Controle de Buzzer com Arduino

Objetivo: Ensinar a controlar um atuador sonoro (Buzzer) usando o Arduino, explorando sons básicos e interações com componentes eletrônicos simples.

1. Introdução ao Buzzer (10 minutos)

Definição: Um **buzzer** é um atuador que emite sons ao ser acionado por uma corrente elétrica. Pode ser usado para gerar alertas sonoros em diversos dispositivos. Um **buzzer** é um atuador eletromecânico ou eletrônico que gera sons ao ser excitado por uma corrente elétrica. Ele é amplamente utilizado em projetos de eletrônica e sistemas embarcados para fornecer feedback sonoro. Existem diferentes tipos de **buzzers**, cada um com suas características e aplicações específicas.

Tipos de Buzzer:

- **Buzzer passivo:** Precisa de uma frequência de entrada para gerar sons (usado em projetos de áudio mais complexos).
- **Buzzer ativo:** Gera som contínuo com corrente elétrica (mais simples, ideal para alertas sonoros).

Aplicações: Alarmes, timers, indicações sonoras em sistemas embarcados.

1.1 Buzzer Ativo

Definição: O buzzer ativo já possui um oscilador interno embutido. Isso significa que ele pode gerar sons de maneira independente, sem precisar de um sinal de frequência externo.

Características Técnicas:

- **Oscilador Interno:** O buzzer ativo tem um circuito oscilador embutido, o que simplifica o seu uso. Ele começa a gerar som automaticamente quando uma tensão é aplicada.
- **Sinal de Entrada:** Apenas precisa de um sinal **ON/OFF**. Ou seja, você só precisa ligar e desligar a alimentação (tensão) para ele gerar som.
- **Frequência Fixa:** A frequência do som que ele gera é fixa e determinada pelo circuito interno. Isso significa que você não pode controlar o tom ou a frequência do som diretamente.
- **Volume:** Normalmente, tem um volume sonoro entre 85dB e 100dB, dependendo da tensão aplicada.
- **Simples de Usar:** O buzzer ativo é ideal para aplicações que requerem alertas ou alarmes sonoros sem necessidade de controle de tom.

Funcionamento:

- Para o buzzer ativo funcionar, basta aplicar uma tensão entre seus dois terminais (positivo e negativo). O som é gerado automaticamente enquanto a tensão estiver presente.
- Exemplo: Um buzzer ativo conectado a um Arduino vai emitir som apenas com um comando de **digitalWrite(buzzerPin, HIGH)** e parar com **digitalWrite(buzzerPin, LOW)**.

Aplicações Comuns:

- **Alarmes Simples:** Alarmes de incêndio, alarme de intrusão, sensores de proximidade.
- **Notificações Sonoras:** Usado em dispositivos como micro-ondas ou temporizadores simples.
- **Sistemas de Alerta:** Pode ser usado para alertar sobre falhas ou eventos em sistemas eletrônicos.

1.2 Buzzer Passivo

Definição:

O buzzer passivo é um atuador que não possui um oscilador interno, ou seja, ele precisa de um sinal de frequência externo para produzir som.

Características Técnicas:

- **Sem Oscilador Interno:** Como ele não tem um oscilador interno, o buzzer passivo precisa receber um sinal de frequência do microcontrolador ou outro circuito externo para gerar som.
- **Controle de Frequência:** O buzzer passivo permite ao usuário controlar o tom do som gerado ao ajustar a frequência do sinal de entrada. Com isso, é possível tocar notas musicais ou diferentes sons.
- **Sinal PWM (Pulse Width Modulation):** Para controlar o buzzer passivo, geralmente usa-se um sinal PWM, que modula a frequência que será convertida em som.

- **Maior Flexibilidade:** Permite gerar sons personalizados e pode ser usado em aplicações mais avançadas, como tocar melodias.
- **Volume:** O volume pode variar dependendo da frequência aplicada e da intensidade do sinal, mas tende a ser mais baixo do que o de um buzzer ativo com mesma tensão de operação.

Funcionamento:

- Para o buzzer passivo funcionar, você deve enviar um sinal de frequência ao pino ao qual ele está conectado. Esse sinal determina a frequência de oscilação do diafragma interno, produzindo sons de diferentes tons.
- No Arduino, isso é feito usando a função **tone()**, que envia sinais de frequência diferentes para o pino conectado ao buzzer.
- Exemplo: Para tocar uma nota de 1000 Hz, usa-se **tone(buzzerPin, 1000);**.

Aplicações Comuns:

- **Sons Personalizados:** Aplicações que exigem diferentes sons ou melodias, como brinquedos eletrônicos, jogos ou sistemas de aviso com alertas customizados.
- **Sistemas Musicais:** Projetos de música com Arduino, onde várias notas podem ser tocadas.
- **Projetos de Robótica e STEM:** Pode ser usado em projetos educacionais para ensinar conceitos de som e frequências.

1.3 Diferenças Chave

Característica	Buzzer Ativo	Buzzer Passivo
Oscilador Interno	Sim	Não
Controle de Frequência	Não, som com frequência fixa	Sim, o som varia conforme a frequência aplicada
Sinal de Entrada	Simples, apenas ON/OFF	Requer um sinal de frequência externo
Complexidade	Muito simples de usar	Requer programação para gerar diferentes sons
Volume	Geralmente mais alto (85-100dB)	Pode ser mais baixo dependendo da frequência
Uso Ideal	Alarmes simples, alertas	Tocar músicas, sons personalizados
Função de Controle (Arduino)	**digitalWrite()** ou **delay()**	**tone()** ou **noTone()** para controlar frequência

---

1.4 Quando Usar Cada Um

Use um Buzzer Ativo quando:

- Você quer gerar um alerta sonoro simples.
- Não há necessidade de controlar o tom ou a frequência do som.
- Você precisa de uma solução rápida e simples para sons de aviso.

Use um Buzzer Passivo quando:

- Você deseja controlar o tom ou a frequência do som (como tocar uma música).
- O projeto requer sons variáveis ou tons específicos.
- Você quer explorar mais o controle de sinais de frequência.

Essas diferenças tornam o **buzzer ativo** ideal para aplicações simples e rápidas, enquanto o **buzzer passivo** oferece mais flexibilidade para projetos que envolvem sons personalizados ou controle de frequências sonoras.

2. Materiais Necessários:

1) 1 Placa Arduino (Uno, Nano, ou similar)
2) 1 Buzzer (preferencialmente ativo para esta aula)
3) 1 Resistor de 220Ω
4) Fios jumpers

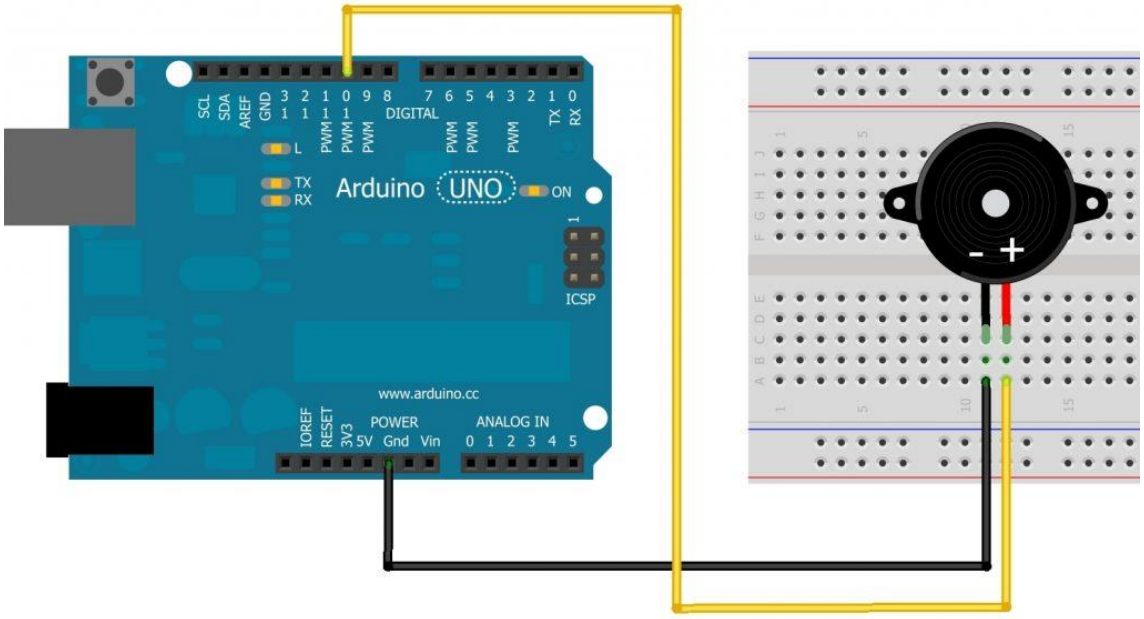
5) 1 Protoboard
6) 1 LED (opcional para indicadores visuais)

2.1 Montagem do Circuito (15 minutos)

Passos:

1. Conecte o pino positivo do Buzzer a um dos pinos digitais do Arduino (exemplo: pino 9).
2. Conecte o pino negativo do Buzzer ao GND (terra) do Arduino.
3. Opcionalmente, adicione um LED com resistor de 220Ω no circuito, ligado a outro pino digital (exemplo: pino 8) para indicar visualmente quando o som é emitido.
4. Conecte o Arduino ao computador via cabo USB.

3. Diagrama do Circuito:



4. Codificação

Explicação: Vamos programar o Arduino para emitir sons com diferentes durações, utilizando o Buzzer.

Código:

```
// Definindo os pinos
int buzzerPin = 10;

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

void loop() {
  // Emitir som por 1 segundo
  digitalWrite(buzzerPin, HIGH);
  delay(1000); // 1000 milissegundos

  // Parar o som por 1 segundo
  digitalWrite(buzzerPin, LOW);
  delay(1000);
}
```

Explicação do Código:

- O **pinMode** configura o pino do Buzzer como saída.
- A função **digitalWrite** envia um sinal **HIGH** para ligar o Buzzer e **LOW** para desligar.
- A função **delay** introduz pausas no código, controlando a duração do som e dos intervalos de silêncio.

5. Desafios Práticos (15 minutos)

- **Desafio 1:** Alterar o código para que o Buzzer emita um som mais curto, por exemplo, com 500 milissegundos.
- **Desafio 2:** Fazer o Buzzer tocar por períodos variáveis, como 200 ms e 800 ms, alternadamente.
- **Desafio 3:** Integrar um botão que, ao ser pressionado, faz o Buzzer emitir som apenas enquanto o botão estiver pressionado.

6. Expansão: Melodia simples (10 minutos)

- **Introdução:** Além de sons simples, é possível gerar frequências para criar notas musicais.
- **Exemplo de Código para tocar uma melodia:**

```
// Definindo o pino do buzzer
int buzzerPin = 9;

// Definindo as frequências das notas
int melody[] = {262, 294, 330, 349, 392, 440, 494, 523};

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

void loop() {
  // Loop para tocar a melodia
  for (int i = 0; i < 8; i++) {
    tone(buzzerPin, melody[i], 500); // Toca a nota por 500 ms
    delay(500); // Espera 500 ms antes da próxima nota
  }
}
```

Explicação: A função **tone()** permite enviar uma frequência específica para o Buzzer, gerando uma nota musical.

7. Encerramento e Discussão (10 minutos)

- **Discussão:** Como os atuadores sonoros podem ser aplicados em projetos do mundo real, como alarmes, notificações sonoras e dispositivos interativos.
- **Dúvidas:** Revisar qualquer dúvida que os alunos possam ter em relação à montagem e codificação do projeto.

8. Tarefas para Casa:

Tarefa 1: Ajuste de Intervalos de Som Modifique o código para que o buzzer emita sons com intervalos de tempo variáveis e crescentes. Por exemplo, comece emitindo um som por 200 milissegundos, depois por 400 ms, 600 ms, e assim por diante, até 1000 ms. Após atingir 1000 ms, o intervalo de som deve voltar a 200 ms e reiniciar o ciclo.

Tarefa 2: Alerta de Emergência Crie um sistema de alerta que faça o buzzer emitir três beeps rápidos (100 ms cada), seguidos de uma pausa de 500 ms. Repita esse ciclo indefinidamente, simulando um alarme de emergência.

Tarefa 3: Buzzer com Controle de Potenciômetro Adicione um potenciômetro ao seu circuito e modifique o código para controlar a frequência do buzzer. Conforme o usuário gira o potenciômetro, o buzzer deve emitir sons com frequências variáveis (mais agudo ou mais grave).

Tarefa 4: Alarme com Sensor de Luminosidade Adicione um sensor de luz (LDR) ao seu circuito. Programe o Arduino para fazer o buzzer emitir um som quando o nível de luz ambiente cair abaixo de um valor específico (indicando escuridão ou baixa luminosidade). Use o monitor serial para verificar o valor da leitura do sensor.

Tarefa 5: Melodia com Botão Crie uma melodia simples com o buzzer usando a função **tone()** e um botão. Cada vez que o botão for pressionado, a melodia deve tocar uma vez.

Tarefa 6: Gerar Alerta Sonoro Baseado em Temperatura Use um sensor de temperatura (como o DHT11 ou LM35) para acionar o buzzer. O buzzer deve emitir um alerta quando a temperatura ambiente ultrapassar um determinado limite (por exemplo, 30°C).

Tarefa 7: Sistema de Senha Sonora Programe um sistema de senha sonora: use dois ou mais botões para controlar o buzzer, onde cada botão representa uma "nota" diferente. Programe uma sequência de botões (notas) que devem ser pressionados na ordem correta para desbloquear o sistema e desligar o buzzer.

Tarefa 8: Jogo de Reflexos com Buzzer Crie um jogo de reflexos simples. O buzzer emite um som por um período aleatório entre 1 e 5 segundos. Quando o som parar, o jogador deve apertar um botão o mais rápido possível. O Arduino deve medir o tempo de reação do jogador e exibir esse tempo no monitor serial.

Tarefa 9: Simulador de Sirene de Emergência Programe o Arduino para simular o som de uma sirene de emergência com o buzzer. O som deve subir e descer de tom continuamente, como uma sirene de ambulância ou carro de polícia. Utilize a função **tone()** para variar a frequência de forma gradual e contínua.

Tarefa 10: Alarme de Proximidade com Sensor Ultrassônico Utilize um sensor ultrassônico (como o HC-SR04) para detectar a proximidade de objetos. Programe o Arduino para fazer o buzzer emitir um som intermitente, que aumenta de frequência conforme o objeto se aproxima do sensor. Quanto mais próximo o objeto, mais rápido o buzzer deve "bipar".

9. Respostas para os três desafios

Desafio 1: Alterar o código para que o Buzzer emita um som mais curto, por exemplo, 500 milissegundos

Neste desafio, o som deve durar 500 milissegundos. Para isso, basta modificar o valor no **delay()** para 500 milissegundos.

Código:

```
int buzzerPin = 9; // Definir o pino do buzzer
void setup() {
  pinMode(buzzerPin, OUTPUT); // Configurar o pino como saída
}
void loop() {
  digitalWrite(buzzerPin, HIGH); // Ligar o buzzer
  delay(500); // Emitir som por 500 milissegundos
  digitalWrite(buzzerPin, LOW);  // Desligar o buzzer
  delay(500); // Pausar por 500 milissegundos
}
```

Desafio 2: Fazer o Buzzer tocar por períodos variáveis, como 200 ms e 800 ms, alternadamente

Aqui, o buzzer deve tocar por dois períodos diferentes: 200 milissegundos e 800 milissegundos, alternadamente. Você pode fazer isso modificando os valores de **delay()** após os comandos de ligar e desligar o buzzer.

```
int buzzerPin = 9; // Definir o pino do buzzer
void setup() {
  pinMode(buzzerPin, OUTPUT); // Configurar o pino como saída
}
void loop() {
  // Emitir som por 200 milissegundos
  digitalWrite(buzzerPin, HIGH);
  delay(200);
  digitalWrite(buzzerPin, LOW);
  delay(200); // Pausar por 200 milissegundos
  // Emitir som por 800 milissegundos
  digitalWrite(buzzerPin, HIGH);
  delay(800);
  digitalWrite(buzzerPin, LOW);
  delay(800); // Pausar por 800 milissegundos
}
```

Desafio 3: Integrar um botão que, ao ser pressionado, faz o Buzzer emitir som apenas enquanto o botão estiver pressionado

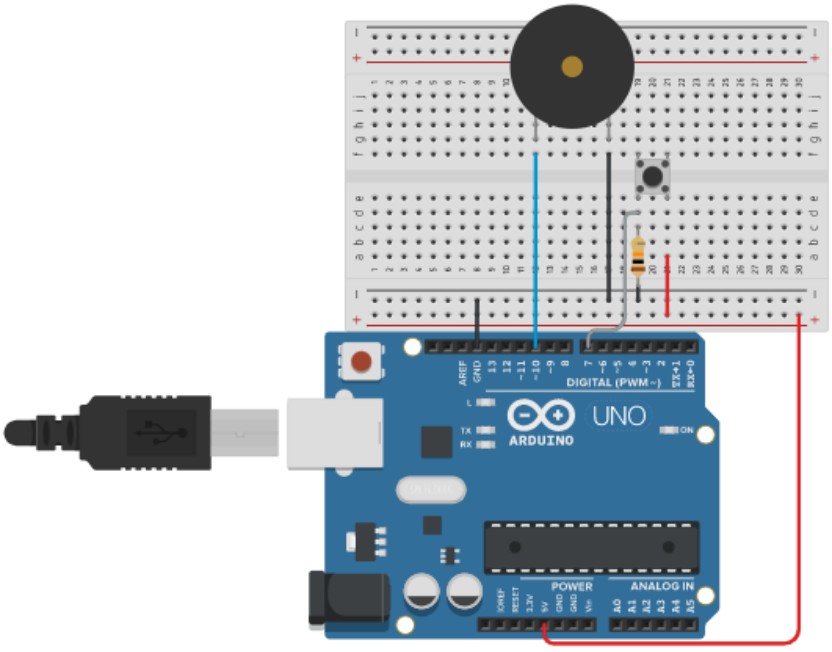
Neste caso, vamos usar um botão para controlar quando o buzzer emitirá som. O buzzer deve tocar apenas enquanto o botão estiver pressionado.

Componentes Adicionais:

1) 1 botão push-button.
2) 1 resistor de pull-down de 10kΩ (para garantir que o botão funcione corretamente).

Montagem:

1) Conecte um terminal do botão ao pino digital (ex: pino 2) do Arduino.
2) Conecte o outro terminal do botão ao GND (terra).
3) Adicione o resistor de 10kΩ entre o pino do botão e o GND para funcionar como pull-down.



Código:

```
int buzzerPin = 10; // Definir o pino do buzzer
int buttonPin = 7; // Definir o pino do botão
int buttonState = 0; // Variável para armazenar o estado do botão

void setup() {
  pinMode(buzzerPin, OUTPUT); // Configurar o pino do buzzer como saída
  pinMode(buttonPin, INPUT);  // Configurar o pino do botão como entrada
}

void loop() {
  buttonState = digitalRead(buttonPin); // Ler o estado do botão

  if (buttonState == HIGH) {
    // Se o botão estiver pressionado, ligar o buzzer
    digitalWrite(buzzerPin, HIGH);
  } else {
    // Se o botão não estiver pressionado, desligar o buzzer
    digitalWrite(buzzerPin, LOW);
  }
}
```

Explicação:

- **buttonPin** é o pino ao qual o botão está conectado.
- O **digitalRead()** lê o estado do botão. Se o botão estiver pressionado (**HIGH**), o buzzer toca. Caso contrário, ele permanece desligado.
- **Resistor Pull-Down:** Mantém o pino em estado **LOW** quando o botão não está pressionado, evitando ruído ou leituras incorretas.

10. Respostas das tarefas para casa

Soluções 1:

Tarefa 1: Ajuste de Intervalos de Som

O buzzer emitirá som com intervalos de tempo crescentes, de 200 ms a 1000 ms, e depois reinicia o ciclo.

```
int buzzerPin = 9;
int delayTime = 200;
void setup() {
  pinMode(buzzerPin, OUTPUT);
}
void loop() {
  digitalWrite(buzzerPin, HIGH);
  delay(delayTime); // Toca som
  digitalWrite(buzzerPin, LOW);
  delay(delayTime); // Pausa
  // Aumenta o intervalo até 1000 ms e depois volta a 200 ms
  delayTime += 200;
  if (delayTime > 1000) {
    delayTime = 200;
  }
}
```

Tarefa 2: Alerta de Emergência

O buzzer emitirá três beeps rápidos, seguidos de uma pausa de 500 ms.

```
int buzzerPin = 9;
void setup() {
  pinMode(buzzerPin, OUTPUT);
}
void loop() {
  for (int i = 0; i < 3; i++) {
    digitalWrite(buzzerPin, HIGH);
    delay(100); // Beep curto de 100 ms
    digitalWrite(buzzerPin, LOW);
    delay(100); // Pausa curta entre beeps
  }
  delay(500); // Pausa maior antes de repetir o ciclo
}
```

Tarefa 3: Buzzer com Controle de Potenciômetro

O potenciômetro controla a frequência do buzzer.

```
int buzzerPin = 9;
int potPin = A0; // Pino do potenciômetro
void setup() {
  pinMode(buzzerPin, OUTPUT);
}
void loop() {
  int potValue = analogRead(potPin); // Lê o valor do potenciômetro (0 a 1023)
  int frequency = map(potValue, 0, 1023, 100, 2000); // Mapeia para uma frequência
entre 100 Hz e 2000 Hz
  tone(buzzerPin, frequency); // Toca som com a frequência definida
  delay(100); // Pequena pausa para estabilidade
}
```

Tarefa 4: Alarme com Sensor de Luminosidade

O buzzer será acionado quando a luminosidade ficar abaixo de um determinado nível.

```
int buzzerPin = 9;
int ldrPin = A0; // Pino do LDR
int threshold = 300; // Definir o nível de luz para acionar o alarme

void setup() {
  pinMode(buzzerPin, OUTPUT);
```

```
  Serial.begin(9600); // Para monitorar a leitura do LDR
}

void loop() {
  int ldrValue = analogRead(ldrPin); // Lê o valor do sensor de luz
  Serial.println(ldrValue); // Mostra no monitor serial
  if (ldrValue < threshold) {
    digitalWrite(buzzerPin, HIGH); // Aciona o buzzer se estiver escuro
  } else {
    digitalWrite(buzzerPin, LOW); // Desliga o buzzer se estiver claro
  }
  delay(500);
}
```

Tarefa 5: Melodia com Botão

Ao pressionar o botão, uma melodia simples toca no buzzer.

```
int buzzerPin = 9;
int buttonPin = 2;
int buttonState = 0;
int melody[] = {262, 294, 330, 349, 392}; // Notas da melodia
int noteDurations[] = {500, 500, 500, 500, 500}; // Duração de cada nota
void setup() {
  pinMode(buzzerPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    // Toca a melodia quando o botão for pressionado
    for (int i = 0; i < 5; i++) {
      tone(buzzerPin, melody[i], noteDurations[i]);
      delay(noteDurations[i] + 100); // Pausa entre as notas
    }
  }
}
```

Tarefa 6: Gerar Alerta Sonoro Baseado em Temperatura

O buzzer soa quando a temperatura ultrapassa um limite.

```
#include <DHT.h>
#define DHTPIN 2 // Pino de dados do sensor
#define DHTTYPE DHT11 // Tipo do sensor
DHT dht(DHTPIN, DHTTYPE);
int buzzerPin = 9;
float temperatureThreshold = 30.0; // Limite de temperatura
void setup() {
  pinMode(buzzerPin, OUTPUT);
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  float temperature = dht.readTemperature(); // Lê a temperatura
  Serial.println(temperature);
  if (temperature > temperatureThreshold) {
    digitalWrite(buzzerPin, HIGH); // Aciona o buzzer se estiver quente demais
  } else {
    digitalWrite(buzzerPin, LOW); // Desliga o buzzer se a temperatura estiver ok
  }
  delay(2000); // Espera antes da próxima leitura
}
```

Tarefa 7: Sistema de Senha Sonora

Dois botões são usados para reproduzir uma senha sonora. Quando a senha correta é pressionada, o sistema "desbloqueia".

```
int buzzerPin = 9;
int buttonPin1 = 2;
int buttonPin2 = 3;
```

```
int password[] = {1, 2, 1}; // Senha de exemplo: botão 1, botão 2, botão 1
int userInput[3]; // Armazena o input do usuário
int inputIndex = 0;
void setup() {
  pinMode(buzzerPin, OUTPUT);
  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);
}
void loop() {
  if (digitalRead(buttonPin1) == HIGH) {
    userInput[inputIndex] = 1;
    inputIndex++;
    tone(buzzerPin, 440, 200); // Toca som associado ao botão 1
  } else if (digitalRead(buttonPin2) == HIGH) {
    userInput[inputIndex] = 2;
    inputIndex++;
    tone(buzzerPin, 660, 200); // Toca som associado ao botão 2
  }

  if (inputIndex == 3) { // Verifica se a senha está completa
    bool correctPassword = true;
    for (int i = 0; i < 3; i++) {
      if (userInput[i] != password[i]) {
        correctPassword = false;
        break;
      }
    }
    if (correctPassword) {
      tone(buzzerPin, 880, 1000); // Senha correta: som longo
    } else {
      tone(buzzerPin, 200, 1000); // Senha errada: som grave
    }
    inputIndex = 0; // Reinicia o input do usuário
  }
}
```

Tarefa 8: Jogo de Reflexos com Buzzer

O buzzer toca após um intervalo aleatório, e o jogador deve apertar um botão o mais rápido possível.

```
int buzzerPin = 9;
int buttonPin = 2;
unsigned long reactionStartTime, reactionEndTime;
void setup() {
  pinMode(buzzerPin, OUTPUT);
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  // Gera um atraso aleatório entre 1 e 5 segundos
  int delayTime = random(1000, 5000);
  delay(delayTime);
    // Toca o som do buzzer e inicia a contagem do tempo de reação
  tone(buzzerPin, 1000);
  reactionStartTime = millis();
  // Espera até que o jogador pressione o botão
  while (digitalRead(buttonPin) == LOW);
  // Para o som e calcula o tempo de reação
  noTone(buzzerPin);
  reactionEndTime = millis();
  unsigned long reactionTime = reactionEndTime - reactionStartTime;
  // Mostra o tempo de reação no monitor serial
  Serial.print("Tempo de reacao: ");
  Serial.print(reactionTime);
  Serial.println(" ms");
  delay(2000); // Pausa antes da próxima rodada
}
```

Tarefa 9: Simulador de Sirene de Emergência

A frequência do buzzer alterna entre alta e baixa, simulando uma sirene.

```
int buzzerPin = 9;
```

```
void setup() {
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  // Frequência subindo (som de sirene)
  for (int freq = 500; freq <= 1000; freq += 10) {
    tone(buzzerPin, freq);
    delay(10);
  }
  // Frequência descendo
  for (int freq = 1000; freq >= 500; freq -= 10) {
    tone(buzzerPin, freq);
    delay(10);
}
```

Tarefa 10: Alarme de Proximidade com Sensor Ultrassônico

O buzzer soa com mais frequência à medida que um objeto se aproxima do sensor ultrassônico.

```
int buzzerPin = 9;
int trigPin = 10; // Pino de disparo do sensor
int echoPin = 11; // Pino de eco do sensor
long duration;
int distance;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(buzzerPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  // Envia o pulso ultrassônico
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Calcula o tempo de resposta
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2; // Converte o tempo em distância (cm)

  Serial.print("Distancia: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Quanto mais próximo o objeto, mais rápido o som
  if (distance < 50) {
    int beepDelay = map(distance, 0, 50, 100, 1000); // Mapeia a distância para
um intervalo de tempo de beep
    tone(buzzerPin, 1000); // Emite som
    delay(beepDelay);
    noTone(buzzerPin); // Pausa o som
    delay(beepDelay);
  } else {
    noTone(buzzerPin); // Se estiver longe, o buzzer fica desligado
  }
  delay(200);
}
```

Outras soluções possíves

Tarefa 1: Ajuste de Intervalos de Som

Descrição:

Modificar o código para que o buzzer emita sons com intervalos de tempo variáveis e crescentes: 200 ms, 400 ms, 600 ms, até 1000 ms, e depois reiniciar o ciclo.

Componentes Necessários:

- 1 Placa Arduino (Uno, Nano, etc.)
- 1 Buzzer (ativo ou passivo)
- Fios jumpers
- 1 Protoboard

Montagem do Circuito:

- Conecte o pino positivo do buzzer ao pino digital 9 do Arduino.
- Conecte o pino negativo do buzzer ao GND do Arduino.

Código:

```
// Definição do pino do buzzer
const int buzzerPin = 9;
// Array com os intervalos de tempo em milissegundos
const int intervals[] = {200, 400, 600, 800, 1000};
const int numIntervals = sizeof(intervals) / sizeof(intervals[0]);

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

void loop() {
  for (int i = 0; i < numIntervals; i++) {
    digitalWrite(buzzerPin, HIGH);  // Liga o buzzer
    delay(intervals[i]);            // Mantém o som por interval[i] ms
    digitalWrite(buzzerPin, LOW);   // Desliga o buzzer
    delay(intervals[i]);            // Pausa por interval[i] ms
  }
}
```

Explicação:

- **Array intervals[]:** Contém os diferentes tempos de ativação e pausa do buzzer.
- **Loop for:** Itera sobre cada intervalo, ligando e desligando o buzzer com os tempos definidos.
- **Função delay():** Controla a duração do som e das pausas.

Tarefa 2: Alerta de Emergência

Descrição:

Criar um sistema de alerta que faça o buzzer emitir três beeps rápidos (100 ms cada), seguidos de uma pausa de 500 ms, repetidamente.

Componentes Necessários:

- 1 Placa Arduino
- 1 Buzzer
- Fios jumpers
- 1 Protoboard

Montagem do Circuito:

- Conecte o pino positivo do buzzer ao pino digital 9 do Arduino.
- Conecte o pino negativo do buzzer ao GND do Arduino.

Código:

```
const int buzzerPin = 9;
void setup() {
  pinMode(buzzerPin, OUTPUT); // Configura o pino do buzzer como saída
}

void loop() {
  // Emitir três beeps rápidos
  for (int i = 0; i < 3; i++) {
    digitalWrite(buzzerPin, HIGH); // Liga o buzzer
    delay(100);                    // Mantém o som por 100 ms
    digitalWrite(buzzerPin, LOW);  // Desliga o buzzer
    delay(100);                    // Pausa de 100 ms entre beeps
  }
```

```
  delay(500); // Pausa de 500 ms após os três beeps
}
```

Explicação:

- **Loop Interno for:** Gera três beeps rápidos com 100 ms de duração cada.
- **Delay de 500 ms:** Pausa após os três beeps antes de repetir o ciclo.

Tarefa 3: Buzzer com Controle de Potenciômetro

Descrição:

Adicionar um potenciômetro ao circuito e modificar o código para controlar a frequência do buzzer conforme a posição do potenciômetro.

Componentes Necessários:

1) 1 Placa Arduino
2) 1 Buzzer (preferencialmente passivo)
3) 1 Potenciômetro (10kΩ)
4) Fios jumpers
5) 1 Protoboard

Montagem do Circuito:

1. **Conexão do Potenciômetro:**

- **Terminal central (wiper):** Conecte ao pino A0 do Arduino.
- **Outros terminais:** Conecte um terminal ao 5V e o outro ao GND.

2. **Conexão do Buzzer:**

- **Pino positivo:** Conecte ao pino digital 9 do Arduino.
- **Pino negativo:** Conecte ao GND do Arduino.

Código:

```
const int buzzerPin = 9;    // Pino do buzzer
const int potPin = A0;      // Pino do potenciômetro

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

void loop() {
  int potValue = analogRead(potPin);           // Lê o valor do potenciômetro (0-
1023)
  int frequency = map(potValue, 0, 1023, 200, 2000); // Mapeia para uma frequência
de 200Hz a 2000Hz

  tone(buzzerPin, frequency); // Emite o tom correspondente
  delay(100);                 // Pequeno delay para estabilidade
}
```

Explicação:

- **Leitura do Potenciômetro:** Utiliza analogRead() para obter o valor analógico do potenciômetro.
- **Mapeamento da Frequência:** Converte o valor lido (0-1023) para uma frequência utilizável (200Hz a 2000Hz) usando map().
- **Função tone():** Emite um tom no buzzer com a frequência calculada.
- **Delay de 100 ms:** Pequena pausa para evitar leituras muito rápidas.

Tarefa 4: Alarme com Sensor de Luminosidade

Descrição:

Adicionar um sensor de luz (LDR) ao circuito e programar o Arduino para fazer o buzzer emitir um som quando o nível de luz ambiente cair abaixo de um valor específico.

Componentes Necessários:

1) 1 Placa Arduino
2) 1 Buzzer
3) 1 LDR (sensor de luminosidade)

4) 1 Resistor de 10kΩ
5) Fios jumpers
6) 1 Protoboard

Montagem do Circuito:

1. **Conexão do LDR:**
   - **Um terminal do LDR:** Conecte ao 5V.
   - **Outro terminal do LDR:** Conecte ao pino A1 do Arduino.
   - **Conecte o resistor de 10kΩ:** Entre o pino A1 e o GND (configuração de divisor de tensão).
2. **Conexão do Buzzer:**

- **Pino positivo:** Conecte ao pino digital 9 do Arduino.
- **Pino negativo:** Conecte ao GND do Arduino.

Código:

```
const int buzzerPin = 9;    // Pino do buzzer
const int ldrPin = A1;      // Pino do LDR
const int threshold = 300;  // Valor de limiar para detecção de baixa luminosidade

void setup() {
  pinMode(buzzerPin, OUTPUT); // Configura o pino do buzzer como saída
  Serial.begin(9600);          // Inicia a comunicação serial
}

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

  if (ldrValue < threshold) {
    digitalWrite(buzzerPin, HIGH);    // Liga o buzzer
  } else {
    digitalWrite(buzzerPin, LOW);     // Desliga o buzzer
  }

  delay(500); // Pausa de 500 ms antes da próxima leitura
}
```

Explicação:

- **Divisor de Tensão com LDR:** Permite que o Arduino leia variações na luminosidade.
- **Valor de Limiar (threshold):** Define o ponto em que o buzzer será ativado. Pode ser ajustado conforme necessário.
- **Comunicação Serial:** Exibe os valores lidos do LDR para monitoramento e ajuste.
- **Controle do Buzzer:** Liga o buzzer quando a luminosidade está abaixo do limiar definido.

Tarefa 5: Melodia com Botão

Descrição:

Criar uma melodia simples que é tocada quando um botão é pressionado, utilizando a função tone().

Componentes Necessários:

1) 1 Placa Arduino
2) 1 Buzzer passivo
3) 1 Botão push-button
4) 1 Resistor de 10kΩ (pull-down)
5) Fios jumpers
6) 1 Protoboard

Montagem do Circuito:

1. **Conexão do Botão:**

- **Um terminal do botão:** Conecte ao pino digital 2 do Arduino.
- **Outro terminal do botão:** Conecte ao GND.
- **Resistor de 10kΩ:** Entre o pino 2 e o GND (pull-down).

2. **Conexão do Buzzer:**

- **Pino positivo:** Conecte ao pino digital 9 do Arduino.
- **Pino negativo:** Conecte ao GND do Arduino.

Código:

```
const int buzzerPin = 9;      // Pino do buzzer
const int buttonPin = 2;      // Pino do botão
bool melodyPlaying = false;   // Estado da melodia
// Definição das notas da melodia
const int melody[] = {
  262, // C4
  294, // D4
  330, // E4
  349, // F4
  392, // G4
  440, // A4
  494, // B4
  523  // C5
};
const int noteDuration = 250; // Duração de cada nota em ms
const int numNotes = sizeof(melody) / sizeof(melody[0]);

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

void loop() {
  int buttonState = digitalRead(buttonPin); // Lê o estado do botão

  if (buttonState == HIGH && !melodyPlaying) {
    melodyPlaying = true;
    playMelody();
    melodyPlaying = false;
  }
}

void playMelody
.....;..... termine.
```

Invente as outras...