



**OFICINA DE ROBÓTICA**  
INVENTAR E RECICLAR PARA EDUCAR  
oficinaderobotica.ufsc.br

# Oficina de Robótica

## Programação Básica em Arduino – Aula 4

Execução:



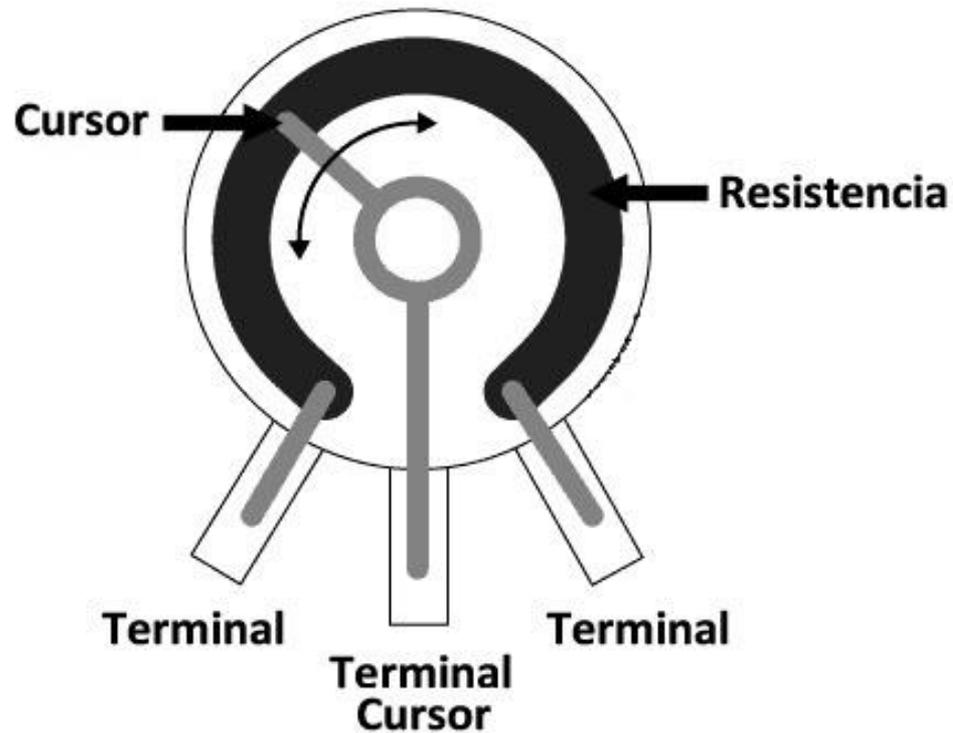
# Potenciômetro

- ▶ Resistor variável



# Resistores Variáveis

## ▶ Potenciômetro



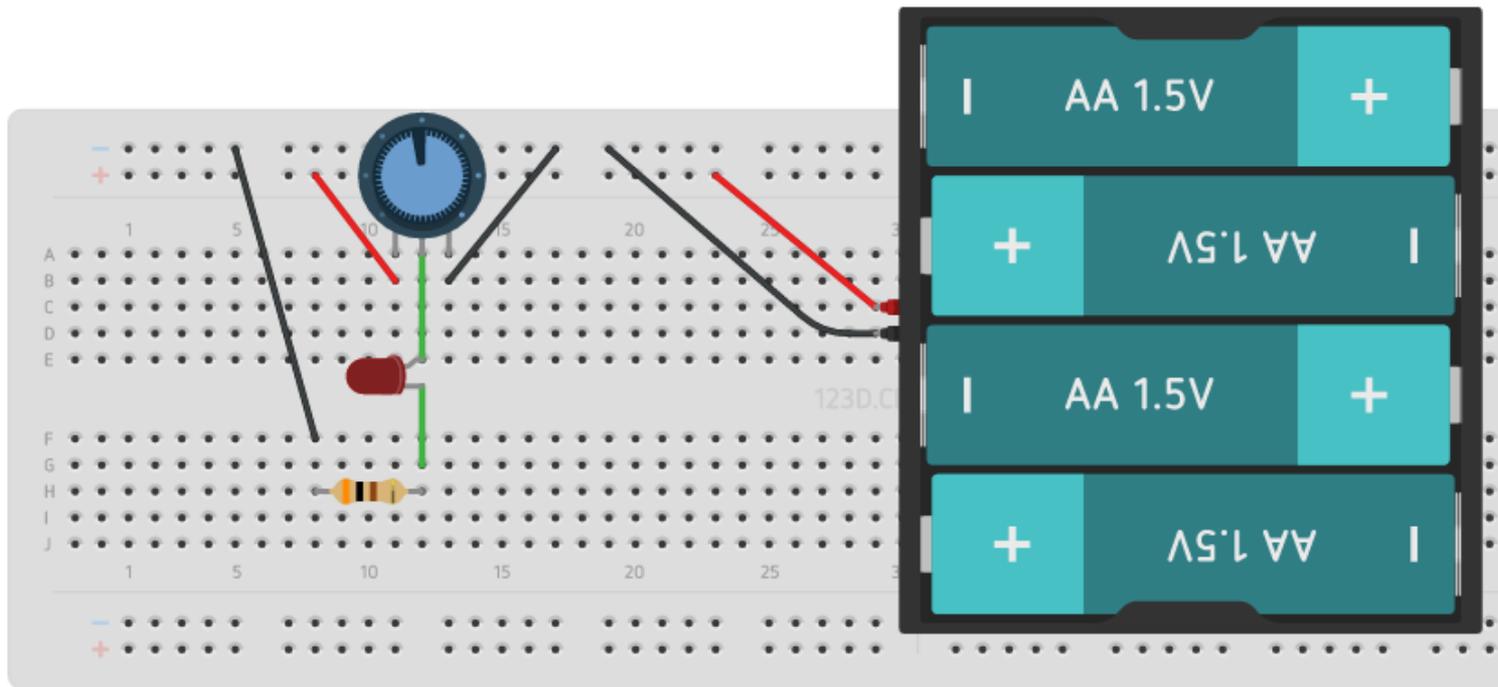
# Exercícios

- ▶ Crie um circuito com potenciômetro controlando a intensidade de um LED.
- ▶ Com o mesmo circuito anterior, controle um buzzer.

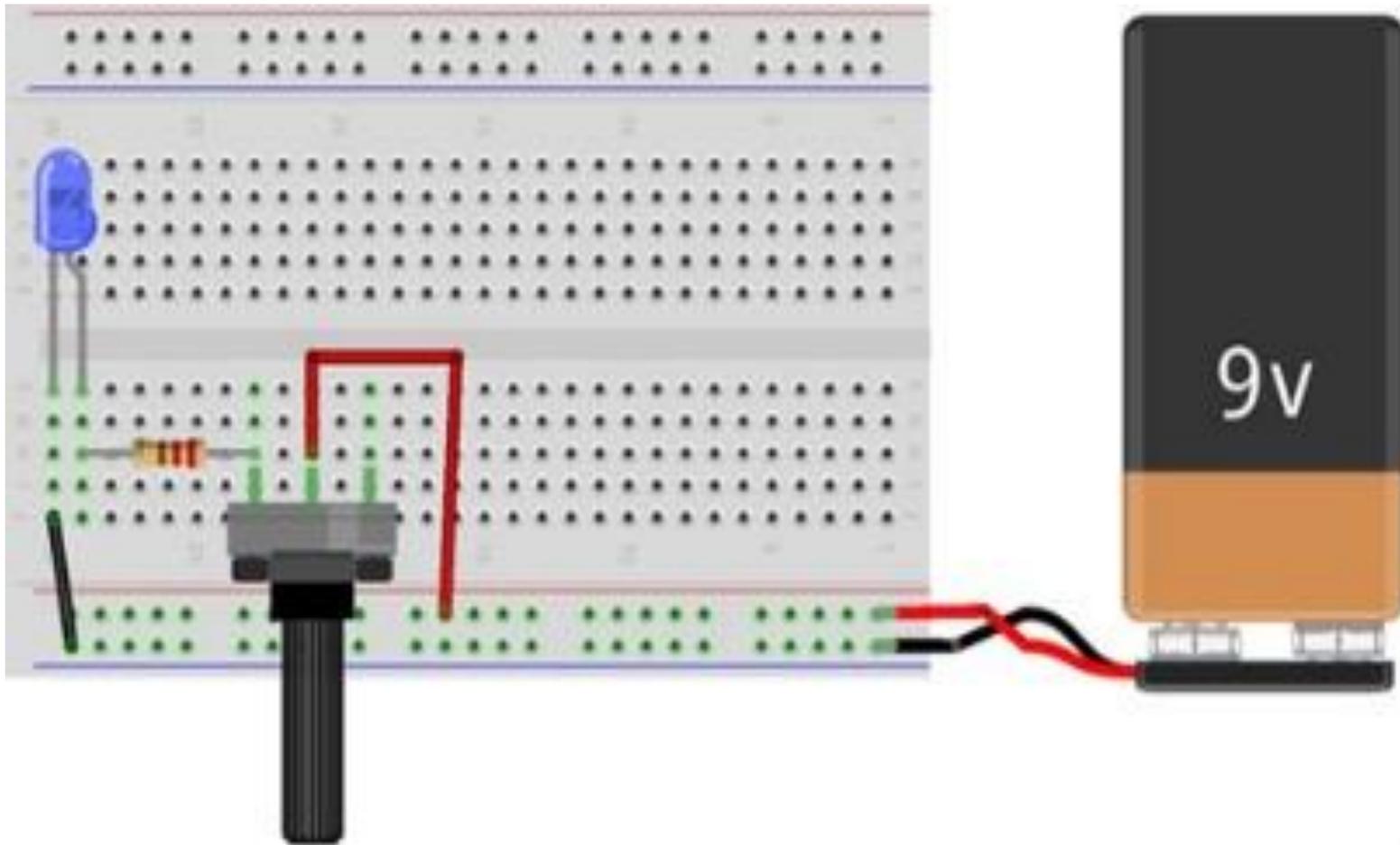


**LARM**

# Potenciômetro



# Montar o circuito abaixo



# Monitor Serial

- ▶ O monitor serial é **utilizado para comunicação entre o Arduino e o computador (PC)**.
- ▶ O monitor serial pode ser aberto no menu *tools* opção *serial monitor*, ou pressionando as teclas **CTRL+SHIFT+M**.
- ▶ As **principais funções** do monitor serial **são**: *begin()*, *read()*, *write()*, *print()*, *println()* e *available()*.



**LARM**

# Monitor Serial

- ▶ Algumas funções bastante usadas:
  - *begin()*: inicializa a comunicação entre o Arduino e um computador;
  - *read()*: recebe caracteres inseridos no monitor serial;
  - *print()*: imprime caracteres no monitor serial;
  - *println()*: imprime caracteres no monitor serial, mas causa uma quebra de linha no final;
  - *available()*: retorna o número de bytes disponíveis no buffer de leitura do monitor serial.



LARM

# Monitor Serial

- ▶ Imprimindo uma mensagem no monitor serial

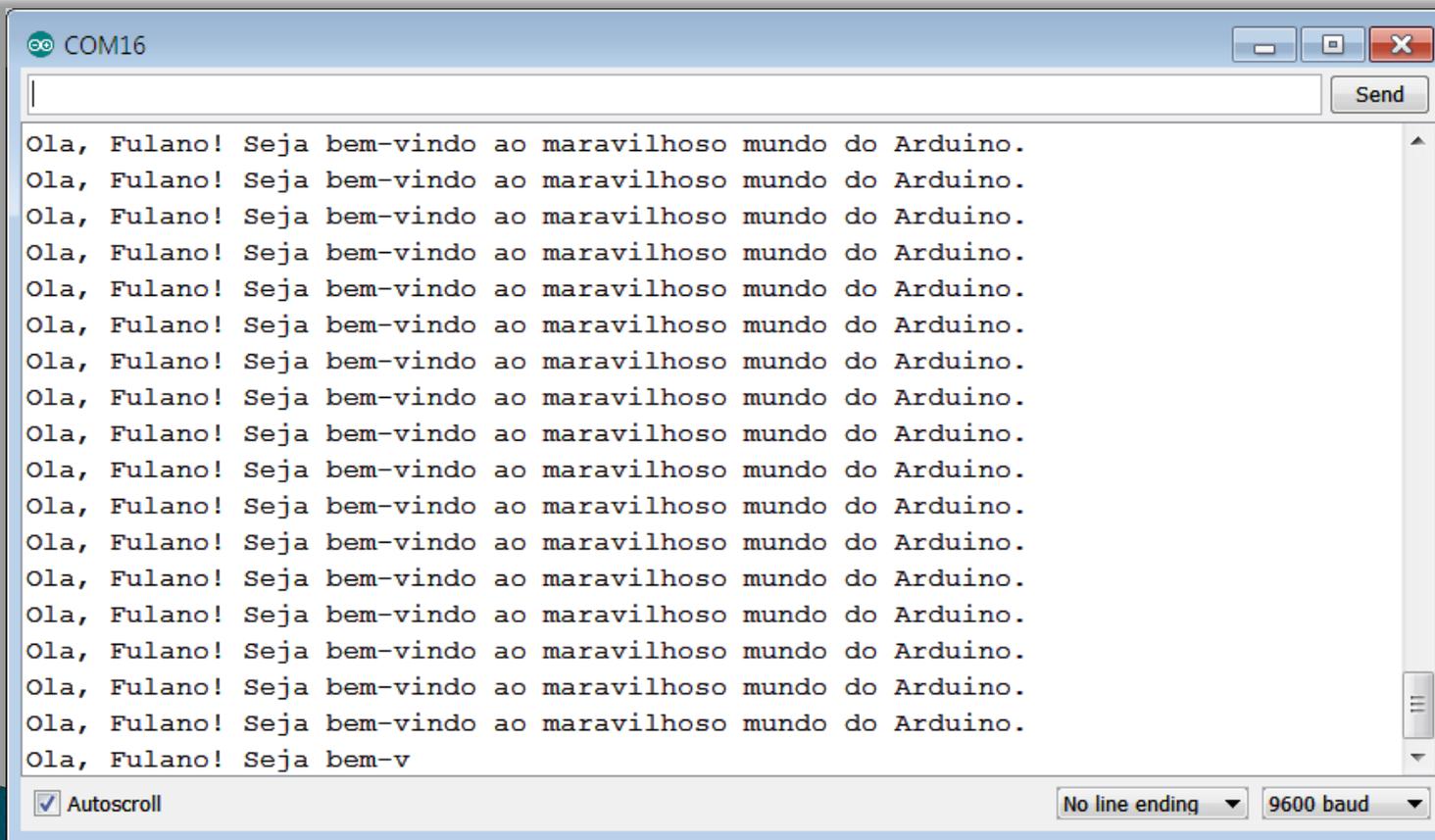
```
monitor_serial $  
void setup ()  
{  
  Serial.begin(9600);  
}  
  
void loop ()  
{  
  Serial.print("Ola, Fulano! Seja bem-vindo ao ");  
  Serial.println("maravilhoso mundo do Arduino.");  
}
```



LARM

# Monitor Serial

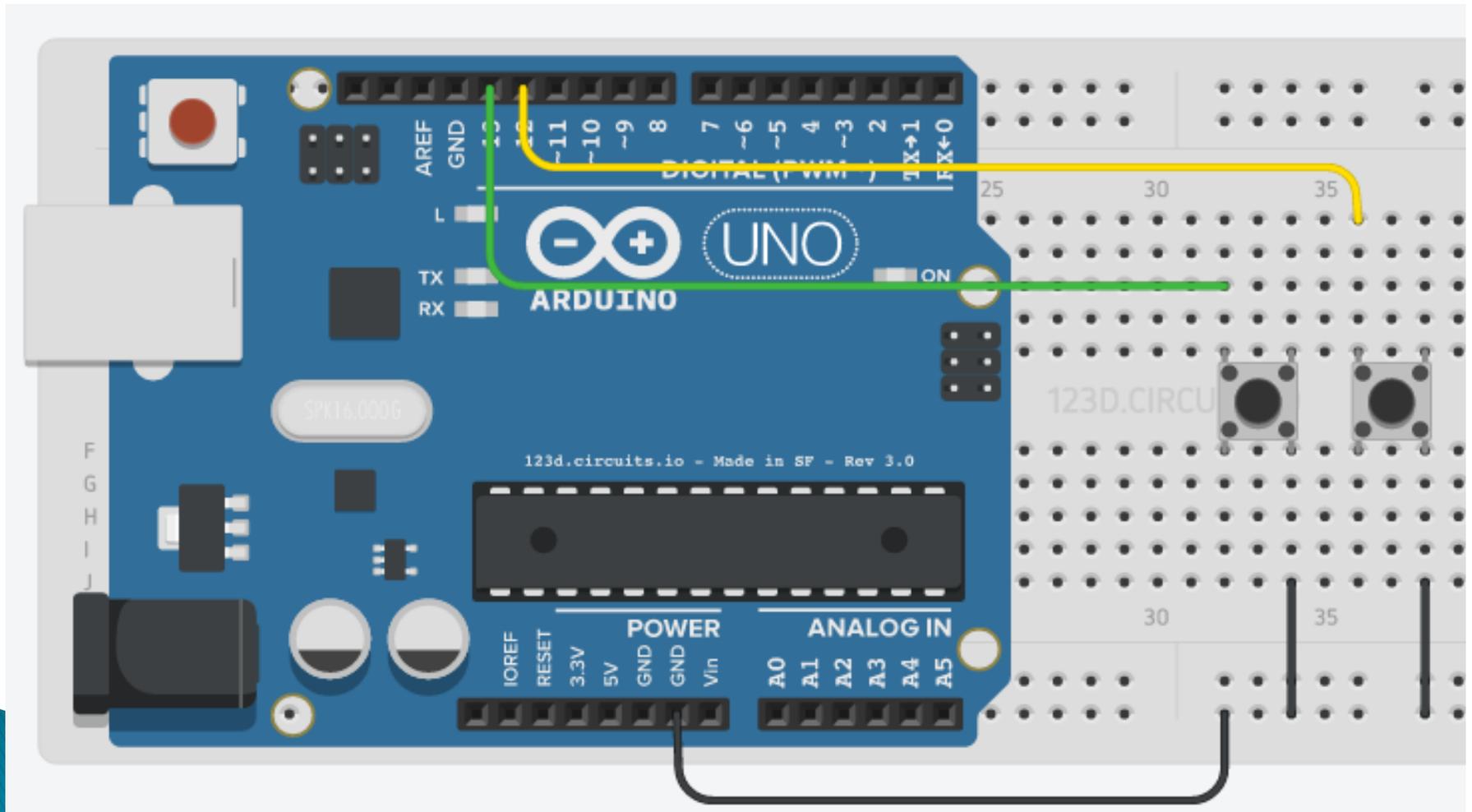
- ▶ Saída no monitor serial



# Exercício

- ▶ Faça um programa que ao apertar um botão, incremente uma variável, e ao clicar em outro botão, diminua o valor da variável, e mostre o valor da variável a cada pressionar de botão.

# Esquemático Exercício



# Entrada Analógica de Dados

- ▶ O **Arduino UNO** possui **6** (seis) **portas analógicas**.
- ▶ O **conversor analógico-digital** do **Arduino** é de **10** (dez) **bits**, logo a **faixa de valores lidos varia de 0 a 1023**.
- ▶ As **portas analógicas** no **Arduino UNO** são **identificadas** como **A0, A1, A2, A3, A4 e A5**. Estas portas **também podem ser identificadas** por **14 (A0), 15 (A1), 16 (A2), 17 (A3), 18 (A4) e 19 (A5)**.



**LARM**

# Entrada Analógica de Dados

- ▶ Na seção “Portas Digitais” vimos que para ler dados em uma porta digital precisávamos usar uma função chamada `digitalRead()`.
- ▶ De forma semelhante, para fazer uma leitura de dados em uma `porta analógica` usaremos `analogRead()`.



LARM

# Entrada Analógica de Dados

- ▶ Lendo dados de um potenciômetro

```
potenciometro
int valor;

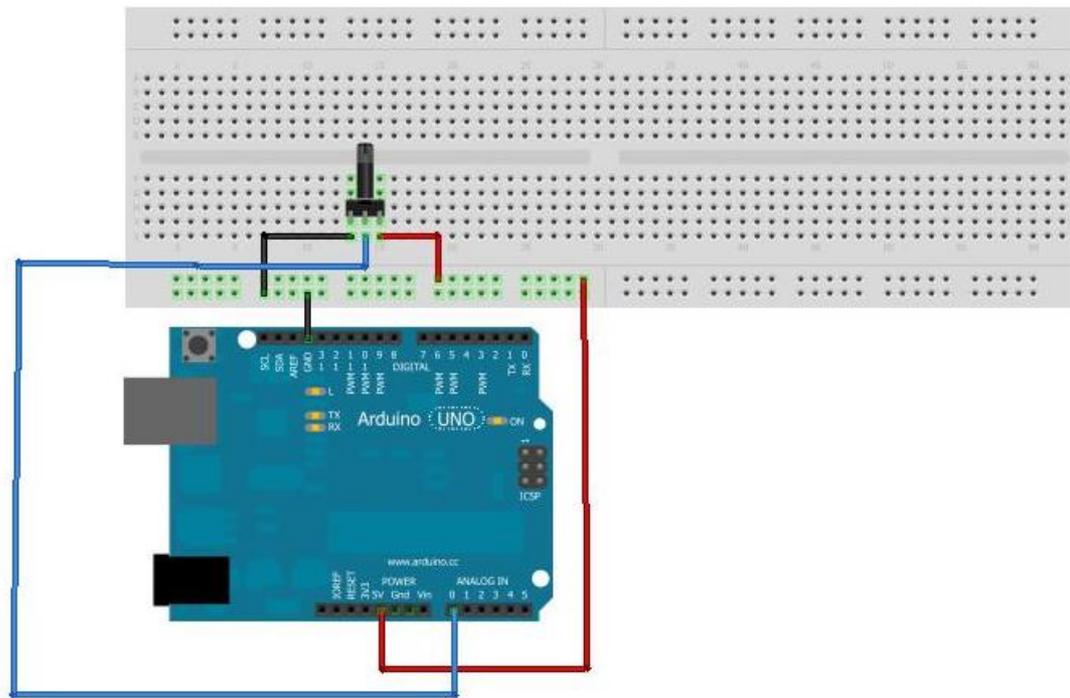
void setup ()
{
  Serial.begin(9600);
}

void loop ()
{
  valor = analogRead(A0);
  Serial.println(valor);
}
```



# Entrada Analógica de Dados

- ▶ Lendo dados de um potenciômetro



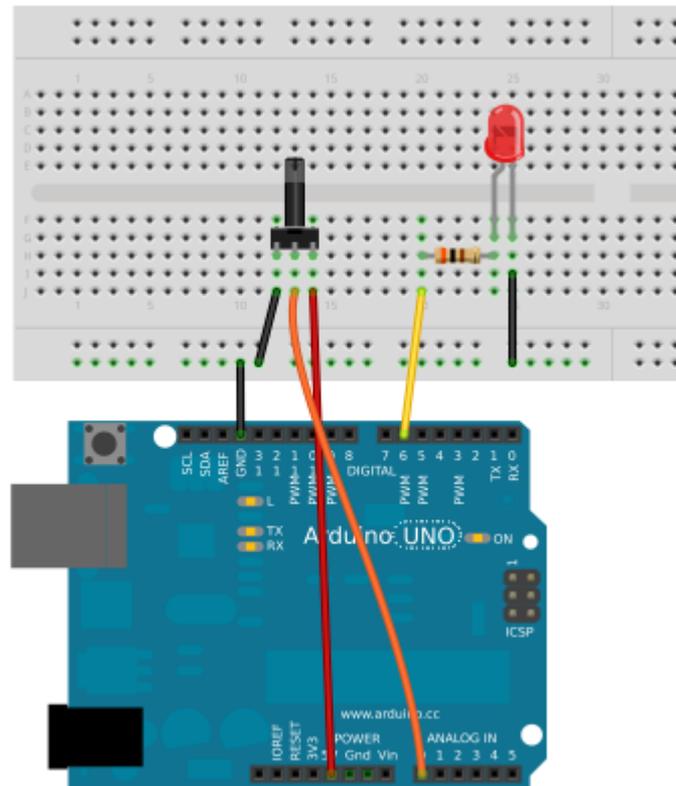
# Entrada Analógica de Dados

## ▶ Exercício:

- Fazer um programa que leia uma entrada analógica de um potenciômetro, e acenda um led caso a leitura for maior que 500, e apague o mesmo led caso for menor.

# Entrada Analógica de Dados

- ▶ Lendo dados de um potenciômetro e acionando um LED



# Entrada Analógica de Dados

- ▶ Lendo dados de um potenciômetro e acionando um LED

```
led_pot
#define POT A0
#define LED 6
int valor;
void setup()
{
  pinMode(LED, OUTPUT);
}
void loop()
{
  valor = analogRead(POT);
  digitalWrite(LED, HIGH);
  delay(valor);
  digitalWrite(LED, LOW);
  delay(valor);
}
```



LARM

# Entrada Analógica de Dados

- ▶ Mapeando valores
  - Algumas vezes precisamos **alterar valores** que se encontram dentro de uma determinada **faixa**, de modo a obter um **novo valor**, proporcional ou inversamente proporcional ao primeiro, e que se enquadre em uma **nova faixa de valores**.
  - A biblioteca do Arduino possui uma função chamada **map()**, que realiza essa tarefa.



**LARM**

# Entrada Analógica de Dados

## ▶ Mapeando valores

- `novo_valor = map(valor, min_in, max_in, min_out, max_out);`

Onde:

- `novo_valor` recebe o valor já modificado pela função `map()`;
- `valor` é o dado a ser alterado;
- `min_in` é o menor valor da faixa de entrada;
- `max_in` é o maior valor da faixa de entrada;
- `min_out` é o menor valor da faixa de saída;
- `max_out` é o maior valor da faixa de saída.



LARM

# Entrada Analógica de Dados

## ▶ Mapeando valores

```
map
#define POT A0
int valor;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  valor = analogRead(POT);
  Serial.print("Valor lido: ");
  Serial.println(valor);
  valor = map(valor, 0, 1023, 0, 100);
  Serial.print("Valor mapeado: ");
  Serial.println(valor);
  Serial.println();
  delay(400);
}
```



LARM

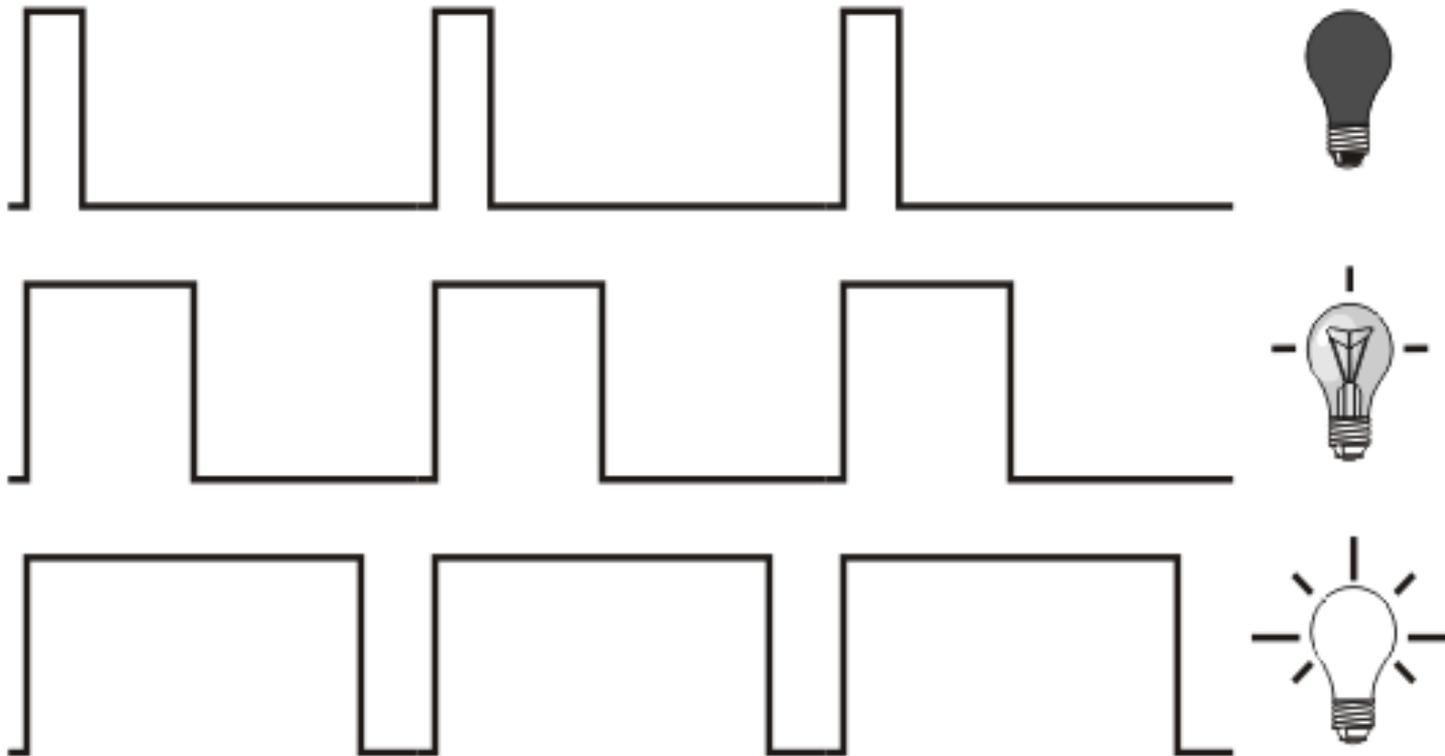
# Pulse Width Modulation (PWM)

- ▶ A **Modulação por Largura de Pulso** (Pulse Width Modulation – PWM) é uma técnica que nos permite controlar a quantidade de energia enviada para uma saída digital.
- ▶ Essa modulação é feita definindo-se um **ciclo de trabalho** que determina a **fração de tempo que o sinal fica no estado ativo**.



**LARM**

# Pulse Width Modulation (PWM)



Extraído de *Teach Yourself PIC Microcontrollers for Absolute Beginners* – M. Amer Iqbal Qureshi, 2006



LARM

# Pulse Width Modulation (PWM)

- ▶ O Arduino UNO possui 6 (seis) portas PWM – 3, 5, 6, 9, 10 e 11.
- ▶ O sinal PWM pode variar de 0 a 255 e para ativá-lo basta usar a seguinte instrução em uma das portas PWM:
  - `analogWrite(pin, valor);`
- ▶ Note que as portas PWM são todas digitais, porém o sinal é modulado “como se fosse” um sinal analógico.



LARM

# Pulse Width Modulation (PWM)

- ▶ **Exemplo:** Usando o PWM para controlar a intensidade de luz emitida por um LED.

```
PWM
#define LED 3

void setup()
{
    pinMode(LED, OUTPUT);
}

void loop()
{
    for (int i = 0; i <= 255; i++) {
        analogWrite(LED, i);
        delay(30);
    }
}
```



LARM

# Exercícios

- ▶ Desenvolva um sistema de controle de intensidade de um buzzer.
- ▶ Para o problema anterior adicione um controle de um LED, que deve ter sua intensidade luminosa, diretamente proporcional ao som do buzzer.
- ▶ Adicione nesse problema uma mensagem em tela, indicando a intensidade do buzzer e do LED.



**LARM**

# Exercícios

- ▶ Desenvolva um sistema para jogos de pergunta e resposta. Esse sistema deve ter 2 botões, um LED para cada botão e um buzzer. No momento em que um botão for clicado, o seu LED correspondente deve ligar e o BUZZER deve apitar. Se isso ocorrer o LED oponente não pode ser acionado.
- ▶ O LED e buzzer devem parar no momento em que o botão for solto.

