UNIVERSIDADE FEDERAL DO CEARÁ

Centro de Tecnologia

Departamento de Engenharia Mecânica

INSTRUMENTAÇÃO COM TINKERCAD - CAPÍTULO 5: SENSORES DE MOVIMENTO

Thiago Victor Albuquerque de Freitas - 494080

2021

Projeto Sistema de Alarme

Descrição:

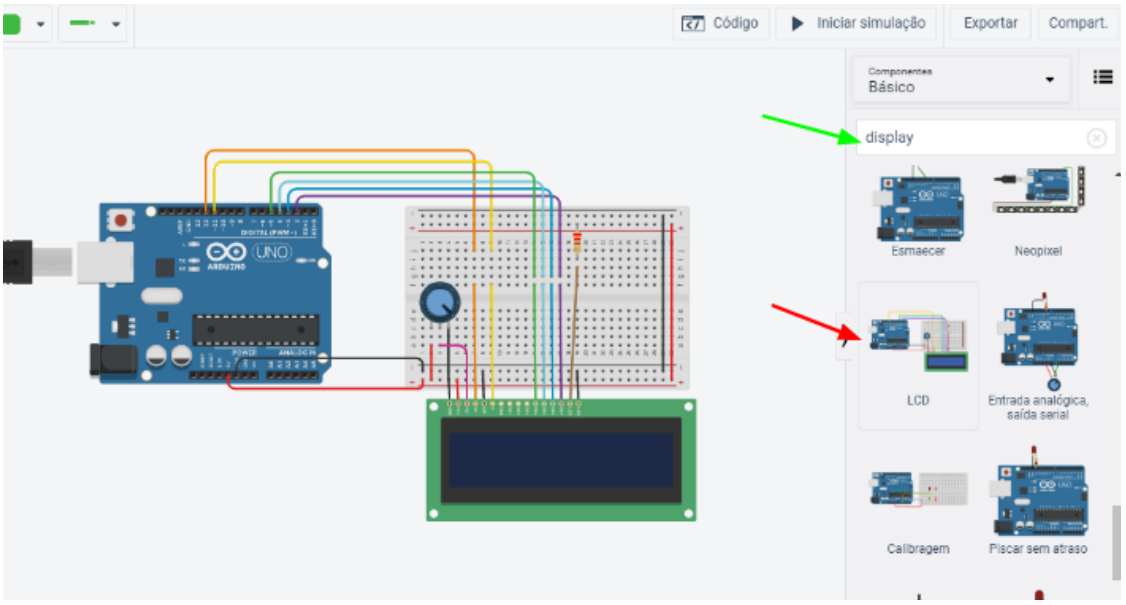
O seguinte circuito irá simular um sistema de alarme, o qual utilizará um buzzer para emitir sons caso o sensor PIR detecte algum movimento. Além disso, também será utilizado um Display LCD para sinalizar o estado do sensor de presença. Por fim, cabe ressaltar que o circuito foi baseado no vídeo Learn Top 5 Arduino Sensors Projects without any Hardware! Tinkercad do canal THE ELECTRONIC GUY.

Circuito:

- O circuito é composto:
  - 1 Arduino;
  - 1 Protoboard;
  - 1 Display LCD 16x2;
  - 1 Buzzer;
  - 1 Sensor PIR;
- Montagem do circuito

Etapa 1: Ao digitar display na busca dos componentes, irá aparecer o circuito com todas as ligações já feitas com o LCD 16x2, assim clique nele e arraste para a tela de circuitos, como mostrado na Fig. 1, o qual a seta verde sinaliza a opção de busca e, a seta vermelha, o circuito que deve ser escolhido.

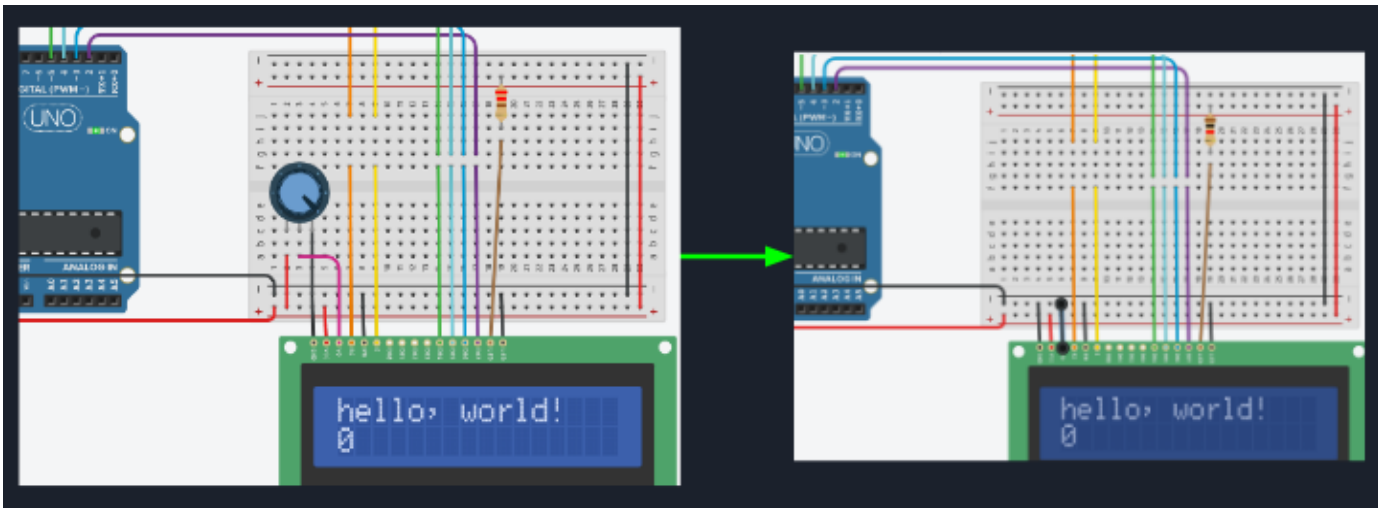
Figura 1 - Circuito do Display LCD16x2.



Fonte: Autor.

Etapa 2: Para simplificar o circuito, tire o potênciometro e conecte o GND na terceira porta do LCD (este é o lugar que estava conectado o fio rosa). Também é importante ressaltar que o resistor serve para definir o brilho no display (quanto maior a resistência, menor será o brilho). Esse processo pode ser visualizado na Fig. 2.

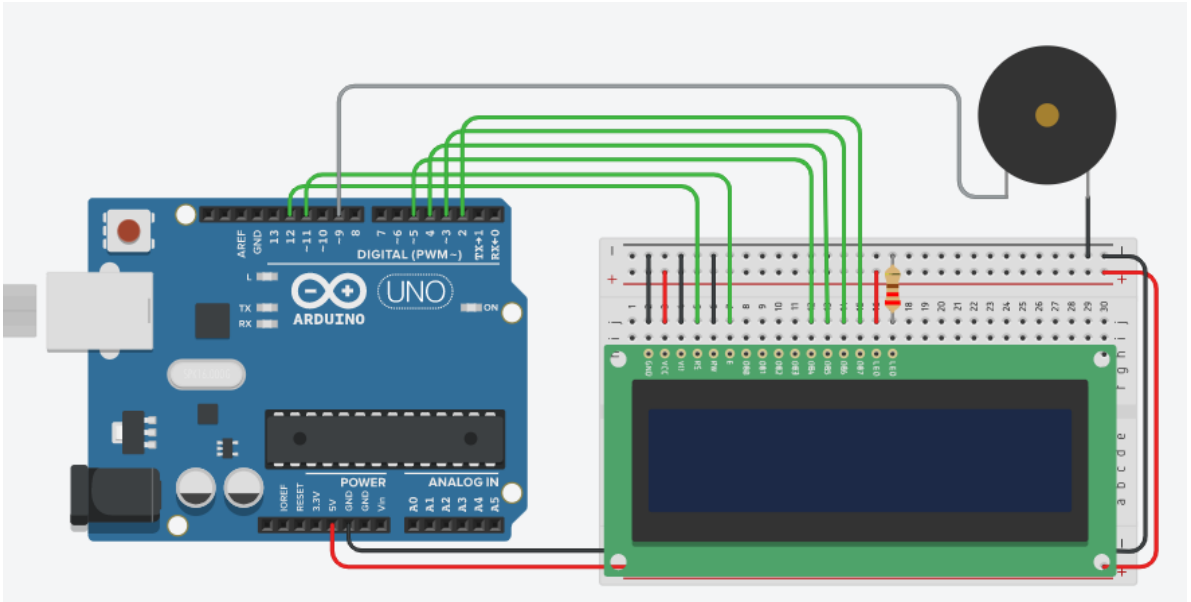
Figura 2 - Processo de remoção do potenciômetro e troca da resistência.



Fonte: Autor.

Etapa 3: Organize o circuito para ficar o mais simples possível, como o da Fig. 3, e inclua o buzzer com o pino positivo conectado à porta 9. Além disso, é interessante separar as cores dos fios que vão para o display e para o buzzer. Nesse caso, foi adotado a cor verde para os fios de sinal do display e a cor cinza para o buzzer.

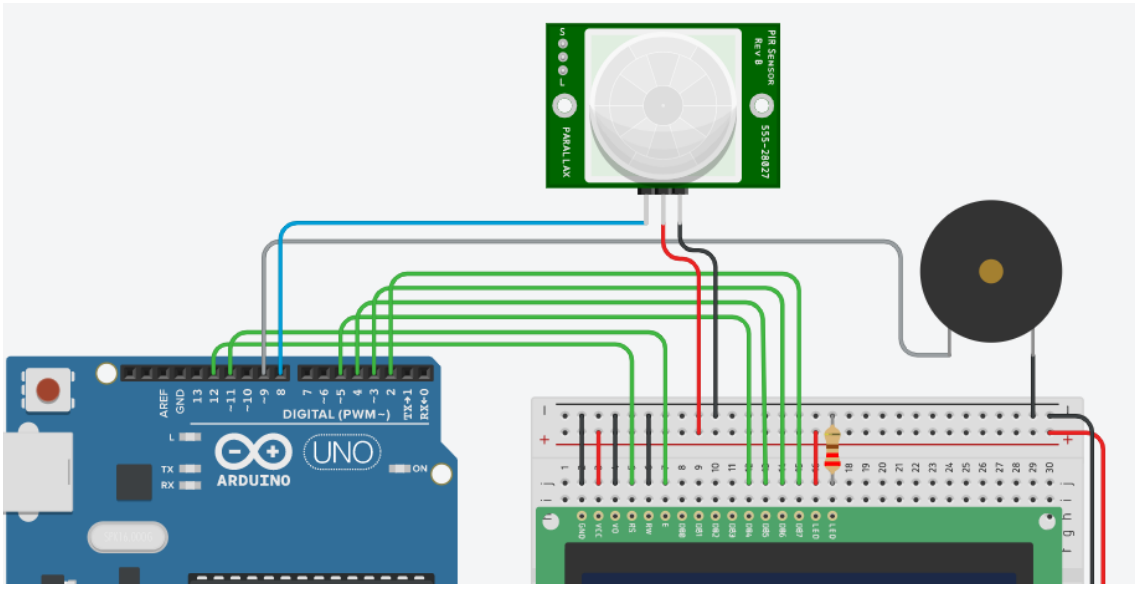
Figura 3 - Circuito com o display e o buzzer organizados.



Fonte: Autor.

Etapa 4: Adicione o sensor PIR com a porta mais à esquerda do sensor conectado à porta 8 do arduino, como é mostrado na Fig. 4.

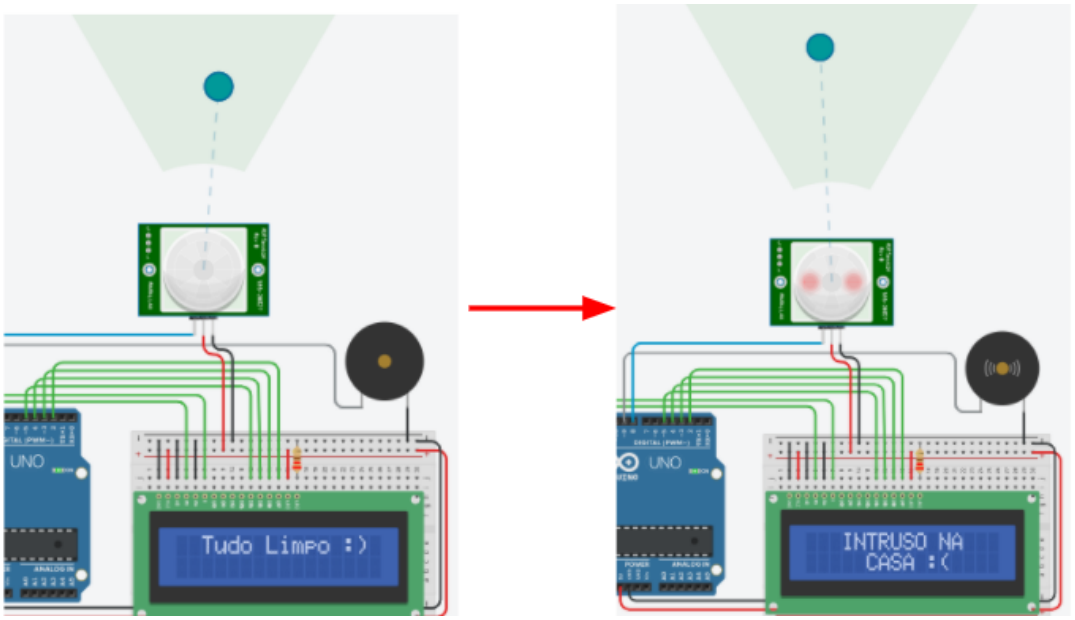
Figura 4 - Adição do sensor PIR ao circuito.



Fonte: Autor.

Por fim, é importante enfatizar que é possível gerar a variação de movimento ao mover o círculo azul após clicar no sensor PIR, como é mostrado na Fig. 5.

Figura 5 - Ativando sensor PIR com o movimento do círculo azul.



Fonte: Autor.

Programação:

A primeira parte da programação é bem simples, mas cabe atenção especial em definir as portas do sensor PIR, que é a 8, e a do buzzer, que é a 9, como é possível visualizar na Fig. 6. Além disso, define-se as configurações iniciais para a utilização do Display da mesma forma que foi mostrado no capítulo 2 sobre transdutores.

Figura 6 - Configurações iniciais do código do sistema de alarme.

```
//Autor: Thiago Victor A. de Freitas - Estudante de Engenharia Mecânica (UFC)
#include <LiquidCrystal.h>
#define PIR_Input 8 //Estabelece o pino 8 como a porta do sensor PIR
#define buzzer_Output 9 //Estabelece o pino 9 como a porta do buzzer
//Criação objeto lcd da classe LiquidCrystal
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
#define LINHAS 2
#define COLUNAS 16

void setup() {

  pinMode(buzzer_Output, OUTPUT);//Estabelece o aporta do buzzer como output
  pinMode(PIR_Input, INPUT);//Estabelece o aporta do PIR como input
  Serial.begin(9600);//Inicia a comunicação serial
  lcd.begin(COLUNAS, LINHAS);//16 Colunas e 2 linhas

}
```

Fonte: Autor.

Em relação ao loop, a mensagem “ Tudo Limpo :) “ é sempre mostrada, porém se o sensor PIR detectar algum movimento, o arduino ativa o buzzer aplicando uma frequência de 300 Hz e escreve ”INTRUSO NA CASA : (“. Após isso, o sensor espera 0,01 segundos para repetir tudo que está no loop. Essa parte do código é mostrada na Fig. 7.

Figura 7 - Loop do código do sistema de alarme.

```
void loop() {

    lcd.setCursor(1,0);//Posiciona o cursor na coluna 2 e linha 1
    lcd.print(" Tudo Limpo :)");//Escreve a mensagem no Display

//Checando se tem intruso
    boolean PIRvalor = digitalRead(PIR_Input);//Ler a porta do PIR
    if (PIRvalor == 1){
        tone(buzzer_Output,300);
/*Comando tone(porta,frequencia,tempo): Gera uma onda quadrada
na frequência especificada em um pino.*/
        lcd.clear();//Apaga oque está escrito no LCD
        lcd.setCursor(3,0);//Posiciona o cursor na coluna 4 e linha 1
        lcd.print("INTRUSO NA");//Escreve a mensagem no Display
        lcd.setCursor(5,1);//Posiciona o cursor na coluna 6 e linha 2
        lcd.print("CASA :(");//Escreve a mensagem no Display
        delay(1500);//Espera 1,5 segundos;
        noTone(buzzer_Output);
/*Comando noTone(porta): Encerra o som iniciado com o comando tone*/
        lcd.clear();//Apaga oque está escrito no LCD
    }
    delay(10);//Espera 0,01 segundos;

}
```

Fonte: Autor.

Projeto Sensor de Distância

Descrição:

O seguinte circuito irá medir a distância de um objetivo até um sensor ultrassônico e mostrar esse valor em um Display LCD.

Circuito:

- O circuito é composto:
  - 1 Display LCD 16x2;
  - 1 Protoboard;
  - 1 Resistor de 220 Ω;
  - 1 Resistor de 25 kΩ;
  - 1 Arduino;
  - 1 Sensor de distância ultrassônico HC-SR04;
- Montagem do circuito

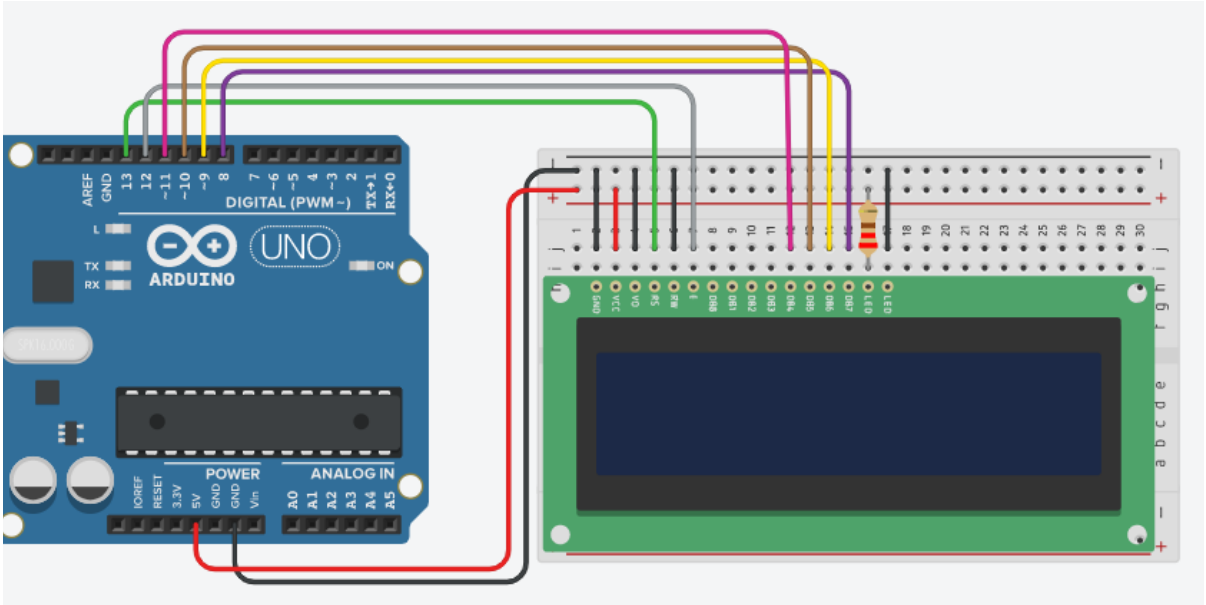
Etapa 1: Para diferenciar dos outros circuitos, iremos colocar o display do zero. Logo, conecte o Display em uma protoboard e ligue os fios no arduino seguindo a Tabela 1. Além disso, as ligações são mostradas na Fig. 8.

Tabela 1 - Sequência de ligações do arduino no Display LCD 16x2.

Pino no arduino	Pino no Display	Cor
GND	Solo	Vermelho
5V	Potência	Preto
GND	Contraste	Preto
13	Seleção de registro	Verde
GND	Leitura e gravação	Preto
12	Ativar	Cinza
11	DB4	Rosa
10	DB5	Marrom
9	DB6	Amarelo
8	DB7	Roxo
Resistor de 220 Ω ao 5V	Led ânodo	-
GND	Led catódico	Preto

Fonte: Autor.

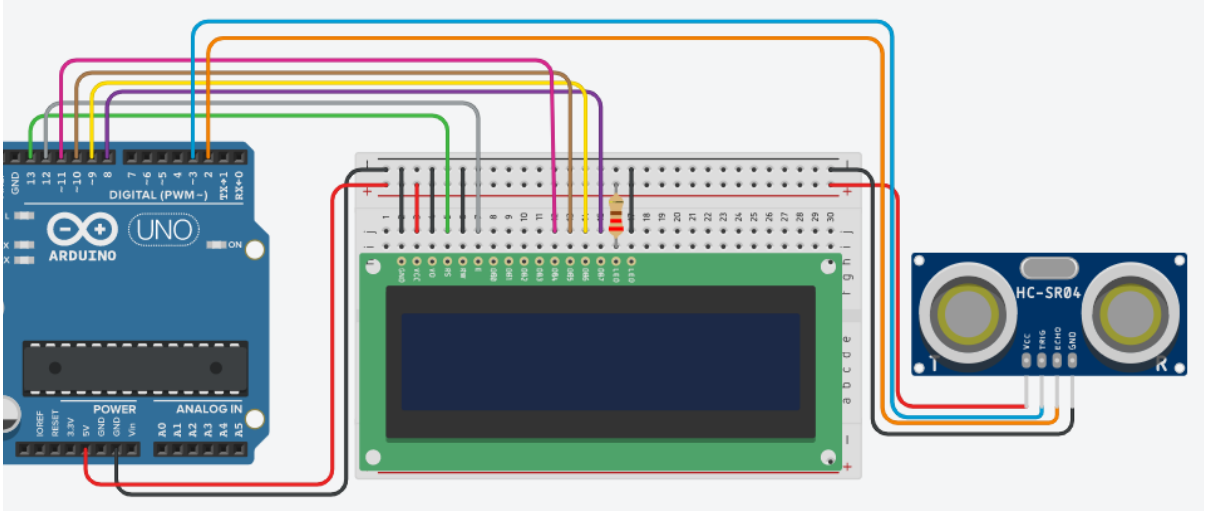
Figura 8 - Circuito do Display LCD16x2.



Fonte: Autor.

Etapa 2: Pesquise o sensor de distância ultrassônico HC-SR04, o qual tem 4 pinos, e inclua ao circuito ligando o pino "Acionador" na porta 3 do arduino e o pino “Eco” na porta 2. As ligações são mostradas na Fig. 9.

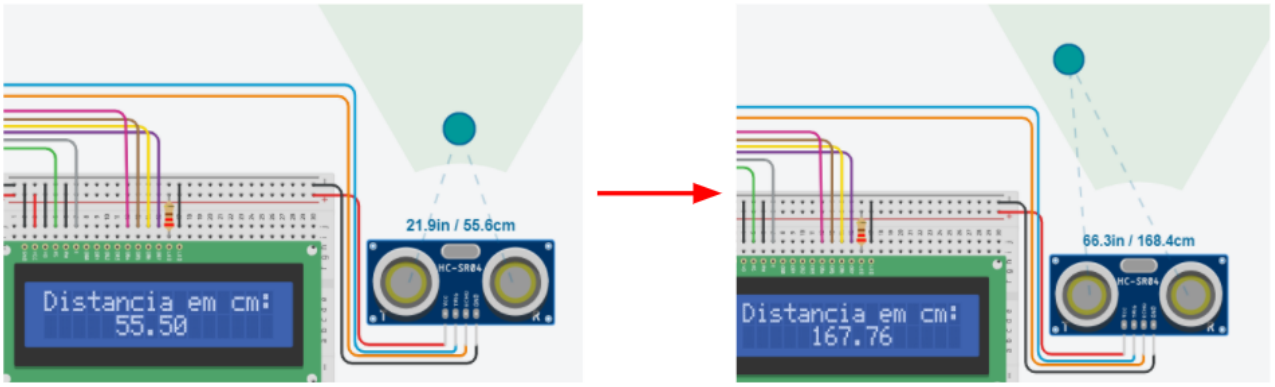
Figura 9 - Ligações do sensor ultrassônico ao Arduino.



Fonte: Autor.

Por fim, cabe ressaltar que é possível modificar a posição do objeto ao movimentar o círculo azul após clicar no sensor, como é mostrado na Fig. 10.

Figura 10 - Alterando a posição do objetivo na simulação.



Fonte: Autor.

Programação:

Conforme o site FilipeFlop, para iniciar a medição com o sensor, é necessário alimentar o módulo colocando o pino "Acionador" em estado HIGH, ou seja 5 V, durante 10 us. Após isso, o sensor irá emitir uma onda sonora que retornará ao sensor ao encontrar algum obstáculo. Durante esse tempo de emissão e recebimento do sinal, o pino “Eco” ficará em estado HIGH. Dessa forma, o cálculo da distância é feito de acordo com o tempo que o pino “Eco” ficou no nível HIGH, após o pino “Acionador” ter sido ligado.

Com base nisso, é possível calcular a distância entre o sensor e o objeto com a seguinte fórmula:

$$D = (V_{som} \, x \, T)/2$$

Sendo D a distância, V_som a velocidade do som, que pode ser considerada 340 m/s, e T o tempo de duração que a porta “Eco” ficou ativada.

Dessa forma, a Fig. 11 mostra a função tempo_Medido ( ) que faz o cálculo seguindo a lógica explicado acima. Além disso, também foi utilizado a função display_distância para mostrar a distância do objeto até o sensor no Display e no serial monitor do arduino.

Figura 11 - Configurações iniciais do código do sistema de alarme.

```
  void tempo_Medido(){
    //Função para medir o tempo que o pulso demora para voltar
    digitalWrite(adicionador, LOW);
    delayMicroseconds(2);//Espera 2 microsegundos;

    digitalWrite(adicionador, HIGH);
    delayMicroseconds(10);//Espera 10 microsegundos;
    digitalWrite(adicionador, LOW);

    duracao = pulseIn(eco, HIGH);
    /*Comando pulseIn(porta,estado):Captura a duração de um pulso em um pino.
    Por exemplo, se o valor HIGH é passado para a função, a função pulseIn()
    espera o pino ir para do estado 'LOW' para HIGH, começa a temporizar,
    então espera o pino ir para o estado LOW e para de temporizar */
  }

  void display_distancia(){
    //Função para mostrar no display e no serial monitor a distância percorrida
    lcd.clear();//Apaga oque está escrito no LCD
    lcd.setCursor(0, 0);//Posiciona o cursor na coluna 1 e linha 1
    lcd.print("Distancia em cm: ");//Escreve a mensagem no Display
    lcd.setCursor(5, 1);//Posiciona o cursor na coluna 6 e linha 2
    lcd.print(distancia);//Escreve a mensagem no Display
    Serial.print("Distancia em cm: ");//Escreve a mensagem no Display
    Serial.print(distancia);//Escreve a mensagem no serial monitor
    Serial.println();//Pula a linha no serial monitor
    delay(1000);//Espera 1 segundos;
   }
```

Fonte: Autor.

Para finalizar, a Fig. 12 mostra a estrutura completa do código, o qual, nas primeiras linhas, são feitas as configurações iniciais e o loop usa as funções criadas para processar e mostrar a distância medida em centímetros.

Figura 12 - Configurações iniciais do código do sistema de alarme.

```
//Autor: Thiago Victor A. de Freitas - Estudante de Engenharia Mecânica (UFC)

#include <LiquidCrystal.h>
//Criação objeto lcd da classe LiquidCrystal
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);//lcd(RS,EN,D4,D5,D6,D7)

#define eco 2 //Estabelece o pino 2 como a porta do eco do sensor
#define adicionador 3 //Estabelece o pino 4 como a porta do adicionador do sensor
float  duracao; /*Variável irá armezar o tempo de duração que a onda
leva para ir até o objeto e voltar*/
float  distancia; //Variável para armazenar a distância

void setup() {
    pinMode(adicionador, OUTPUT);//Estabelece a porta como OUTPUT
    pinMode(eco, INPUT);//Estabelece a porta como INPUT
    Serial.begin(9600);//Inicia a comunicação serial
    lcd.begin(16, 2);//Inicia o lcd
 }

void loop() {
    tempo_Medido();
    distancia = duracao * (0.0343) / 2;
    //Calcula a distancia percorrida pelo pulso em cm
    display_distancia();
    //Mostra no display e no serial monitor a distância percorrida
}
```

Fonte: Autor.

Projeto Sensor de Inclinação

Descrição:

O sensor SW-200D é ideal para medir a inclinação de um sistema sem precisar de alta precisão e complexidade como o giroscópio. Com base nisso, este projeto irá utilizar esse sensor para indicar se o sistema está horizontalmente ou não em relação a um plano. Por fim, será utilizado um buzzer e um led para sinalizar caso o sistema não esteja horizontalmente a esse plano.

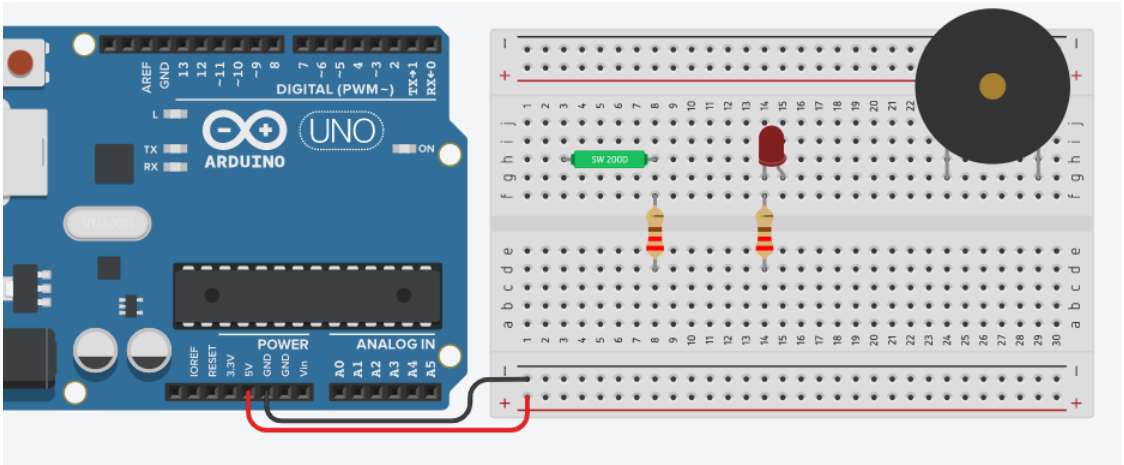
Circuito:

- O circuito é composto:
  - 1 Fonte de energia de 5 V;
  - 1 Arduino;
  - 1 Led;
  - 1 Buzzer;
  - 1 Resistor de 220 Ω;
  - 1 Resistor de 180 Ω;
  - 1 Sensor de inclinação SW-200D;
- Montagem do circuito

Etapa 1: Adicione a protoboard, o sensor de inclinação, Led, buzzer e o arduino ao circuito buscando "Breadboard ", "Tilt sensor", "LED", "Piezo" e "Arduino", respectivamente, nos componentes.

Etapa 2: Adicione e conecte um resistor de 220 Ω em um dos pinos do sensor de inclinação. Também é necessário adicionar outro resistor de mesma resistência no pino negativo do LED, como é mostrado na Fig. x.

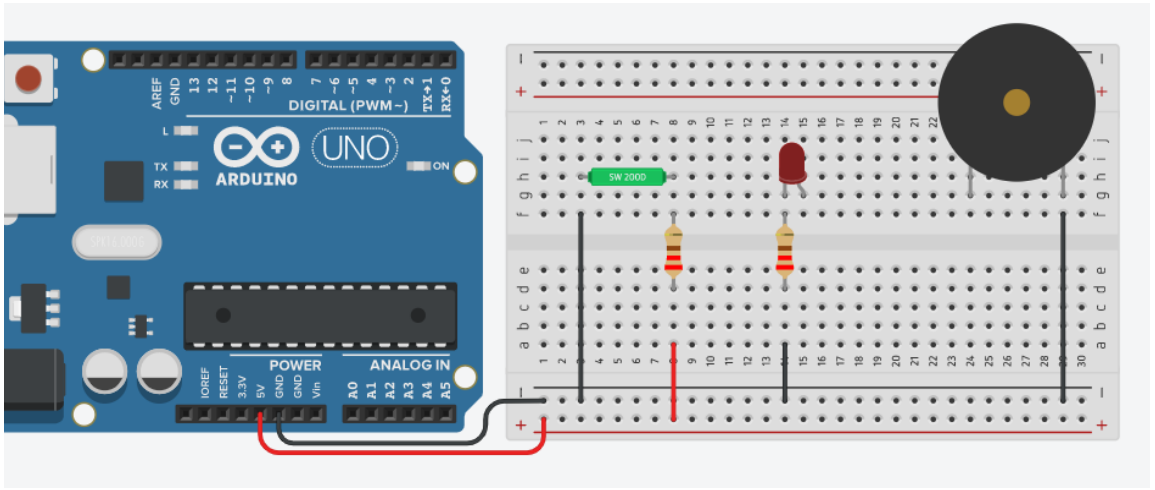
Figura x - Conexões dos resistores no circuito.



Fonte: Autor.

Etapa 3: Energije a protoboard conectando o 5 V e GND do Arduino na parte de alimentação da protoboard. Além disso, conecte 5 V no resistor do sensor de inclinação, já o GND, conecte tanto no sensor, quanto no resistor do LED e em um dos pinos do buzzer, como é mostrado na Fig. x

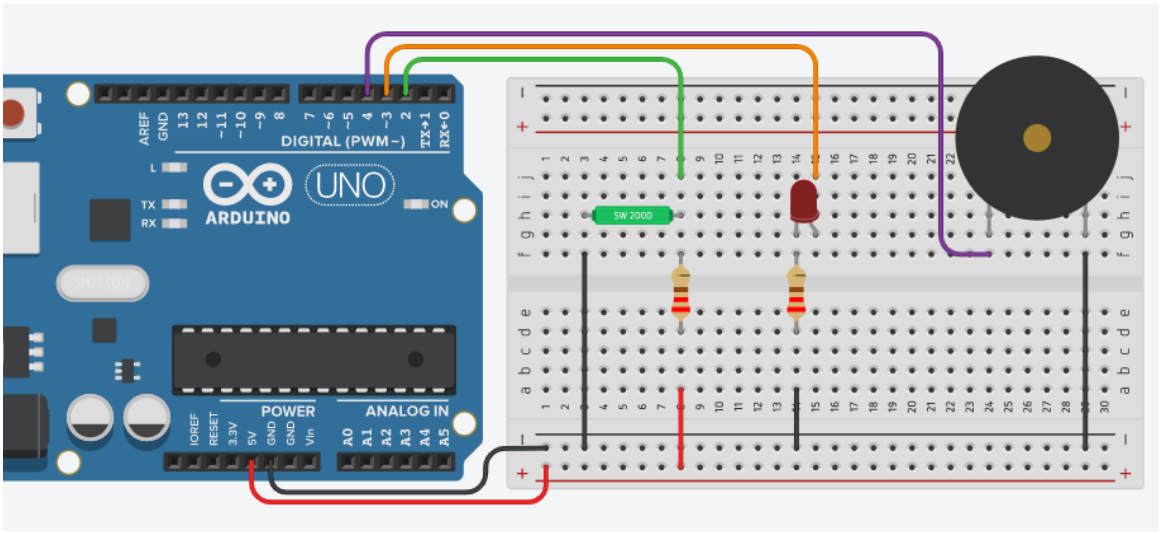
Figura x - Alimentação do circuito.



Fonte: Autor.

Etapa 4: Por fim, basta realizar as conexões com o Arduino. Dessa forma, conecte o pino positivo do sensor à porta 2, conecte o pino positivo do LED à porta 3 e conecte o outro pino do buzzer à porta 4, como é mostrado na Fig. x.

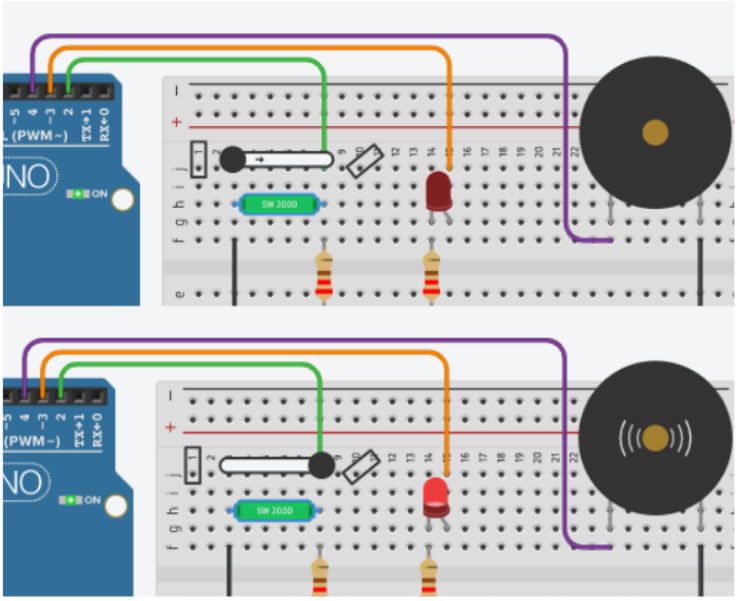
Figura x - Alimentação do circuito.



Fonte: Autor.

Por fim, cabe ressaltar que é possível modificar a inclinação do sistema após clicar no sensor e movimentar o círculo preto, como é mostrado na Fig. x.

Figura x - Alterando a inclinação do sistema.



Fonte: Autor.

Programação:

Inicialmente, é definido as configurações das portas, como a porta 2 que irá receber o valor do sensor SW-200D, por exemplo. Após isso, cria-se 3 variáveis para auxiliar no cálculo da frequência que o Buzzer irá emitir caso o sensor esteja indicando que o sistema saiu da horizontal. Dessa forma, a Fig. x mostra essas configurações iniciais e os comandos no setup.

Figura 1 - Configurações iniciais da programação do sistema.

```
//Autor: Thiago Victor A. de Freitas - Estudante de Engenharia Mecânica (UFC)

//Inicialmente, devemos nomear quem vai está conectado a cada porta
#define sensor 2 //Estabelece o pino 2 como a porta do sensor SW200D
#define led 3 //Estabelece o pino 3 como a porta do LED
#define buzzer 4 //Estabelece o pino 9 como a porta do buzzer
//Configurações para a frequencia de toque do buzzer
float seno;//Variável para armazenar o valor do seno
int frequencia;//Variável que irá armazenar o valor da frequencia
int x = 0;//Valor de referencia para a frequencia do buzzer

void setup(){
  pinMode(sensor, INPUT);//Configura sensor como porta de entrada de dados.
  pinMode(led, OUTPUT);//Configura led como porta de saída.
  pinMode(buzzer, OUTPUT);
  Serial.begin(9600);//Iniciando a comunicação serial do arduino
}
```

Fonte: Autor.

Dentro do loop, a primeira ação é calcular o valor da frequência aplicando o valor da variável x no comando seno e multiplicando por 500. A cada loop, incrementa-se 1 na variável x e, após ela ter um valor maior que 90, ela é reiniciada para dar uma variação de frequência constante ao buzzer.

Após isso, o valor lido pela porta do sensor é armazenado na variável valor_do_sensor. **Aqui é importante ressaltar que o sensor transmite 0 se o sistema estiver inclinado em relação a um plano de referência.** Dessa forma, se o valor lido for igual a 0, utiliza-se os comandos digitalWrite e tone para ativar, respectivamente, o LED e o buzzer. Em contrapartida, caso o valor seja diferente de 0, esses componentes são desligados e o sistema não irá sinalizar nada. A Fig. x mostra toda essa lógica utilizada no código do Arduino.

Figura 1 -Programação do loop do projeto.

```
19 void loop(){
20   //Calculando a frequencia do buzzer
21   seno = sin(x*3.1416/180);
22   frequencia = (int(seno*500));
23   if(x>90){// se for maior do que 90, reinicia
24     x = 0;
25   }else{//senão for maior que 90, segue incrementando
26     x++;
27   }
28   //Analisando valor do sensor
29   int valor_do_sensor = digitalRead(sensor);//Armazenando o valor transmitido
30   //pelo sensor na variavel inteira valor_do_sensor
31   Serial.println( valor_do_sensor);//Mostra o valor lido no serial monitor
32   if ( valor_do_sensor == 0){
33     //Se o sensor tiver na vertical, ele ativa os sinalizadores
34     digitalWrite(led,HIGH);//Ligando o LED
35 /*Comando tone(porta,frequencia,tempo): Gera uma onda quadrada
36 na frequência especificada em um pino.*/
37     tone(buzzer,frequencia);//Ligando o buzzer
38   }else{
39     //Se o sensor tiver na vertical, ele desliga os sinalizadores
40     digitalWrite(led,LOW);//Desligando o LED
41 /*Comando noTone(porta): Encerra o som iniciado com o comando tone*/
42     noTone(buzzer);//Desligando o buzzer
43   }
44   delay(10);//Espera 0,01 segundos antes de reiniciar o loop
45 }
```

Fonte: Autor.

Referências

Sites:

- **Circuito do projeto Sistema de Alarme**
- Descrição do sensor ultrassônico pelo site FIlipeFLOP
- **Circuito do projeto Sensor de Distância**
- Como funciona o sensor de inclinação
- Vídeo mostrando um circuito básico com o sensor de inclinação
- **Circuito do projeto Sensor de Inclinação**