4 – CIRCUITOS EM ARDUINO

Neste capítulo é abordado, elaborado e desenvolvido a construção de 3 circuitos Arduino, sensor de presença, sensor de proximidade sem e com display, no qual, possui desenvolvimento em linguagem de programação Arduino, através de várias técnicas e estudos para seja totalmente adequado aos parâmetros deste assunto.

4.1 DESENVOLVIMENTO

Em consideração a este estudo, a disciplina do curso Programação Orientada Objeto Avançada, orientou a realizar a criação do código e execução do circuito de Arduino com suas especificações para cada um.

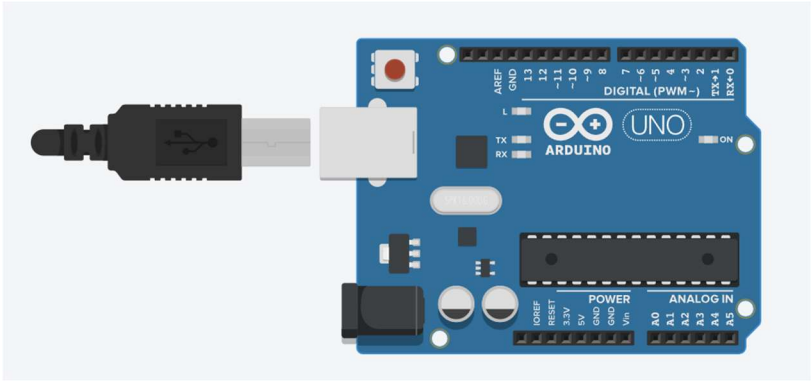
No Arduino existe um componente principal para todo projeto ele estará presente, no qual é, a placa para prototipagem eletrônica. Para este trabalho foi usado um ambiente virtual para simular os componentes chamado Tinkercad, podendo ser acessado em Tinkercad.com. Para os circuitos apresentados neste trabalho, a placa de prototipagem utilizada foi a Arduino Uno R3 e uma Protoboard (Placa perfurada para prototipagem rápida) no qual serve como um extensor da Placa principal.

O primeiro circuito consiste em um sensor de movimento, um buser e um LED. Os detalhes específicos para a resolução da atividade são, ao ser detectado um movimento o buser e o LED acenderá juntos e terá um intervalo de 1 segundo para ligado e 1 segundo para desligado. O Buser é nome do componente que é responsável por emitir som

Os componentes apresentados neste trabalho possuem a imagem de como é no ambiente virtual, porém o designer deles foi criado baseado nos componentes físicos existentes e eles são:

Arduino Uno R3 (Placa de Prototipagem Eletrônica):

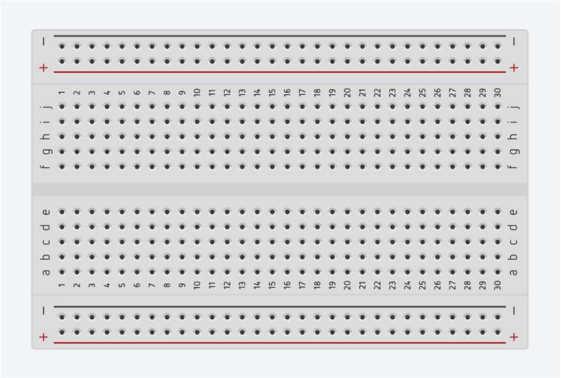
Figura 4.1 Arduino Uno R3



Fonte: Elaborada pelo Autor

Protoboard ou Breadboard (Placa perfurada para prototipagem rápida)

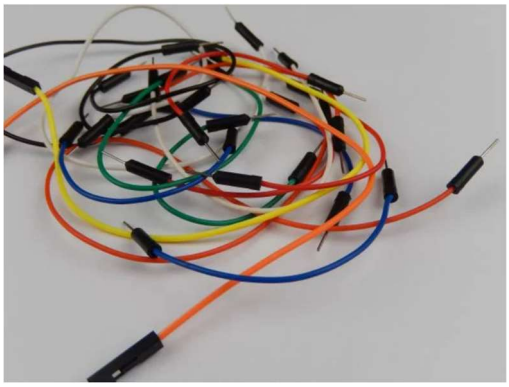
Figura 4.2 Protoboard



Fonte: Elaborada pelo Autor

Jumpers – Fios coloridos para interligar os componentes.

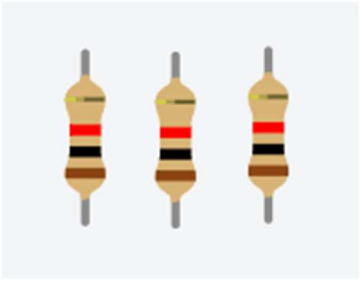
Figura 4.3 Jumpers



Fonte: FabioCosta, 2024

Resistores – Limitadores de corrente elétrica

Figura 4.4 Resistores

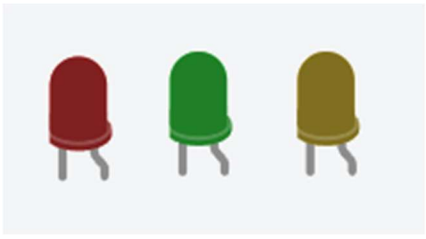


Fonte:

pelo Autor

LEDs

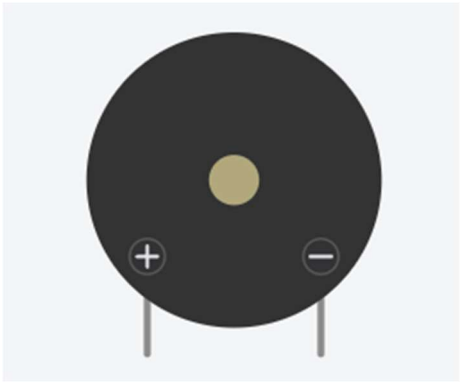
Figura 4.5 LEDs



Fonte: Elaborada pelo Autor

Buzzer – Emissor de Som

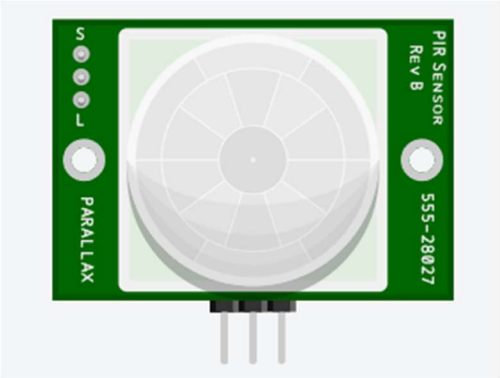
Figura 4.6 Buzzer



Fonte: Elaborada pelo Autor

Sensor de Movimento

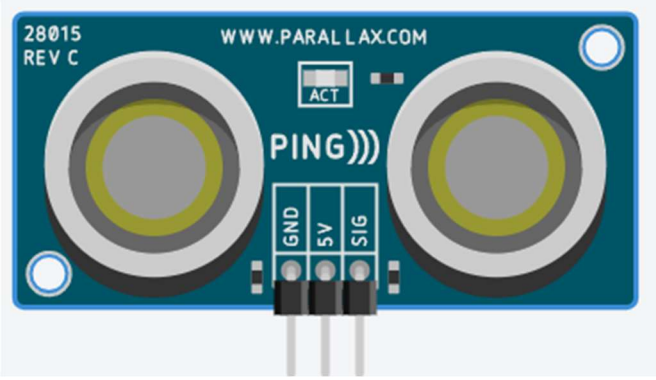
Figura 4.7 Sensor de Movimento



Fonte: Elaborada pelo Autor

Sensor de Proximidade

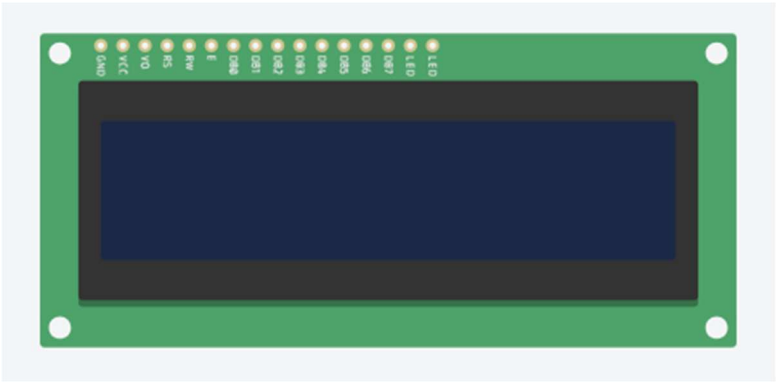
Figura 4.8 Sensor de Proximidade



Fonte: Elaborada pelo Autor

Display de LED

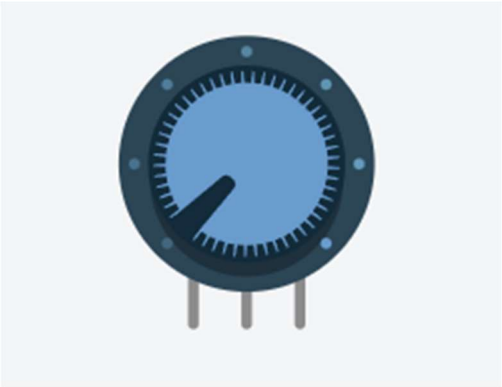
Figura 4.9 Display de LED 16x2



Fonte: Elaborada pelo Autor

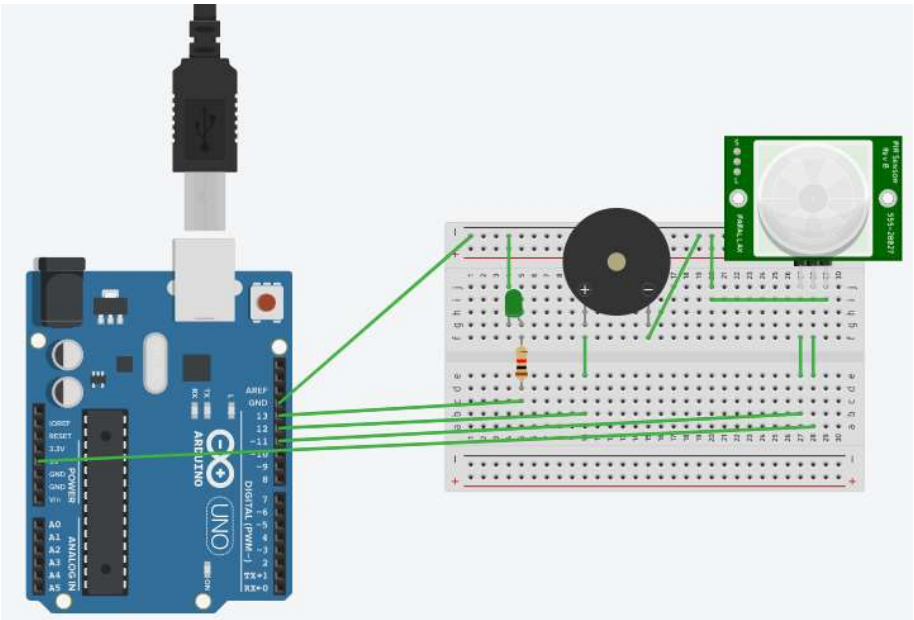
Potenciômetro

Figura 4.10 Potenciômetro



Fonte: Elaborada pelo Autor

Figura 4.11 Circuito Sensor de Movimento



Fonte: Elaborada pelo Autor.

Na figura 4.11 pode-se verificar como fica a estrutura do circuito montada que foi disponibilizado para desenvolvimento do código de programação.

Figura 4.12 Declaração de variáveis Circuito 1

```
1  // C++ code
2  //
3  int led = 13;
4  int sensor = 11;
5  int buzina = 12;
6
```

Fonte: Elaborada pelo Autor

A figura 4.12 exibe o início da construção do código onde efetua-se a declaração de variáveis que identifica qual porta o componente está ligado na placa de prototipagem. No caso da figura 4.11, o LED está conectado à porta 13, sensor porta 11 e a buzina (Buser) a porta 12.

Figura 4.13 Configuração de Output e Input circuito 1

```
 7  void setup()
 8  {
 9    pinMode(led, OUTPUT);
10    pinMode(sensor, INPUT);
11    pinMode(buzina, OUTPUT);
12
13  }
```

Fonte: Elaborada pelo Autor

Na figura 4.13 necessita-se a configuração para as portas declaradas, no qual será, *INPUT* para quando a porta recebe informação do componente ou *OUTPUT* para quando a porta emite informação para componente. No caso do circuito da figura 4.11 as portas de led e buzina recebem *OUTPUT* devido ao fato que elas recebem informação da placa de prototipagem e quanto sensor de movimento é *INPUT* pois capta a informação do movimento e envia para a placa.

Figura 4.14 Construção do código infinito.

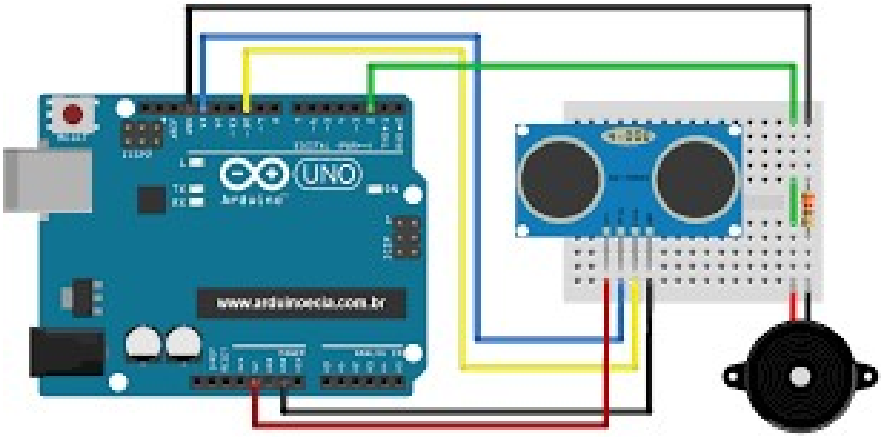
```
15  void loop()
16  {
17    if(digitalRead(sensor) == 1){
18      digitalWrite(led, 1);
19      digitalWrite(buzina, 1);
20      delay(1000);
21      digitalWrite(led, 0);
22      digitalWrite(buzina, 0);
23      delay(1000);
24      digitalWrite(led, 1);
25      digitalWrite(buzina, 1);
26      delay(1000);
27      digitalWrite(led, 0);
28      digitalWrite(buzina, 0);
29      delay(1000);
30      digitalWrite(led, 1);
31      digitalWrite(buzina, 1);
32      delay(1000);
33      digitalWrite(led, 0);
34      digitalWrite(buzina, 0);
35      delay(1000);
36    }
37
38  }
```

Fonte: Elaborada pelo Autor

Por fim, esta etapa é responsável pelo código de comportamento dos componentes. Nele declaramos o que fazer, quando fazer e quanto tempo fazer, todo código inserido dentro do *void loop* (Infinito) permanece em execução até que ser interrompido. Como na figura 4.14 declara-se que quando o sensor identificar um

movimento o seu valor passará ser 1 e ao acontecer isso, dispara a função no qual realiza o acionamento do buser para emissão de som e o LED ascender. Esse processo dura 1 segundo com os componentes ligados e 1 segundo desligado, essa ação irá acontecer até que o sensor não detecte nada e o valor volte a ser 0.

Figura 4.15 Circuito Sensor de Proximidade



Fonte: Elaborada pelo Autor

A figura 4.15 representa a forma no qual o circuito 2 é montado, sendo utilizado Arduino Uno R3, Protoboard, Sensor de proximidade, buser, um resistor, um buser e jumpers. O objetivo deste circuito é calcular a distância de um objeto através tendo por base o som sendo emitido. Quando a distância estiver acima de 20 centímetros o buzzer permanecerá desligado, entre 20 e 10 centímetros o buzzer deverá ligar e desligar em intervalos de 1 segundo e menos de 10 centímetros o buzzer permanecerá ligado constantemente.

Figura 4.16 Declaração de variáveis circuito 2

```
1  int trig = 13;
2  int echo = 8;
3  int buzina = 2;
4
5  float distancia;
```

Fonte: Elaborada pelo Autor

A figura 4.16 demonstra o escopo do código para este circuito onde declaramos as variáveis para representar quais portas os componentes estão conectados. O sensor de proximidade sempre é declarado duas portas, chamada TRIG e ECHO, elas estão informadas no componente, no caso do circuito a TRIG está na porta 13 e o ECHO está na porta 8, enquanto o buzzer está na porta 2 e por fim declaramos uma variável para conter o valor da distância que for detectada.

Figura 4.17 Configuração de Output e Input circuito 2

```
7  void setup()
8  {
9    pinMode(buzina, OUTPUT);
10   pinMode(trig, OUTPUT);
11   pinMode(echo, INPUT);
12   Serial.begin(9600);
13 }
```

Fonte: Elaborada pelo Autor

A figura 4.17 representa a configuração das portas registrada para identificação sobre ser OUTPUT ou INPUT. O buzzer sempre será configurado como OUTPUT, para o sensor devemos registrar uma informação a mais, a porta TRIG é OUTPUT, a porta ECHO é INPUT e o comando “Serial.begin(9600)” é para inicializar a comunicação serial entre Arduino e o dispositivo externo e o termo “9600” seria a taxa de transmissão de bytes por segundos.

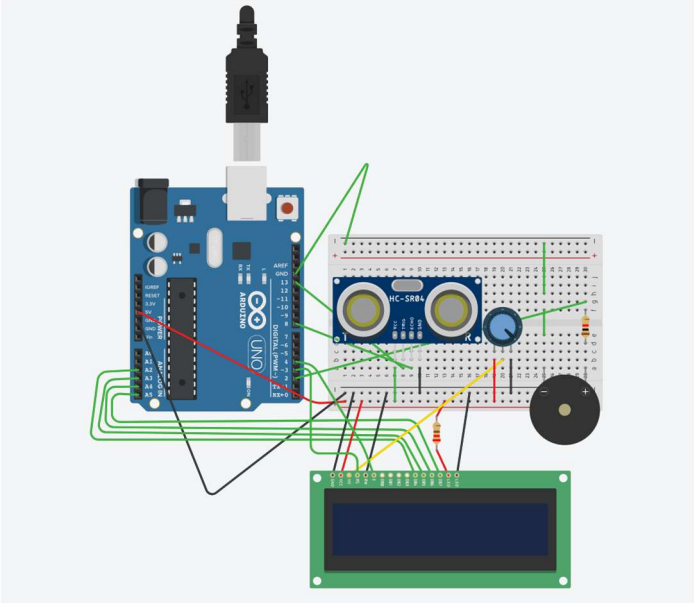
Figura 4.18 Construção do código infinito.

```
14 void loop()
15 {
16   digitalWrite(trig, 0);
17     delay(0005);
18     digitalWrite(trig, 1);
19     delay(0010);
20     digitalWrite(trig, 0);
21
22     distancia = pulseIn (echo, 1);
23     distancia = distancia/58;
24     Serial.println (distancia);
25   if(distancia > 20){
26     digitalWrite(buzina, 0);
27   }
28   if(distancia <= 20 && distancia >= 10){
29     digitalWrite(buzina, 1);
30     delay(1000);
31     digitalWrite(buzina, 0);
32     delay(1000);
33   }
34
35   if(distancia < 10){
36   digitalWrite(buzina, 1);
37   }
38 }
39
```

Fonte: Elaborada pelo autor.

Neste bloco de código é responsável pelo comportamento dos dispositivos externo. O início do código é responsável por dar o acionamento do sensor para a leitura através do pulo ultrassónico que ele emite. Posteriormente é realizado o cálculo com os dados coletados pelo sensor. Por fim é processado a informação da distância dando ela comportamentos diferentes para determinados resultados. Para distância acima de 20 centímetros não haverá nenhum comportamento, entre 20 e 10 centímetros o buzzer terá acionamento a cada 1 segundo e abaixo de 10 centímetros o buzzer ficará constantemente ligado.

Figura 4.19 Sensor de Proximidade com Display



Fonte: Elaborada pelo Autor

A figura 4.19 representa o circuito 3, ele apresenta o mesmo comportamento do circuito 2, o extra é a utilização de um display de led para informar qual a distância do objeto detectado. Para o funcionamento do display, deve-se adicionar alguns comandos a mais durante todo o código.

Figura 4.20 Declaracoes de variáveis para o Display

```
1  #include <LiquidCrystal.h>
2  const int rs = 3, en = 4, d4 = A2, d5 = A3, d6 = A4, d7 = A5;
3  LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
4
```

Fonte: Elaborada pelo autor

A figura 4.20 demonstra a declaração de variáveis para o funcionamento do display, primeiro efetuamos a inclusão de uma biblioteca do Arduino que auxilia o funcionamento do display chamada “LiquidCrytal.h”, em seguida, declaramos as portas que estão sendo utilizadas e por fim o comando “LiquidCrystal” para instanciar o objeto da classe “LiquidCrystal” que permite-nos controlar o display LCD.

Figura 4.21 Configuração do Void Setup do Display

```
21    lcd.begin(16, 2);
22    lcd.setCursor(1,1);
23    lcd.print("Inicializando...");
24    delay(1000);
25    lcd.clear();
26    lcd.setCursor(2,0);
27    lcd.print("Sistema OK!");
28    delay(1000);
29    lcd.clear();
```

Fonte: Elaborada pelo autor

A figura 4.21 representa a configuração que deve ser realizada no void Setup do código. Esta configuração faz a inicialização do display e emite a mensagem confirmando sua inicialização.

Figura 4.22 Código adicional do display

```
41    lcd.setCursor(0,0);
42    lcd.print("Distanticia: ");
43    delay(1000);
44
45    lcd.setCursor(0,1);
46    lcd.print(distancia);
47    delay(1000);
```

Fonte: Elaborada pelo autor

A figura 4.22 é o código que se deve incluir para que o display demonstre a informação que desejar, o posicionamento de bloco de código deve ficar após o cálculo da distância, primeira parte do código emite a mensagem na primeira linha do display e posteriormente a segunda parte transmite na segunda linha. A figura 4.23 demonstra o resultado do display.

Figura 4.23 Exemplo de Resultado do Display

Fonte: Elaborada pelo autor

4.2 Conclusão

O tema abordado neste capítulo, circuitos em Arduino, foi uma experiencia enriquecedora. Por meio destes circuitos pude aplicar os conhecimentos adquiridos na respectiva matéria. Enfrentando um desafio real e desenvolver soluções eficientes.

Estes circuitos em Arduino não é apenas um projeto qualquer, mas também representa um marco em minha jornada de conhecimento. Consolidando todo conhecimento, técnicas em apenas um lugar.