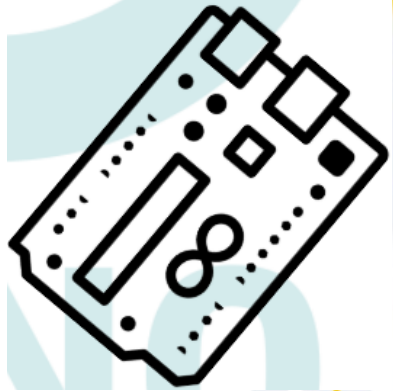
Arduino: Conceitos, Programação e Aplicação Prática



AULA
4

Display LCD Sensores (TMP36, LDR e Ultrassônico)

Coordenador
Fábio Favarim

Instrutor
Gabriel S. Folly

CAMPUS PATO BRANCO

UniversidadequeTransforma

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS PATO BRANCO

Sumário

Introdução ao Display LCD 16x2

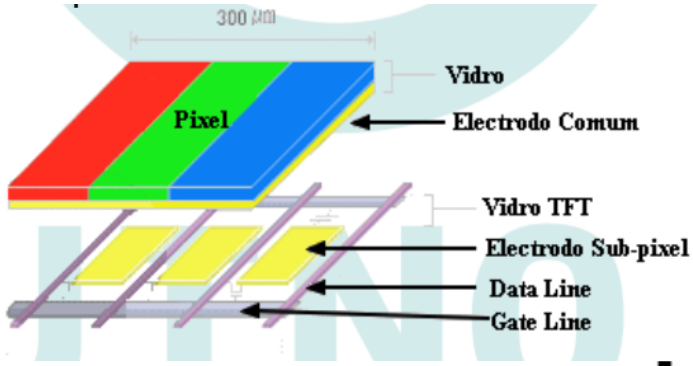
- Versatilidade e Popularidade:
  - Componente eletrônico para projetos em placas e microcontroladores.
  - Destaque para iniciantes devido à versatilidade.
- Controlador HD44780:
  - Equipado com o controlador HD44780.
  - Facilita o desenvolvimento de interfaces homem-máquina.

Características Técnicas e Aplicações

- Configuração de Pinos:
  - 16 pinos, com 12 na conexão básica.
  - 11 I/O, 2 alimentação, 2 controle de backlight e contraste.
- Principais Características:
  - Corrente de trabalho, tensão de operação, dimensões, área do visor, etc.
- Aplicações Comuns:
  - Letreiros, amperímetros, voltímetros, painéis de controle, calculadoras digitais, entre outros.

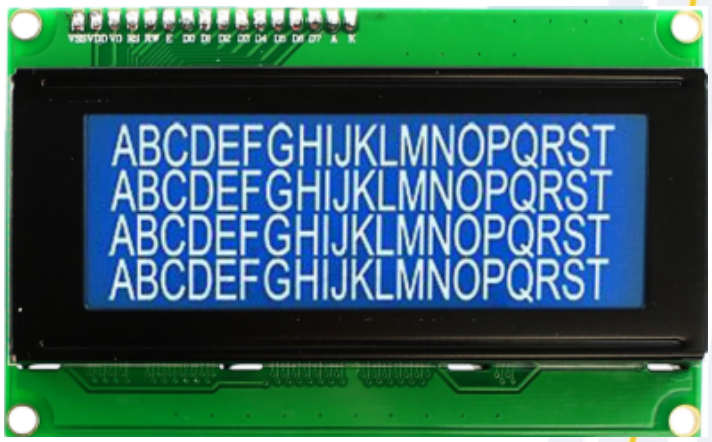
Tecnologia de Cristal Líquido

- Princípio de Funcionamento:
  - Utiliza a tecnologia de cristal líquido.
  - Estado intermediário entre líquido e sólido.
- Eficiência e Iluminação LED:
  - Alternativa eficiente, iluminada por LED.
  - Adequada para projetos menos complexos.



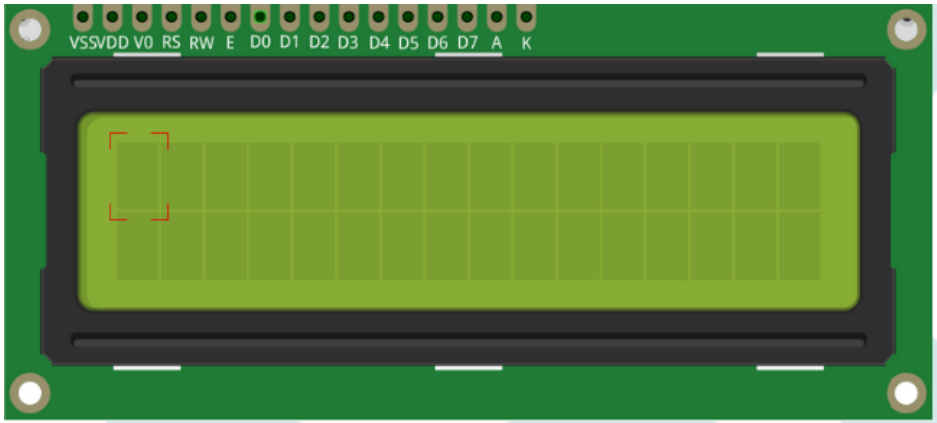
Diferença entre 16x2 e 20x4

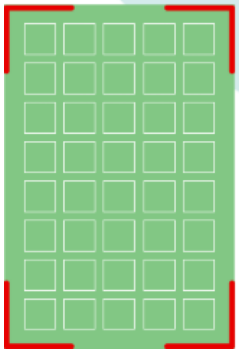
- Pinagem Idêntica:
  - Mesma pinagem entre os displays 16x2 e 20x4.
- Principal Distinção:
  - Quantidade de caracteres exibidos.
  - 20x4 permite mais informações na tela.



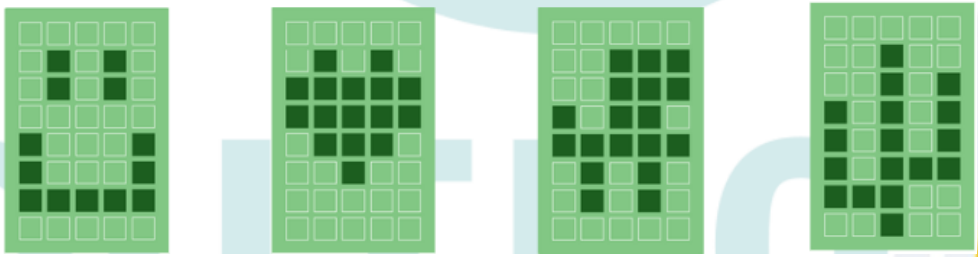
Uso do Display LCD no Arduino

- Conexão ao Arduino:
  - Duas maneiras:
    - barramento paralelo (4 ou 8 bits) ou barramento serial I2C.
- Configuração Paralela:
  - Mais ligações necessárias.
- Configuração I2C:
  - Simplifica o processo, requer apenas dois pinos.





É possível através do display não somente apresentar textos e números, mas também criar formas.



Funções dos Pinos

Os displays LCD 16x2 geralmente possuem 16 pinos, cada um com uma função específica. Aqui estão as funções dos pinos mais comuns em um display LCD 16x2:

- VSS (Ground): Conectado ao terra (GND) do circuito para fornecer a referência de potencial.
- VDD (Power Supply): Conectado à alimentação positiva (VCC) para fornecer a energia necessária ao display.
- VO (Contrast): Usado para ajustar o contraste do display. Ajustado por meio de um potenciômetro.
- RS (Register Select): Define se os dados fornecidos aos pinos D0-D7 são dados ou comandos. RS=0 para comandos, RS=1 para dados.

- RW (Read/Write): Define se a operação é de leitura (RW=1) ou escrita (RW=0). Muitas vezes, mantido em GND para operação de escrita.
- E (Enable): Ativa a leitura ou escrita de dados quando transita de 0 para 1.
- D0-D7 (Data Lines): Linhas de dados utilizadas para enviar dados ou comandos ao display. A quantidade de linhas (4 ou 8) depende da configuração do display.
- A (Anode): Alimentação positiva do backlight (luz de fundo).
- K (Cathode): Conectado ao terra (GND) para fechar o circuito do backlight.
- BLA e BLK (Backlight Anode e Backlight Cathode): Variações dos pinos A e K, são usados para alimentação do backlight em alguns modelos.

Bibliotecas para Display LCD 16x2

- LiquidCrystal Library:
  - Biblioteca padrão no ambiente de desenvolvimento do Arduino.
  - Oferece funções básicas para controlar displays LCD, incluindo o 16x2.
  - Documentação oficial fornece detalhes: LiquidCrystal Library.
- HD44780 Library:
  - Implementação moderna do controlador HD44780.
  - Compatível com diversos modelos, incluindo o 16x2.
  - Recursos adicionais e otimizações em comparação com a LiquidCrystal Library.
  - Instalação direta pelo Gerenciador de Bibliotecas do Arduino.

- LCD I2C Library:
  - Projetada para displays LCD com interface I2C.
  - Simplifica comunicação com displays 16x2 que possuem adaptadores I2C.
  - Instalação direta pelo Gerenciador de Bibliotecas do Arduino.
- Simplificando o Controle:
  - Bibliotecas facilitam o controle do display LCD 16x2.
  - Foco na lógica do projeto, sem lidar diretamente com detalhes de baixo nível.
- Escolha da Biblioteca:
  - Para nossos projetos, utilizaremos a biblioteca LiquidCrystal.
  - Certifique-se de instalar a biblioteca adequada ao seu display e adaptador I2C, se aplicável.

Funções da Biblioteca LiquidCrystal

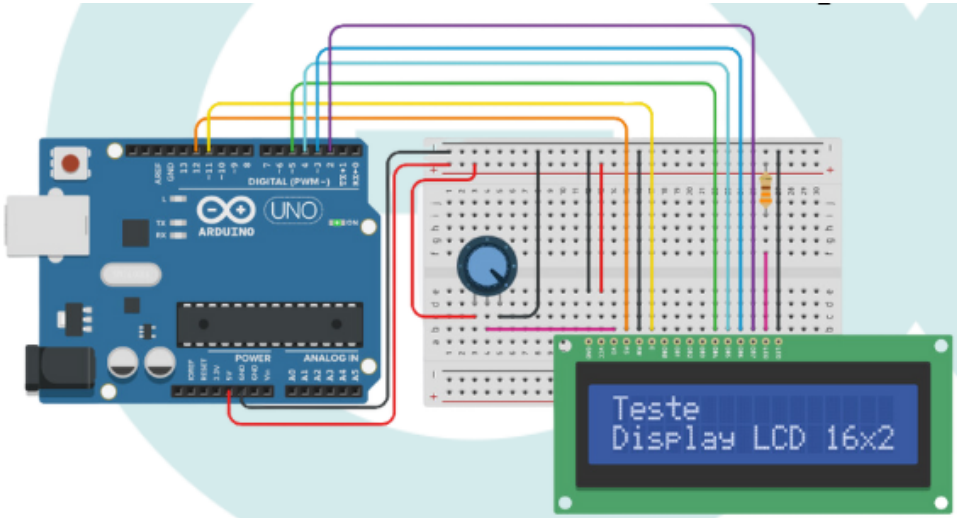
- LiquidCrystal():
  - Construtor para criar uma instância da classe 'LiquidCrystal'. É preciso definir os pinos de controle do seu display no construtor.
- begin(cols, rows):
  - Inicializa o display com o número de colunas e linhas especificado.
- clear():
  - Limpa o conteúdo do display.
- home():
  - Move o cursor para a posição inicial (1ª coluna, 1ª linha).
- setCursor(col, row):
  - Define a posição do cursor no display.

- print(data):
  - Exibe texto ou dados no display. Pode receber strings, números, caracteres, etc.
- write(character):
  - Escreve um único caractere no display.
- scrollDisplayLeft() e scrollDisplayRight():
  - Rola o conteúdo do display para a esquerda ou direita.
- blink() e noBlink():
  - Ativa ou desativa o modo de piscar o cursor.
- cursor() e noCursor():
  - Ativa ou desativa a exibição do cursor.
- createChar():
  - Define padrões de pixels específicos que serão utilizados para criar caracteres personalizados.

Exemplo Prático

```
1   #include <LiquidCrystal.h> // importando a biblioteca para que se possa utilizar de suas funções
2
3   LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  // Pinos RS, E, D4, D5, D6, D7
4   lcd.begin(16, 2);
5   lcd.clear();
6   lcd.home();
7   lcd.setCursor(0, 1); // Define o cursor na 1ªcoluna, 2ªlinha
8   lcd.print("Hello, World!");
9   lcd.write("K");
10  lcd.scrollDisplayLeft();
11  lcd.scrollDisplayRight();
12  lcd.blink();
13  lcd.noBlink();
14  lcd.cursor();
15  lcd.noCursor();
```

Exemplo Prático




```arduino
#include <LiquidCrystal.h> // incluindo a biblioteca

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // portas do arduino que estão sendo utilizadas pelo display LCD

void setup()
{
  lcd.begin(16, 2);
}
void loop()
{
  lcd.setCursor(0,0); // setando a posicao em que o cursor irá inicializar, coluna 1 linha 1.
  lcd.print("Teste"); // imprime o texto
  lcd.setCursor(0,1); // setando a posicao em que o cursor irá estar agora, coluna 1 linha 2.
  lcd.print("Display LCD 16x2"); // imprime na segunda linha.
}
```

Exemplo Prático Caracteres Especiais


```
#include <LiquidCrystal.h>
LiquidCrystal lcd = LiquidCrystal(12, 11, 5, 4, 3, 2);

// Campo de caracteres personalizados 5x8
// Onde esta com valor 1 quer dizer que existe conteudo onde ten 0 nao tem conteudo
byte Coracao[8] = {
  B00000,
  B01010,
  B10101,
  B10001,
  B01010,
  B00100,
  B00000,
  B00000
};
byte Sorriso[8] = {
  B00000,
  B00000,
  B01010,
  B01010,
  B00000,
  B10001,
  B01110,
  B00000
};
```

```
byte Sound[8] = {
  B00001,
  B00011,
  B00101,
  B01001,
  B01001,
  B01011,
  B11011,
  B11000
};
byte Cadeado[8] = {
  B01110,
  B10001,
  B10000,
  B11111,
  B11011,
  B11011,
  B11111,
  B00000
};
byte Lua[8] = {
  B00111,
  B01110,
  B11100,
  B11100,
  B11100,
  B11100,
  B01110,
  B00111
};
```

```
byte Dinossauro[8] = {
  B00000,
  B00111,
  B00111,
  B10110,
  B11111,
  B01010,
  B01010,
  B00000
};
byte Arvore[8] = {
  B00000,
  B00100,
  B00101,
  B10101,
  B10101,
  B10111,
  B11100,
  B00100
};
```

```
void setup()
{
  lcd.begin(16, 2);
  // criando os caracteres personalizados
  lcd.createChar(0, Coracao);
  lcd.createChar(1, Sorriso);
  lcd.createChar(2, Lua);
  lcd.createChar(3, Caceado);
  lcd.createChar(4, Dirossauro);
  lcd.createChar(5, Arvore);
  lcd.clear();
  delay(100);
  lcd.print("Testando...");
}
void loop() {
  // Imprime todos os caracteres personalizados:
  lcd.setCursor(0, 1); //seta onde será impresso coluna 1 linha 2
  lcd.write(byte(0)); // Seleciona o caracter especial
  lcd.setCursor(2, 1); // coluna 3 linha 2
  lcd.write(byte(1));
  lcd.setCursor(4, 1);
  lcd.write(byte(2));
  lcd.setCursor(6, 1);
  lcd.write(byte(3));
  lcd.setCursor(8, 1);
  lcd.write(byte(4));
  lcd.setCursor(10, 1);
  lcd.write(byte(5));
}
```

Funcionamento do Sensor TMP36

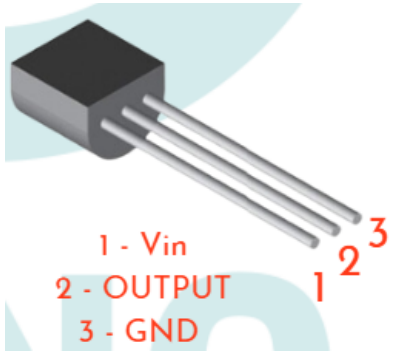
- Sensor Analógico:
  - Converte a temperatura em uma voltagem proporcional.
  - Saída analógica linear proporciona leitura direta em graus Celsius.

Características Técnicas do TMP36

- Faixa de Temperatura:
  - Varia de -40°C a 125°C.
- Saída Linear:
  - 10 mV por grau Celsius.
- Alimentação:
  - Funciona com ampla faixa de tensão (geralmente 2,7V a 5,5V).
- Baixo Consumo de Energia:
  - Ideal para projetos alimentados por bateria.

Pinagem e Calibração

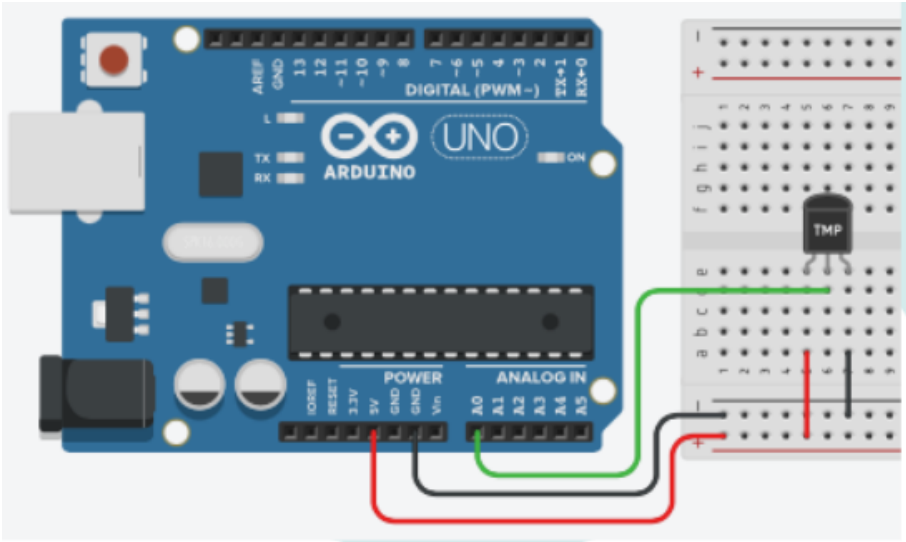
- Pinagem:
  - Pino de Sinal (Vout): Voltagem proporcional à temperatura.
  - Pino de Alimentação (V+): Conectado à fonte de alimentação.
  - Pino de Terra (GND): Conectado ao terra do circuito.
- Calibração e Precisão:
  - Geralmente não requer calibração adicional.
  - Possui precisão aceitável para muitas aplicações.



Aplicações e Considerações de Uso

- Aplicações Comuns:
  - Projetos de monitoramento de temperatura (medir a temperatura ambiente).
  - Controle de dispositivos térmicos em sistemas de resfriamento ou aquecimento.
- Considerações de Uso:
  - Isolamento térmico pode ser necessário em aplicações sensíveis.

Exemplo Prático 2




```
Monitor serial

Temperatura: 24.78 Graus Celsius
Temperatura: 24.78 Graus Celsius
Temperatura: 51.17 Graus Celsius
Temperatura: 51.17 Graus Celsius
Temperatura: 51.17 Graus Celsius
```

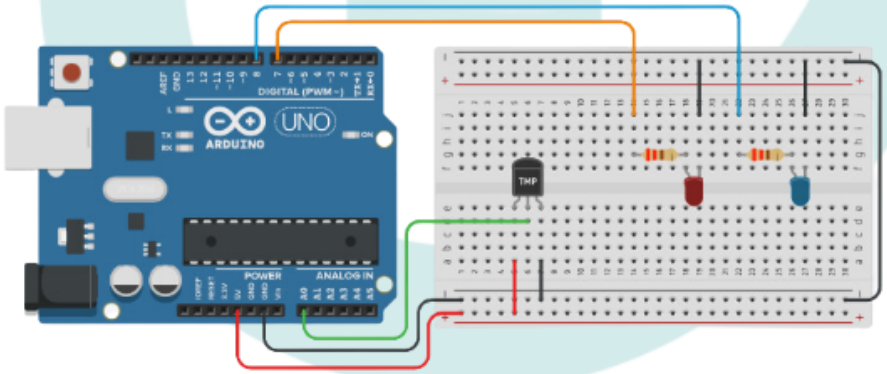
```
1  // Define o pino ao qual o pino de saída do TMP36 está conectado
2  const int temp = A0;
3
4  void setup()
5  {
6    // Inicializa a comunicação serial para monitoramento
7    Serial.begin(9600);
8  }
9
10 void loop()
11 {
12   // Lê a voltagem analógica do sensor TMP36
13   int leitura = analogRead(temp);
14
15   // Converte a leitura analógica para temperatura em graus Celsius
16   float voltage = leitura * (5.0 / 1023.0); // Converte para voltagem
17   float temperatura = (voltage - 0.5) * 100.0; // Converte para temperatura em Celsius
18
19   // Exibe a temperatura no Monitor Serial
20   Serial.print("Temperatura: ");
21   Serial.print(temperatura);
22   Serial.println(" Graus Celsius");
23
24   delay(1000); // Aguarda 1 segundo antes de realizar a próxima leitura
25 }
```

Laboratório 01

Simulando temperatura visualmente com sensor de temperatura e LED Azul e LED Vermelho
Neste guia, você aprenderá a montar um circuito utilizando um sensor de temperatura e 2 LEDs, para simular um ambiente frio com azul e um ambiente quente com o vermelho.

- Materiais Necessários:
  - Arduino Uno, Protoboard e Jumpers.
  - 2 Leds (vermelho e azul).
  - 2 Resistores de 220 (para os LEDs).
  - 1 Sensor TMP36.

Solução Laboratório 01



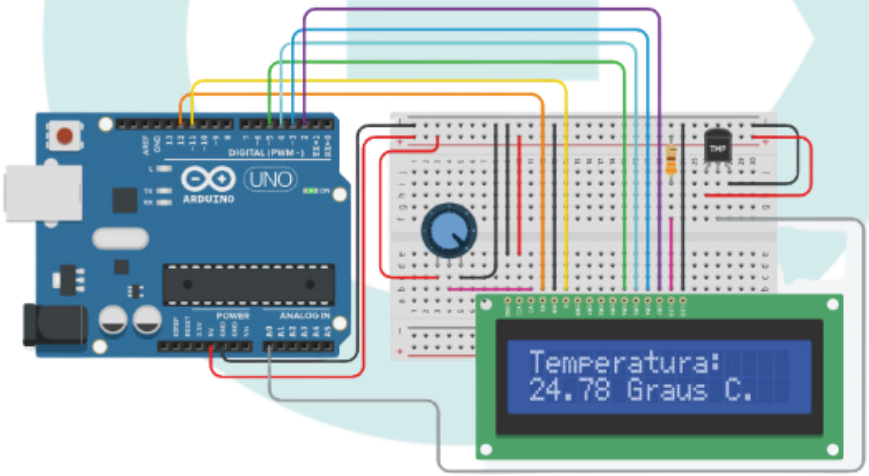

```arduino
// Define o pino ao qual o pino de saída do TMP36 está conectado
const int temp = A0;
int vermelho = 7;
int azul = 8;

void setup()
{
  pinMode(vermelho, OUTPUT);
  pinMode(azul, OUTPUT);
}

void loop()
{
  // Lê a voltagem analógica do sensor TMP36
  int sensorValue = analogRead(temp);

  // Converte a leitura analógica para temperatura em graus Celsius
  float voltage = sensorValue * (5.0 / 1023.0);  // Converte para voltagem
  float temperatureC = (voltage - 0.5) * 100.0;   // Converte para temperatura em Celsius

  // Se a temperatura for menor que 10 graus celsius liga azul. FRIO
  if(temperatureC < 10.0)
  {
    digitalWrite(azul, HIGH);
    digitalWrite(vermelho, LOW);
  }
  // Se a temperatura for maior que 25 graus celsius liga vermelho. QUENTE
  if(temperatureC > 25.0)
  {
    digitalWrite(azul, LOW);
    digitalWrite(vermelho, HIGH);
  }

  delay(1000);  // Aguarda 1 segundo antes de realizar a próxima leitura
}
```

Laboratório 02

Temperatura ambiente impressa no display LCD 16x2
Neste guia, você aprenderá a imprimir no display LCD 16x2 a temperatura captada no sensor TMP36.

- Materiais Necessários:
  - Arduino Uno, Protoboard e Jumpers.
  - 1 Resistor de 330 ohms.
  - 1 Sensor TMP36.
  - 1 Display LCD 16x2.
  - 1 Potenciômetro.

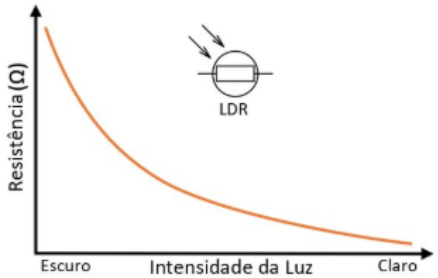
Solução Laboratório 02




```arduino
#include <LiquidCrystal.h>

const int temp = A0;
LiquidCrystal lcd = LiquidCrystal(12, 11, 5, 4, 3, 2);

void setup()
{
  lcd.begin(16, 2);
}
void loop()
{
  int sensorValue = analogRead(temp);

  // Converte a leitura analógica para temperatura em graus Celsius
  float voltage = sensorValue * (5.0 / 1023.0);  // Converte para voltagem
  float temperatureC = (voltage - 0.5) * 100.0;   // Converte para temperatura em Celsius

  lcd.setCursor(0, 0);
  lcd.print("Temperatura: ");
  lcd.setCursor(0, 1);
  lcd.print(temperatureC);
  lcd.setCursor(5, 1);
  lcd.print(" Graus C.");

  delay(1000);
}
```

Introdução ao Sensor LDR

- LDR (Light Dependent Resistor) é um componente eletrônico sensível à luz.
- Sua resistência elétrica varia conforme a intensidade luminosa.
- Amplamente utilizado em automação, eletrônica e projetos diversos.
- Composto por material semicondutor.
- Condução influenciada pela luz.
- Resistência diminui em ambientes iluminados e aumenta na escuridão.



Medição de Resistência com Multímetro

- Possibilidade de medir resistência usando um multímetro.
- Em ambientes iluminados, a resistência diminui, permitindo a passagem de corrente.
- Em ambientes escuros, a resistência aumenta.

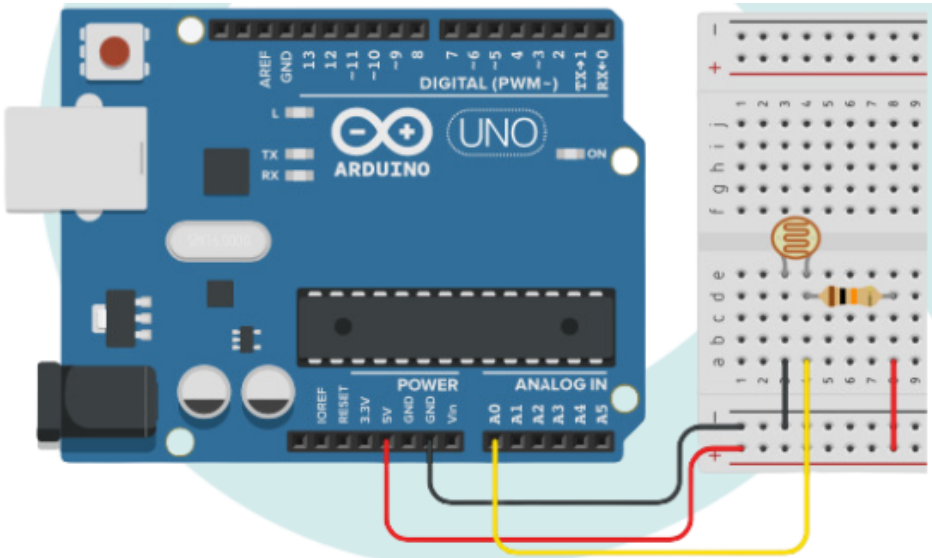
Aplicações Práticas

- Sistemas de iluminação automática.
- Reguladores de brilho.
- Dispositivos de segurança.
- Circuitos de controle de exposição em câmeras fotográficas.

Funcionamento Analógico

- O LDR faz leituras analógicas.
- Conectado em portas analógicas (A0 - A5).
- Conversão para escala digital de 0 a 1023.
  - 0 representa ausência de luz.
  - 1023 representa a maior intensidade possível de luz.

Conexão e Leitura




```
1   // declaração do pino do LDR e variável que armazenara a leitura
2   int LDR = A0;
3   int leitura = 0;
4
5   void setup()
6   {
7     pinMode(LDR, INPUT); // como LDR é um sensor ele irá captar informações logo é declarado como ENTRADA
8     Serial.begin(9600);
9   }
10
11  void loop()
12  {
13    leitura = analogRead(LDR);
14
15    Serial.print("Valor lido pelo LDR: ");
16    Serial.println(leitura);
17    delay(100);
18
19  }
```

Sensor Ultrassônico

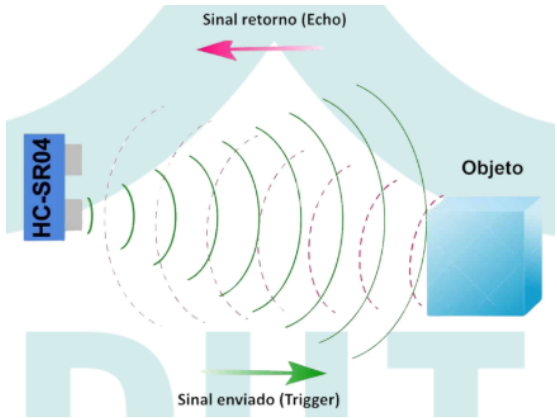
- Utiliza ondas sonoras ultrassônicas para medir distâncias com alta precisão.
- Composto por emissor e receptor ultrassônico.
- deal para detecção de obstáculos e medição de distâncias em projetos robóticos e automação industrial.
- O sensor ultrassônico HC-SR04 possui 4 pinos.
  - VCC (5V), TRIG (disparo), ECHO (eco), GND (terra).

Vantagens e Aplicações

- Precisão na medição.
- Independência de luz ambiente.
- Resistência a interferências eletromagnéticas.
- Alcance de 2 cm a 4 metros com margem de erro de 3 mm.
- Vantagens: Precisão, independência de luz ambiente e resistência a interferências.
- Aplicações: Segurança, navegação autônoma e controle à distância.

Funcionamento

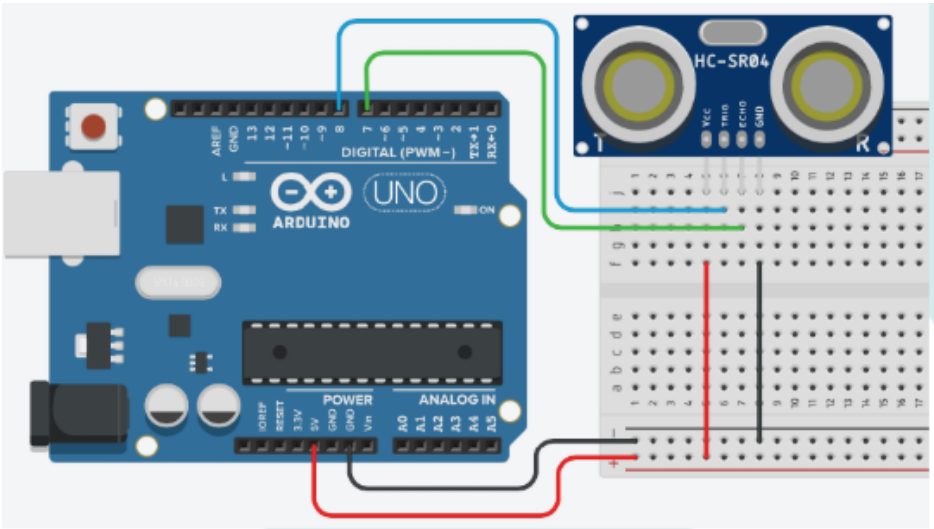
- Emissão de ondas ultrassônicas pelo transdutor piezoelétrico.
- Reflexão ao encontrar obstáculo e recepção no segundo transdutor.
- Tempo de voo medido entre emissão e recepção das ondas.
- Fórmula: Distância = (Tempo de Voo * Velocidade do Som) / 2.



Função ‘pulseIn()’

- Captura a duração de um pulso em um pino (HIGH ou LOW).
- Utilizada para medir o tempo de voo das ondas ultrassônicas.
- Funciona em pulsos de 10 microssegundos a 3 minutos de duração.
- ’pulseIn()’ mede tempo de pulso, fundamental para o sensor ultrassônico.

Conexão com o Arduino



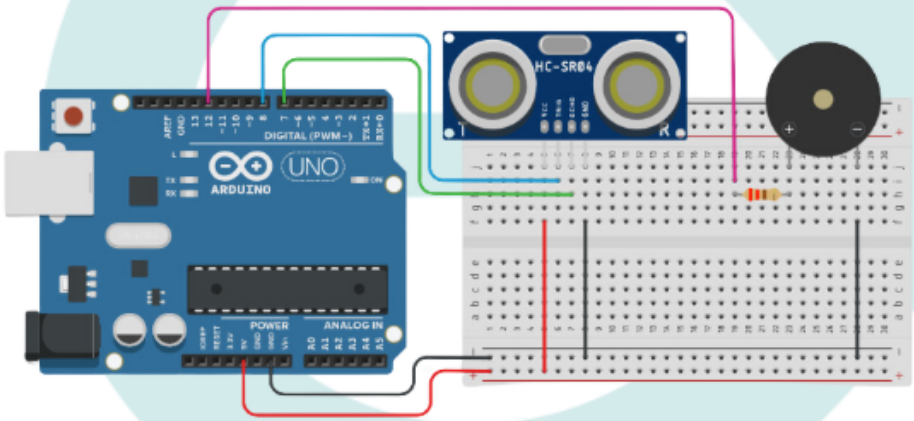

```
1  int trig = 8; // declarando o pino trigger SINAL EMITIDO
2  int echo = 7; // declarando o echo SINAL RECEBIDO
3
4  float distancia; // variável que armazenará a distancia
5  unsigned long tempo; // variável que armazenará o tempo
6
7  void setup()
8  {
9    pinMode (trig, OUTPUT); // trigger é declarado como saída pois emite o sinal
10   pinMode (echo,INPUT); // echo é declarado como entrada pois recebe o sinal
11   Serial.begin (9600);
12 }
13 void loop()
14 {
15   digitalWrite(trig, HIGH); // envia um sinal de nível alto por 10 microssegundos
16   delayMicroseconds(10);
17   digitalWrite(trig, LOW);
18
19   tempo = pulseIn(echo, HIGH); // obtém em quanto tempo o sinal foi recebido
20   //Função para calcular a distandia -> distancia = (tempo voo * velocidade som) / 2;
21   distancia = (tempo * 0.034029) / 2;
22   Serial.print(distancia);
23   Serial.println (" cm");
24   delay(1000);
25 }
```

Laboratório 01

Sensor de estacionamento veicular
Neste guia, você aprenderá a montar um circuito para simular um sensor de estacionamento de um veículo com alerta sonoro.

- Materiais Necessários:
  - Arduino Uno, Protoboard e Jumpers.
  - 1 Resistor de 220 ohms.
  - 1 Sensor Ultrassônico.
  - 1 Buzzer.

Solução Laboratório 01




```arduino
int trig = 8; // declarando o pino trigger SINAL EMITIDO
int echo = 7; // declarando o echo SINAL RECEBIDO
int buzzer = 12;
float distancia;
unsigned long tempo;

void setup()
{
  pinMode(trig, OUTPUT); // trigger é declarado como saída pois emite o sinal
  pinMode(echo,INPUT); // echo é declarado como entrada pois recebe o sinal
  pinMode(buzzer, OUTPUT);
  Serial.begin (9600);
}
```

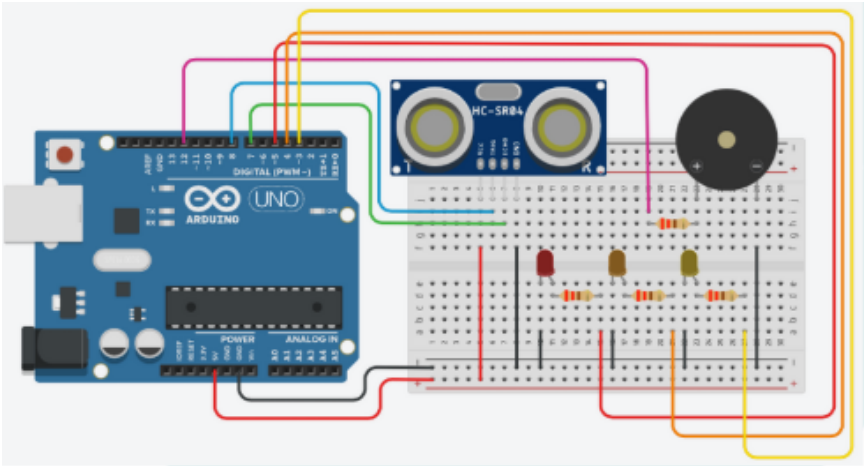
```arduino
void loop()
{
  digitalWrite(trig, HIGH); // envia um sinal de nível alto por 10 microssegundos
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  tempo = pulseIn(echo, HIGH); // obtém em quanto tempo o sinal foi recebido
  //Função para calcular a distandia -> distancia = (tempo voo * velocidade som) / 2;
  distancia = ((tempo * 0.034029) / 2);
  Serial.print(distancia);
  Serial.println(" cm");

  // se a distancia estiver entre 200 cm e 150 cm toque mais tranquilo
  if(distancia <= 200.0 && distancia > 150.0)
  {
    tone(buzzer, 294, 500);
    delay(1000);
  }
  // se estiver entre 150 cm e 100 cm toque um pouco mais corrido
  if(distancia <= 150.0 && distancia > 100.0)
  {
    tone(buzzer, 294, 800);
    delay(500);
  }
  // se estiver entre 100 cm e 50 cm toque bem mais acelerado
  if(distancia <= 100.0 && distancia >= 50.0)
  {
    tone(buzzer, 294, 1000);
    delay(250);
  }
  delay(1000);
}
```

Laboratório 02

Sensor de estacionamento veicular completo sonoro e visual
Neste guia, você aprenderá a montar um circuito para simular um sensor de estacionamento de um veículo com resposta sonora e visual com LEDs e Buzzer.

- Materiais Necessários:
  - Arduino Uno, Protoboard e Jumpers.
  - 3 LEDs (amarelo, laranja e vermelho).
  - 4 Resistores de 220 ohms.
  - 1 Buzzer.
  - 1 Sensor Ultrassônico.

Solução Laboratório 02




```ino
int trig = 8; // declarando o pino trigger SINAL EMITIDO
int echo = 7; // declarando o echo SINAL RECEBIDO
int buzzer = 12;
int led_vermelho = 5;
int led_laranja = 4;
int led_amarelo = 3;
float distancia;
unsigned long tempo;

void setup()
{
  pinMode(trig, OUTPUT); // trigger é declarado como saída pois emite o sinal
  pinMode(echo,INPUT); // echo é declarado como entrada pois recebe o sinal
  pinMode(buzzer, OUTPUT);
  pinMode(led_vermelho, OUTPUT);
  pinMode(led_laranja, OUTPUT);
  pinMode(led_amarelo, OUTPUT);
  Serial.begin (9600);
}

```

```ino
void loop()
{
  digitalWrite(trig, HIGH); // envia um sinal de nível alto por 10 microssegundos
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  tempo = pulseIn(echo, HIGH); // obtém em quanto tempo o sinal foi recebido
  //Função para calcular a distandia -> distancia = (tempo voo * velocidade som) / 2;
  distancia = ((tempo * 0.034029) / 2);
  Serial.print(distancia);
  Serial.println(" cm");

  // se a distancia estiver entre 200 cm e 150 cm toque mais tranquilo
  if(distancia <= 200.0 && distancia > 150.0)
  {
    digitalWrite(led_amarelo, HIGH);
    tone(buzzer, 294, 500);
    delay(1000);
  }
  // se estiver entre 150 cm e 100 cm toque um pouco mais corrido
  if(distancia <= 150.0 && distancia > 100.0)
  {
    digitalWrite(led_amarelo, HIGH);
    digitalWrite(led_laranja, HIGH);
    tone(buzzer, 294, 800);
    delay(580);
  }
  // se estiver entre 100 cm e 50 cm toque bem mais acelerado
  if(distancia <= 100.0 && distancia >= 50.0)
  {
    digitalWrite(led_amarelo, HIGH);
    digitalWrite(led_laranja, HIGH);
    digitalWrite(led_vermelho, HIGH);
    tone(buzzer, 294, 1000);
    delay(250);
  }
  // fora da area de interesse os LEDs permanecem desligados
  digitalWrite(led_amarelo, LOW);
  digitalWrite(led_vermelho, LOW);
  digitalWrite(led_laranja, LOW);
  delay(1000);
}
```

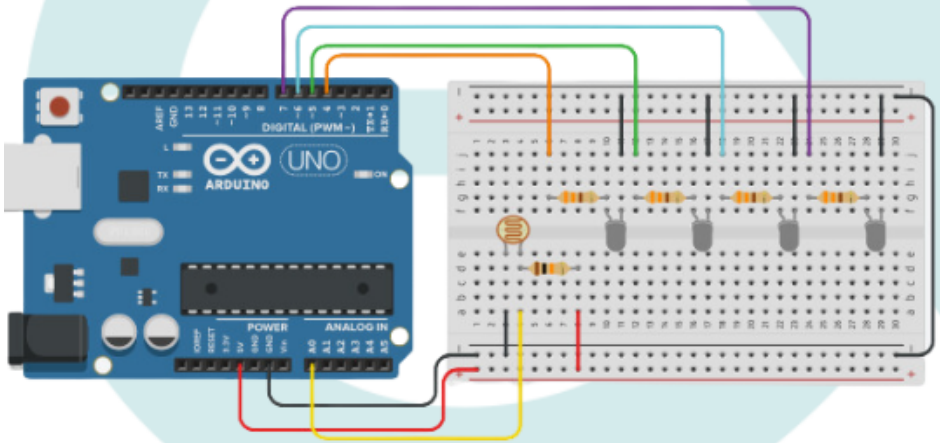
Laboratório 03

Simulação de iluminação pública

Neste guia, você aprenderá a montar um circuito com LEDs e o sensor de luminosidade para economia de energia, assim como na iluminação de ruas.

- Materiais Necessários:
  - Arduino Uno, Protoboard e Jumpers.
  - 4 LEDs (preferencialmente alto brilho).
  - 4 Resistores de 220 ohms.
  - 1 Resistor de 10k ohms.
  - 1 Sensor de Luminosidade.

Solução Laboratório 03




```arduino
// declaração do pino do LDR e variável que armazenara a leitura
int LDR = A0;
int leitura = 0;
int led_1 = 4;
int led_2 = 5;
int led_3 = 6;
int led_4 = 7;

void setup()
{
  pinMode(LDR, INPUT);// como LDR é um sensor ele irá captar informações logo é declarado como ENTRADA
  pinMode(led_1, OUTPUT);
  pinMode(led_2, OUTPUT);
  pinMode(led_3, OUTPUT);
  pinMode(led_4, OUTPUT);
  Serial.begin(9600);
}
```

```arduino
void loop()
{
  leitura = analogRead(LDR);
  // a impressão no monitor serial esta sendo utilizada somente como apoio para identificar se a leitura esta correta
  Serial.print("Valor lido pelo LDR: ");
  Serial.println(leitura);
  // se o valor lido for maior que 500 entendesse que esta esta escuro
  if(leitura > 500)
  {
    liga_iluminacao();
  }
  else // se o valor for menor que 500 entendesse que esta de dia
  {
    desliga_iluminacao();
  }
  delay(100);

}
//funcoes para ligar e desligar a iluminação
void liga_iluminacao()
{
  digitalWrite(led_1, HIGH);
  digitalWrite(led_2, HIGH);
  digitalWrite(led_3, HIGH);
  digitalWrite(led_4, HIGH);
}
void desliga_iluminacao()
{
  digitalWrite(led_1, LOW);
  digitalWrite(led_2, LOW);
  digitalWrite(led_3, LOW);
  digitalWrite(led_4, LOW);
}
```