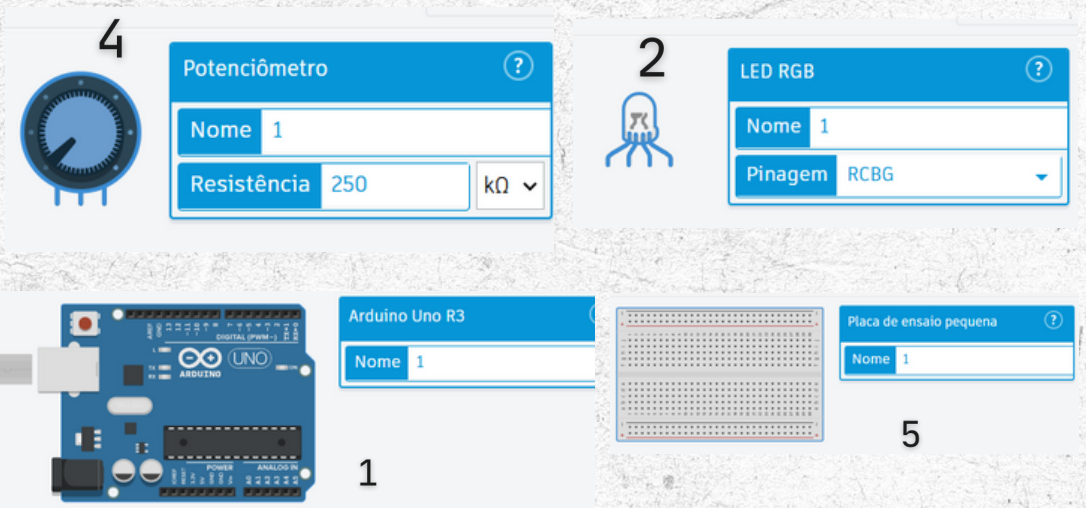
Neste capítulo, vamos explorar como usar um **Potenciômetro** para controlar as cores de um LED RGB. Este projeto combina entradas analógicas e digitais, permitindo controlar a intensidade das cores vermelho, verde e azul com movimentos do **Potenciômetro.**

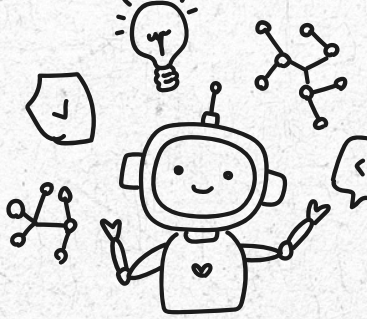
OBJETIVO DO PROJETO

Substituir o controle via joystick por um potenciômetro para ajustar a intensidade de um LED RGB ou controlar outros componentes analógicos.

COMPONENTES VIRTUAIS NECESSÁRIOS

1. Arduino Uno
2. LED RGB (com 4 terminais: ânodo ou cátodo comum)
3. 3 Resistores de 220Ω
4. 1 Potenciômetro (10 kΩ)
5. Protoboard
6. Fios de conexão





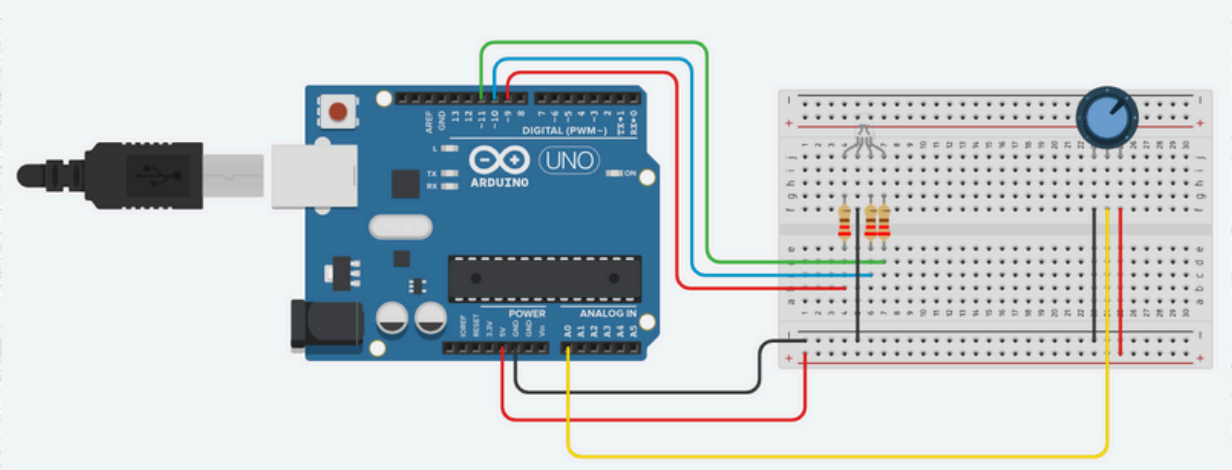
PASSO 1: MONTAGEM DO CIRCUITO

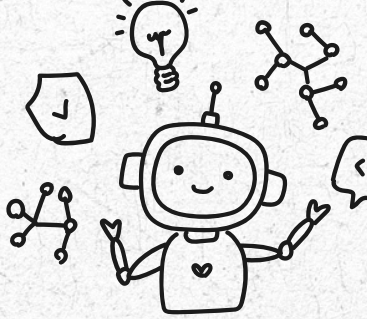
1. **LED RGB:**
    - Conecte o pino comum do LED RGB (cátodo comum ao GND ou ânodo comum ao 5V, dependendo do modelo).
    - Conecte os pinos R (vermelho), G (verde) e B (azul) aos pinos digitais 9, 10 e 11 do Arduino, respectivamente, usando resistores de 220Ω.

2. **Potenciômetro:**
    - Conecte o pino central do potenciômetro à entrada analógica A0 do Arduino.
    - Conecte um dos outros pinos do potenciômetro ao 5V e o outro ao GND.

Esquema do Circuito

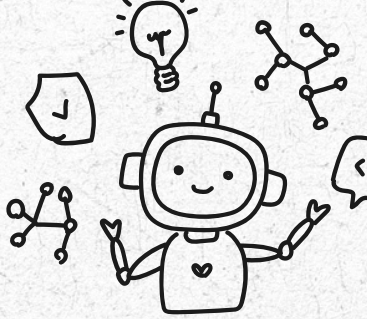
- Pinos digitais 9, 10, 11 → R, G, B do LED RGB.
- Pino analógico A0 → Saída central do potenciômetro.
- 5V e GND → Alimentação do potenciômetro e LED RGB.





PASSO 2: O CÓDIGO EM ARDUINO

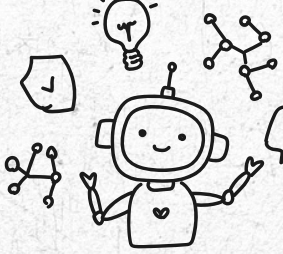
```
const int redPin = 9;   // Pino do LED vermelho
const int greenPin = 10; // Pino do LED verde
const int bluePin = 11; // Pino do LED azul
const int potPin = A0;  // Pino do potenciômetro

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  Serial.begin(9600); // Inicia a comunicação serial
}

void loop() {
  int potValue = analogRead(potPin); // Lê o valor do
potenciômetro (0-1023)

  // Mapeia o valor do potenciômetro para o intervalo PWM (0-255)
  int redValue = map(potValue, 0, 1023, 0, 255);
  int greenValue = map(potValue, 0, 1023, 255, 0); // Invertemos
para criar contraste
  int blueValue = map(potValue, 0, 1023, 0, 128);  // Ajusta para
uma cor intermediária

  // Ajusta a intensidade das cores no LED RGB
  analogWrite(redPin, redValue);
  analogWrite(greenPin, greenValue);
  analogWrite(bluePin, blueValue);
```



```
  // Envia os valores para o Monitor Serial
  Serial.print("Potenciômetro: ");
  Serial.print(potValue);
  Serial.print(" | Vermelho: ");
  Serial.print(redValue);
  Serial.print(" | Verde: ");
  Serial.print(greenValue);
  Serial.print(" | Azul: ");
  Serial.println(blueValue);

  delay(50); // Pequeno atraso para estabilidade
}
```

Explicação do Código

1. Definição dos pinos:
   - Os pinos digitais 9, 10 e 11 são configurados como saídas para controlar as cores do LED RGB.
   - O pino A0 lê o valor do potenciômetro.
2. Leitura do potenciômetro:
   - O valor lido pelo analogRead() varia de 0 a 1023.
   - A função map() converte esse valor para o intervalo PWM de 0 a 255.
3. Controle do LED RGB:
   - A função analogWrite() ajusta a intensidade de cada cor do LED RGB com base nos valores mapeados.
4. Monitor Serial:
   - Os valores lidos do potenciômetro e calculados para cada cor são exibidos no Monitor Serial para facilitar o monitoramento.
5. Delay:
   - Um atraso de 50 ms garante uma atualização fluida e estável dos valores.



PASSO 3: TESTANDO NO TINKERCAD

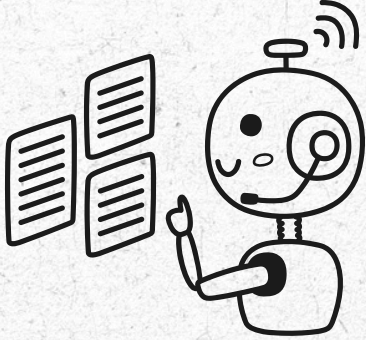
1. Monte o circuito no Tinkercad Arduino conforme descrito.
2. Carregue o código na simulação.
3. Gire o potenciômetro e observe a alteração na cor do LED RGB.

Dicas para Melhorar o Projeto

1. Controle de brilho geral:
    - Adicione um segundo potenciômetro para ajustar o brilho total do LED RGB.
2. Múltiplos LEDs:
    - Controle múltiplos LEDs RGB com valores diferentes.
3. Display LCD ou OLED:
    - Exiba os valores do potenciômetro e as intensidades das cores em um display.
    -

CONCLUSÃO

Este projeto substitui o joystick por um potenciômetro, mantendo a funcionalidade de controle analógico. É ideal para aprender a usar entradas analógicas no Arduino e controlar saídas PWM.



PROJETO 09

IRRIGAÇÃO AUTOMÁTICA COM SENSOR DE UMIDADE DO SOLO

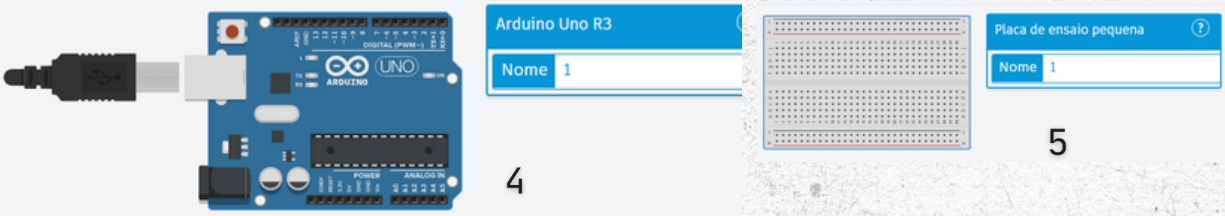
Neste capítulo, você aprenderá a construir um sistema de irrigação automática que utiliza um sensor de umidade do solo para monitorar os níveis de água no solo e ativar ou desativar uma bomba de água (ou um LED representando a bomba) automaticamente.

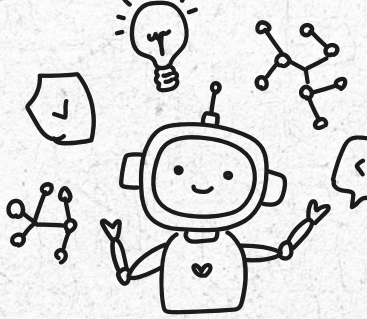
OBJETIVO DO PROJETO

Criar um sistema de irrigação automática que mede a umidade do solo e aciona uma bomba de água quando o solo estiver seco.

COMPONENTES VIRTUAIS NECESSÁRIOS

1. 1 Sensor de Umidade do Solo.
2. 1 LED (para simular a bomba de água).
3. 1 Resistor de 220 ohms (para o LED).
4. Placa Arduino Uno.
5. Protoboard.
6. Fios de conexão.





PASSO 1: MONTAGEM DO CIRCUITO

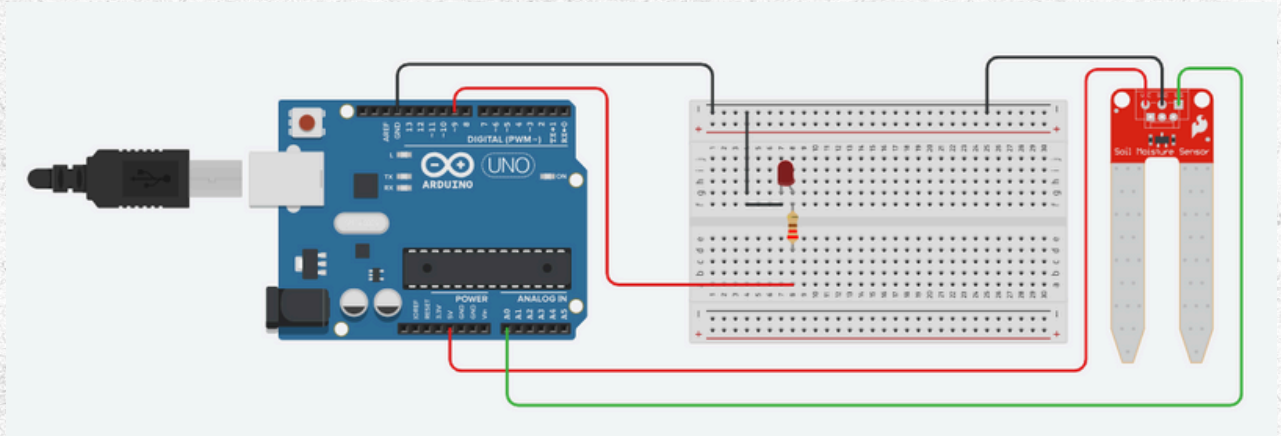
1. **Conecte o Sensor de Umidade do Solo:**

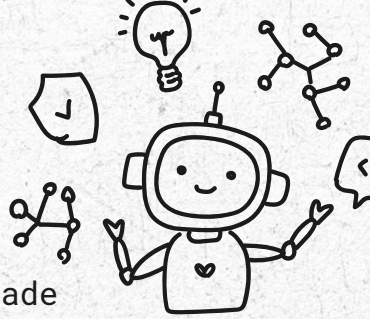
- O sensor possui três pinos:
  - VCC: Conecte ao pino 5V do Arduino.
  - GND: Conecte ao GND do Arduino.
  - AO: Saída analógica do sensor, conecte ao pino A0 do Arduino.

2. **Conecte o LED (representando a bomba):**
   - Conecte o terminal positivo (anodo) do LED ao pino 9 do Arduino, através de um resistor de 220 ohms.
   - Conecte o terminal negativo (catodo) ao GND.

Esquema do Circuito

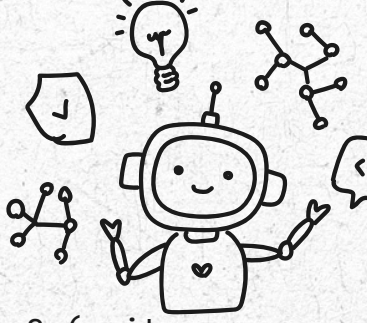
- O sensor mede a umidade do solo e envia o valor ao pino analógico A0.
- O LED no pino 9 representa a ativação da bomba de água.





PASSO 2: O CÓDIGO EM ARDUINO

```
const int soilSensorPin = A0; // Pino do sensor de umidade
const int ledPin = 9;         // Pino do LED (simulando a bomba)

int threshold = 600; // Limiar de umidade (ajuste conforme
necessário)

void setup() {
  pinMode(ledPin, OUTPUT);   // Configura o LED como saída
  Serial.begin(9600);        // Inicializa a comunicação serial
}

void loop() {
  int soilValue = analogRead(soilSensorPin); // Lê o valor do
sensor de umidade

  Serial.print("Umidade do Solo: ");
  Serial.println(soilValue); // Exibe o valor no monitor serial

  if (soilValue > threshold) {
    digitalWrite(ledPin, HIGH); // Liga o LED (bomba de água)
    Serial.println("Solo seco - Bomba ligada");
  } else {
    digitalWrite(ledPin, LOW); // Desliga o LED (bomba de água)
    Serial.println("Solo úmido - Bomba desligada");
  }

  delay(1000); // Aguarda 1 segundo antes de repetir
}
```



Explicação do Código

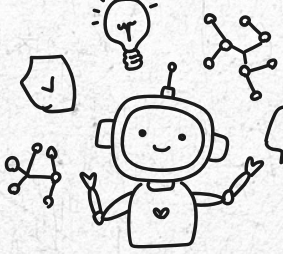
1. Leitura do Sensor:
   - O sensor de umidade envia um valor analógico entre 0 (muito úmido) e 1023 (muito seco).
2. Limiar de Umidade (threshold):
   - O valor definido em threshold determina quando o solo é considerado seco. Ajuste conforme o ambiente.
3. Ativação da Bomba:
   - Se o valor do sensor for maior que o limiar, o LED (bomba) acende.
   - Caso contrário, o LED permanece apagado.

PASSO 3: TESTANDO NO TINKERCAD

1. Monte o circuito e insira o código no simulador.
2. Clique em Iniciar Simulação.
3. Ajuste manualmente o nível de umidade do solo no simulador do Tinkercad para observar o LED acendendo ou apagando.

Dicas para Melhorar o Projeto

- Bomba Real: Substitua o LED por um relé que controle uma bomba de água real.
- Exibição em Display: Adicione um display LCD para mostrar os níveis de umidade.
- Alerta Sonoro: Use um buzzer para emitir um som quando o solo estiver muito seco.
- Sistema Autônomo: Acrescente um painel solar para alimentar o sistema, tornando-o completamente autônomo.



CONCLUSÃO

Neste projeto, você aprendeu a criar um sistema básico de irrigação automática usando um sensor de umidade do solo. Este sistema pode ser adaptado para aplicações reais em pequenos jardins ou plantações, promovendo o uso eficiente da água.