Ministério da Educação
Universidade Tecnológica Federal do Paraná
Coordenação de Engenharia de Computação
Materiais e Equipamentos Elétricos
Campus Apucarana

Laboratório de Materiais e Equipamentos Elétricos

Roteiro 9: Arduino - Entrada Analógica

1. Objetivos

- Analisar e compreender as portas (pinos) de um Arduino.
- Utilizar a comunicação serial
- Ler dados de um potenciômetro pelo Arduino.
- Instrumentar um sensor

2. Material utilizado

- 5 LEDs
- 5 Resistores de 150 Ohms
- Resistor de 5kOhms
- Potenciômetro de 10k Ohms
- Fotoresistor LDR
- Protoboard
- Arduino UNO

2.1. Entradas Analógicas (Arduino)

Um microcontrolador, como o Arduino, possui portas de entrada analógicas (que irão receber como entrada tensões entre 0 V e 5 V). Essa leitura acontece por meio de um conversor Analógico/Digital (A/D), que transforma o valor analógico do lido (0 a 5 V) em um valor digital inteiro (0 a 1023 no caso do Arduino UNO). A

Figura 1 mostra como as portas de um Arduino são distribuídas. As portas de entrada analógica são marcadas com a **letra A (A0 até A5)**. O arduino irá converter tensões entrre **0 V** e **5 V** como um valor inteiro entre **0 e 1023** (valor digital de 10 bits).

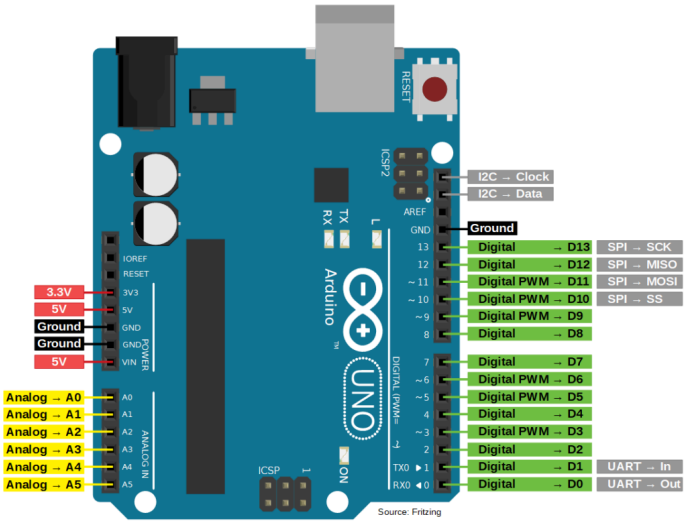


Figura 1: pinos disponíveis no Arduino UNO

2.2. Potenciômetro

Um potenciômetro é um componente eletrônico que permite controlar a resistência elétrica de um circuito de forma variável. Ele é composto por três terminais: o terminal central, chamado de "cursor", e dois terminais externos, conhecidos como "terminais de extremidade" como mostrado na Figura 2. Ele é usado para controlar características como brilho, volume, velocidade e outras variáveis elétricas em uma ampla variedade de aplicações.

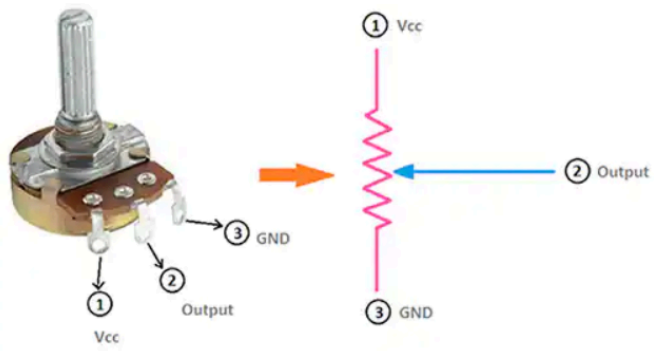


Figura 2: Ligação de um potenciômetro como um divisor de tensão

Ele funciona alterando a resistência entre os terminais de extremidade, enquanto o cursor se move ao longo de um arco de resistência. À medida que o cursor é movido, a resistência elétrica entre o terminal central e um dos terminais de extremidade varia, o que altera a quantidade de corrente elétrica que flui através dele. Ligando Vcc em uma extremidade e GND na outra, temos que o pino do cursor (Output) terá uma tensão que varia entre GND e Vcc.

2.3. Fotoresistor (LDR)

O LDR é a sigla em inglês para Light Dependent Resistor, que traduzido significa resistor dependente de luz. O LDR também é conhecido como fototoresistor, e ele é um tipo de resistor que tem a capacidade de variar a sua resistência em função da intensidade de luz que incide sobre ele. Quando as partículas de luz (fótons) incidem sobre a superfície do sensor, os elétrons que estão no material semicondutor são liberados, dessa forma a condutividade do LDR aumenta e a sua resistência diminui.

Quando um LDR é usado como sensor, ele geralmente é combinado com outros componentes para formar um circuito simples. Um exemplo comum é usar um LDR em conjunto com um resistor fixo e um microcontrolador como mostrado na Figura 3.

O LDR e o resistor fixo são conectados em uma configuração chamada de "divisor de tensão". O ponto médio entre o LDR e o resistor fixo é conectado a um pino de entrada analógica do microcontrolador. O microcontrolador é programado para ler o valor de tensão nesse pino e, a partir desse valor, pode-se determinar a intensidade luminosa detectada pelo LDR.

Quando a luz incide sobre o LDR, sua resistência diminui e a tensão no ponto médio do divisor de tensão aumenta. Isso resulta em uma leitura maior no pino analógico do microcontrolador. Por outro lado, quando a luz é reduzida, a resistência do LDR aumenta e a tensão no ponto médio do divisor diminui, gerando uma leitura menor no pino analógico. Inverter a posição do LDR e do resistor irá inverter essa lógica.

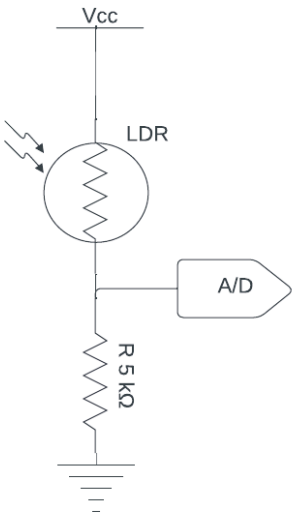


Figura 3: Ligação de um LDR como um divisor de tensão.

2.4. Comunicação Serial

A comunicação serial é uma forma de transmitir dados entre o Arduino e outro dispositivo, como um computador, outro Arduino ou um módulo externo. É uma maneira eficiente de enviar e receber informações em formato de sequência de bytes. Vamos escrever um código básico para enviar a mensagem "Olá, mundo!" para o computador através da comunicação serial:

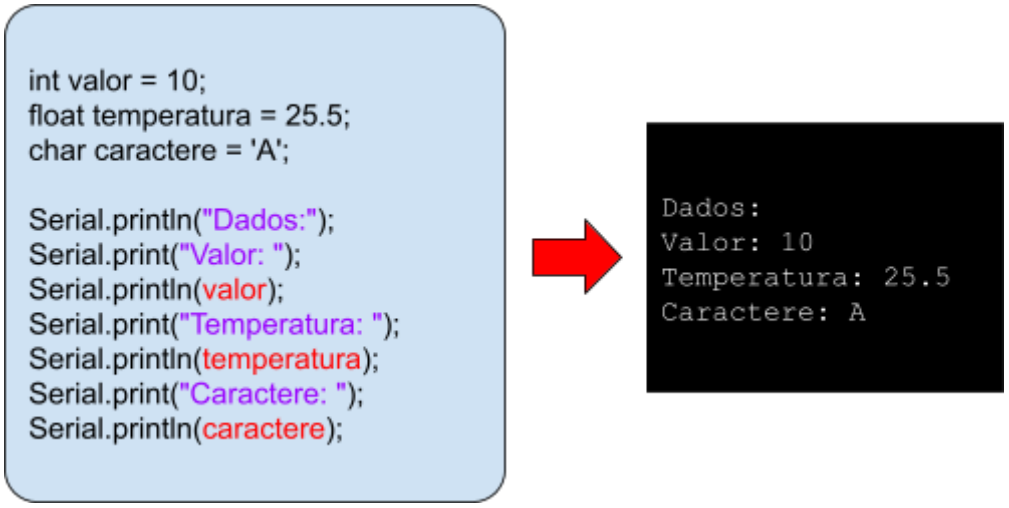
```
void setup() {
  // Inicializa a comunicação serial com uma taxa de 9600 bps
  Serial.begin(9600);
}

void loop() {
  // Envia a mensagem "Olá, mundo!" pela porta serial
  Serial.println("Olá, mundo!");

  // Espera um curto período de tempo antes de repetir o loop
  delay(1000);
}
```

Para monitorar as mensagens enviadas pelo Arduino, vá para a IDE do Arduino e clique no ícone da lupa na barra de ferramentas superior para abrir a "Serial Monitor" (Monitor Serial). Verifique se a velocidade (baud rate) na parte inferior direita da janela é configurada para 9600, a mesma taxa definida no código.

Serial.print() é uma função do Arduino usada para enviar dados pela porta serial. Ela permite imprimir valores de diferentes tipos de dados, como números inteiros, números de ponto flutuante, caracteres e strings. Ao usar **Serial.print()**, os valores são enviados sequencialmente, um após o outro, sem adicionar uma nova linha (quebra de linha) no final. Por outro lado, **Serial.println()** também é uma função do Arduino utilizada para enviar dados pela porta serial, porém, ela adiciona uma quebra de linha (nova linha) após imprimir os valores. Isso significa que cada chamada de **Serial.println()** resulta em uma nova linha na saída.



2.5. Exemplos de configuração

Considere o circuito mostrado na Figura 4 onde desejamos ler o valor do potenciômetro e mostrar o valor lido na serial. Conectando as extremidadas do potenciômetro em GND e 5 V, teremos que o terminal de saída (cursor) irá apresentar uma tensão que varia entre 0 a 5 V conforme giramos o potenciômetro. Essa saída é conectada na entrada analógica do Arduino (A0).

Iniciamos o código declarando uma variável que irá receber o valor da conversão analógico/digital da entrada analógica e outra variável que definirá o pino utilizado para leitura do potenciômetro:

```
const int pinoPotenciometro = A0;
int valorPotenciometro = 0;
```

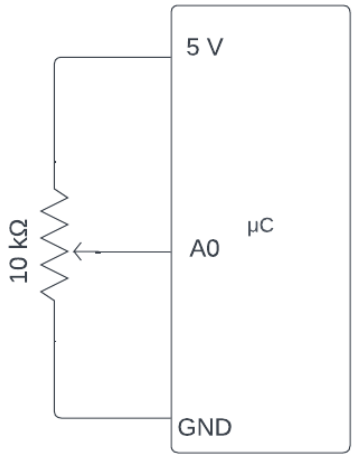


Figura 4: Ligação de um potenciômetro na entrada analógica do Arduino.

Em seguida iremos configurar a serial no laço de **setup()**. Não é necessário configurar a entrada analógica como entrada. Usamos a função Serial.begin para inicializar a comunicação serial com uma taxa de transmissão de 9600 bps (outras taxas também podem ser escolhidas).

```
void setup() {
  Serial.begin(9600);
}
```

No laço principal iremos ler o valor da entrada analógica pelo comando **analogRead()** e imprimir o valor pela serial utilizando **Serial.println()**. Observe que a função de leitura analógica recebe a variável com o pino onde o potenciômetro está conectado:

```
void loop() {
  // Lê o valor do potenciômetro
  valorPotenciometro = analogRead(pinoPotenciometro);

  // Imprime o valor lido na saída serial
  Serial.println(valorPotenciometro);

  // Aguarda um pequeno intervalo
  delay(100);
}
```

O código irá mostrar o valor convertido, de 0 a 1023, no monitor serial. Caso queiramos mostrar de 0 a 100%, é possível fazer uma conversão. A conversão pode ser feita por meio de uma fórmula ou pela função **map()**. A função **map()** é utilizada para converter valores de um intervalo para outro. Nesse caso, queremos mapear o valor lido do potenciômetro (que varia de 0 a 1023) para o intervalo de 0 a 100, correspondente a uma porcentagem. O código abaixo mostra como a função pode ser usada. A função irá mapear valores entre 0 e 1023 para valores entre 0 e 100.

```
const int pinoPotenciometro = A0;
int valorPotenciometro = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  valorPotenciometro = analogRead(pinoPotenciometro);

  // Mapeia o valor lido do A/D para o intervalo de 0 a 100
  int porcentagem = map(valorPotenciometro, 0, 1023, 0, 100);

  Serial.println(porcentagem);

  delay(100);
}
```

3. Parte prática

Considerando os circuitos das Figuras 5 e 6, elabore a prática como se pede nos itens abaixo.

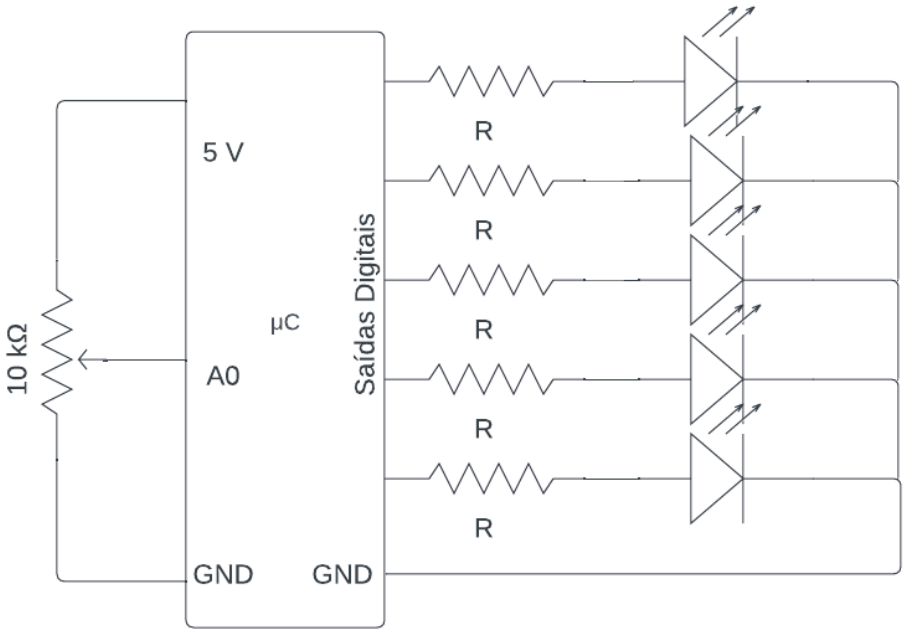


Figura 5: Circuito para a montagem prática (A)

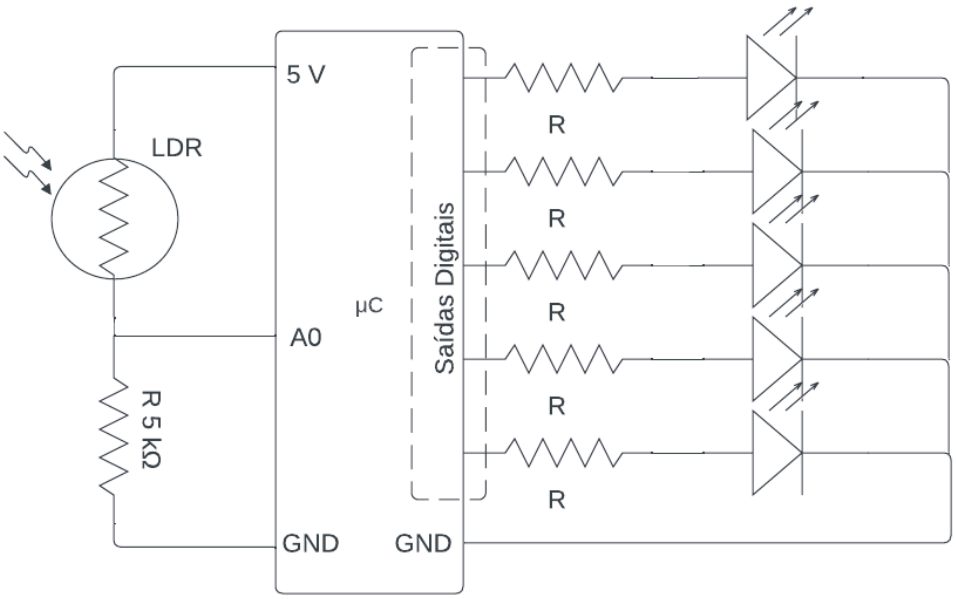


Figura 6: Circuito para a montagem prática (B e C)

A. **Medidor de VU**: Elabore um código para ler o valor do potenciômetro e mostrar ele ao longo dos 5 LEDs. A medida que o potenciômetro for sendo rotacionado, os LEDs irão acender progressivamente. Também mostre os valores do potenciômetro na serial. Esse tipo de visualização de dados de amplitude é comumente chamado de medidor de “volume unitário” (VU).
B. **Sensor de Luminosidade:** Elabore o código de um sensor de luminosidade. Para isso leia o valor do LDR mostre o valor proporcionalmente nos LEDs (como no exercício anterior). Como o sensor não estará definido para toda a faixa do conversor A/D (0 a 1023), faça uma calibração da seguinte forma: realize uma leitura no claro (lanterna do celular) e escuro (tampando o sensor). Verifique os valores mínimos e máximos lidos. Ajuste a faixa utilizando a função map().
C. Desafio: O sensor anterior opera como um detector de luminosidade, porém é muito sensível à variações rápidas. Modifique o código anterior para que o valor mostrado nos LEDs e na serial seja uma média das últimas 30 leituras (feitas a cada 100 ms). Dica: é possível utilizar vetores para evitar a criação de muitas variáveis auxiliares.