

Leitura de valores

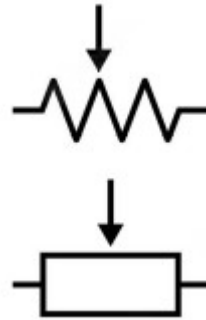
Leitura de valores analógicos

- algum valor entre '0' e '1'
- uma faixa de valores entre um mínimo e um máximo

O potenciômetro



Aspecto
físico



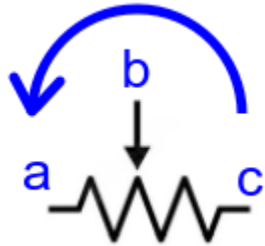
Símbolos
usuais

Trata-se de um resistor variável, ou ajustável.

O terminal 'do meio', corresponde ao ajuste. A resistência será ajustada entre um ponto e outro à medida em que eixo é deslocado.

Desta forma, poderemos ter, entre o terminal mediano e uma extremidade, valores entre '0' e o máximo (que é o valor do potenciômetro, por exemplo, 1k ohm).

Valores proporcionais à posição

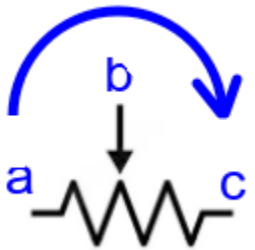


Quando o cursor for levado totalmente em direção a esta extremidade, a resistência será nula (0 ohm).

Neste caso, entre 'a' e 'b' teremos 0 ohm

Quando o cursor for levado totalmente em direção a esta extremidade, a resistência será máxima (valor do potenciômetro).

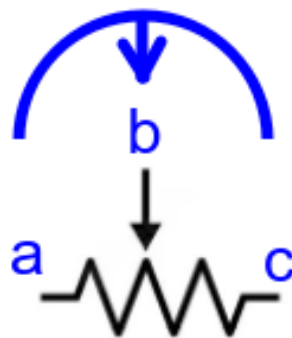
Neste caso, entre 'b' e 'c' teremos 0 ohm



Valores proporcionais à posição

Quando o cursor for levado exatamente ao meio do curso, a resistência será a metade do valor do potenciômetro. Da mesma forma, para valores de deslocamento intermediários, terá valores proporcionais.

Aqui teremos entre 'a' e 'b' metade do valor do potenciômetro



Aqui teremos entre 'b' e 'c' metade do valor do potenciômetro

Entre 'a' e 'c' sempre teremos a resistência total do potenciômetro

Divisor de tensão

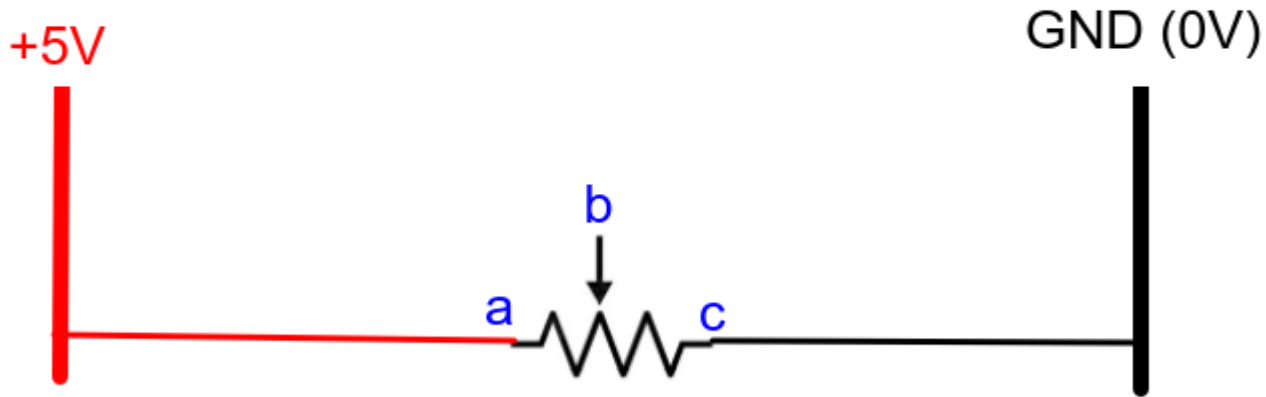
- Podemos utilizar um potenciômetro como divisor de tensão:
 - por exemplo, podemos ligar uma extremidade no +5V e outra no GND (0V); no cursor, em função de sua posição, obteremos tensões proporcionais, entre o mínimo (0V) e o máximo (5V, para este exemplo)

Divisor de tensão

Caso o cursor 'b' esteja em uma posição 20% em relação à extremidade 'a', teremos:

-> entre 'a' e 'b', 1V (20% de 5V = 1V)

-> entre 'b' e 'c', 4V (80% de 5V = 4V)

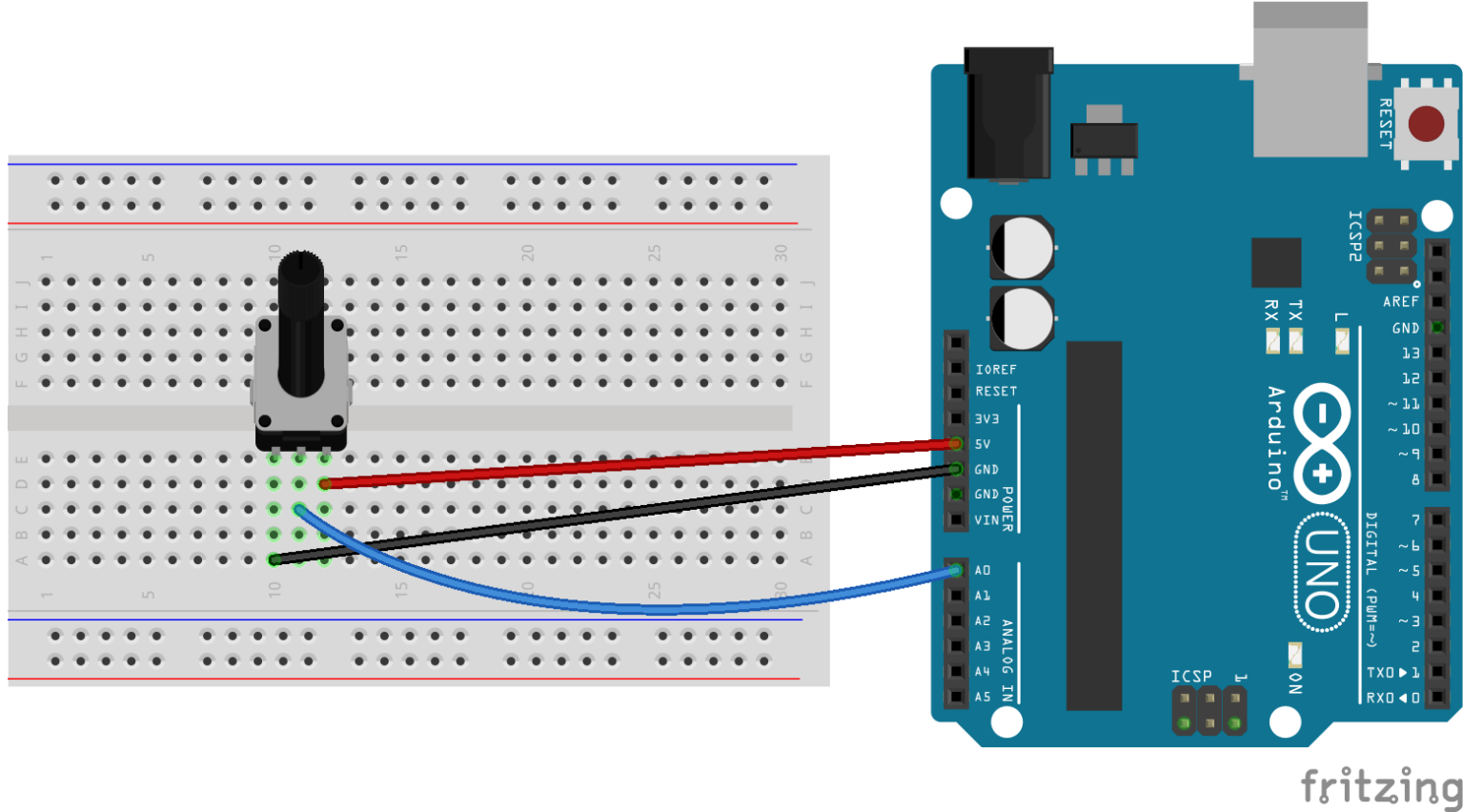


Observe que a tensão obtida depende do ponto de referência utilizado: 'b' com 'a' ou 'b' com 'c'

Vamos praticar



Monte um circuito como abaixo



fritzing

Código para leitura do valor

```
1 #define potenciometro A0
2
3 void setup() {
4     Serial.begin(9600);
5     pinMode(potenciometro, INPUT);
6 }
7
8 void loop() {
9     Serial.println(analogRead(potenciometro));
10 }
```

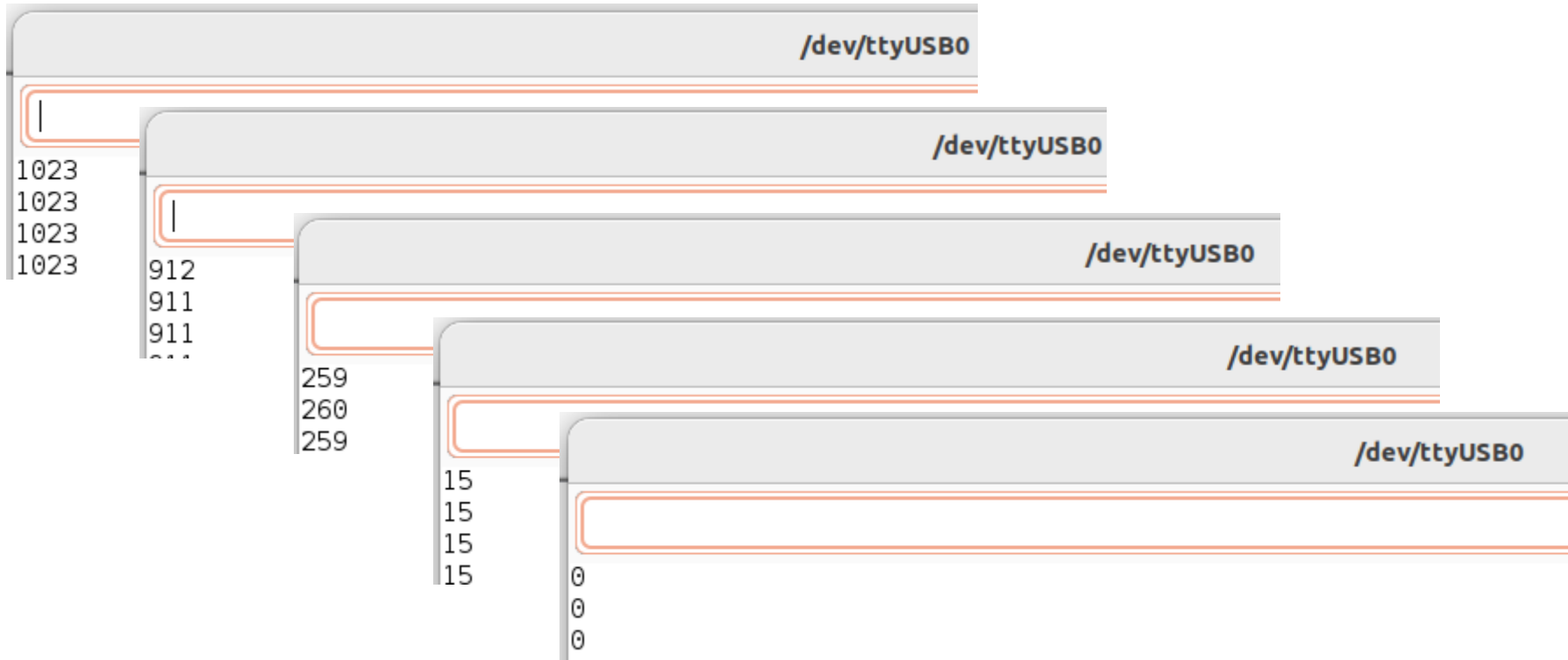
Monitor serial

Ligue o monitor serial: Ferramentas/ Monitor serial



Monitor serial

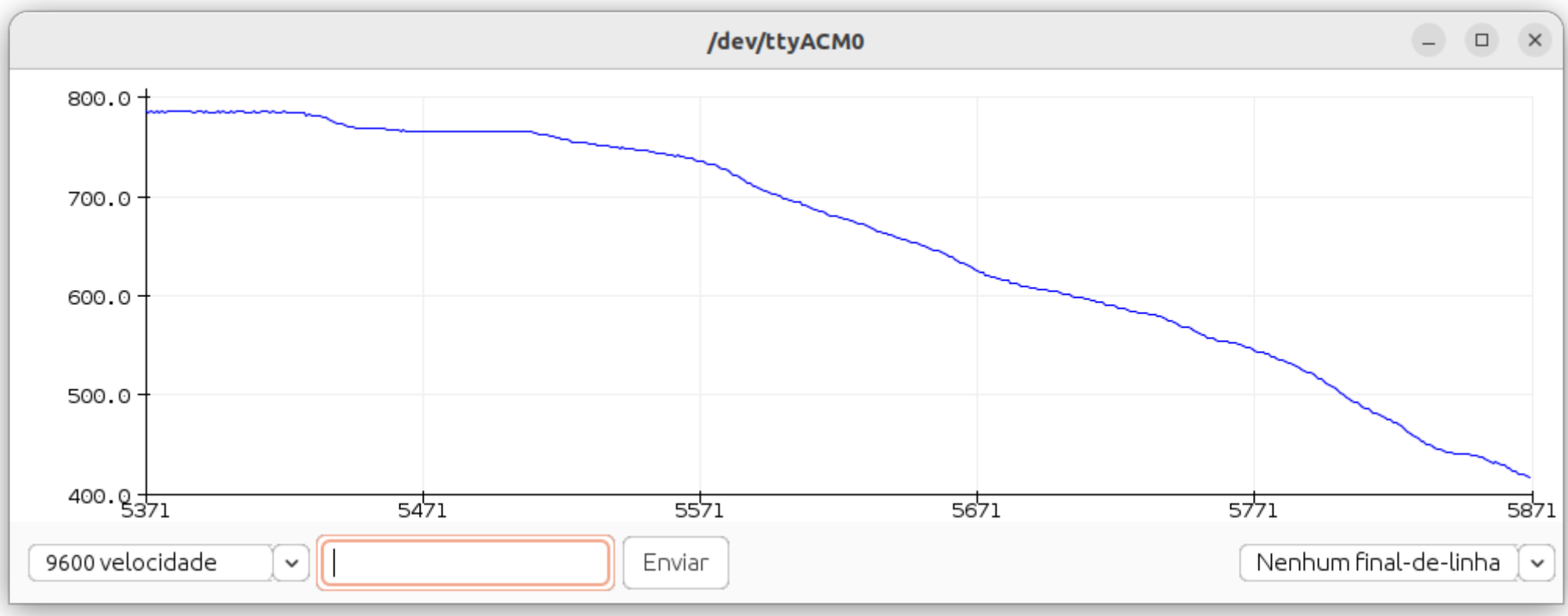
Ajuste o potenciômetro e note a variação (que deverá ir entre 0 e 1023)



Feche a tela do monitor serial.

Ligue o Plotter Serial (Ferramentas/ Plotter
Serial)

Ajuste o potenciômetro para atualizar o gráfico



Comentários

- O valor apresentado estará entre 0 e 1023, na sequência veremos o por quê
- O Arduino está constantemente lendo e apresentando o resultado; quando você varia a posição do potenciômetro o valor resultante da leitura é atualizado
- Vamos entender o código →

Entendendo o código

```
#define potenciometro A0
```

```
void setup() {  
    Serial.begin(9600);  
    pinMode(potenciometro, INPUT);  
}
```

```
void loop() {  
    Serial.println(analogRead(potenciometro));  
}
```


Entendendo o código

```
1 #define potenciometro A0
2
3 void
4   S
5   p
6 }
7
8 void
9   S
10 }
```

Declara uma constante chamada 'potenciometro', dando para ela o valor 'A0' (o qual corresponde à entrada analógica A0 do Arduino, que foi utilizada na conexão de hardware sugerida – poderia ser uma das outras...)

Entendendo o código

```
1 #define potenciometro A0
2
3 void setup() {
4     Serial.begin(9600);
5     pinMode(potenciometro, INPUT);
6 }
7
8 void loop() {
9     Serial.println(analogRead(potenciometro));
10 }
```

Funções básicas:
setup() e loop()

Funções básicas: setup() e loop()

Entendendo o código

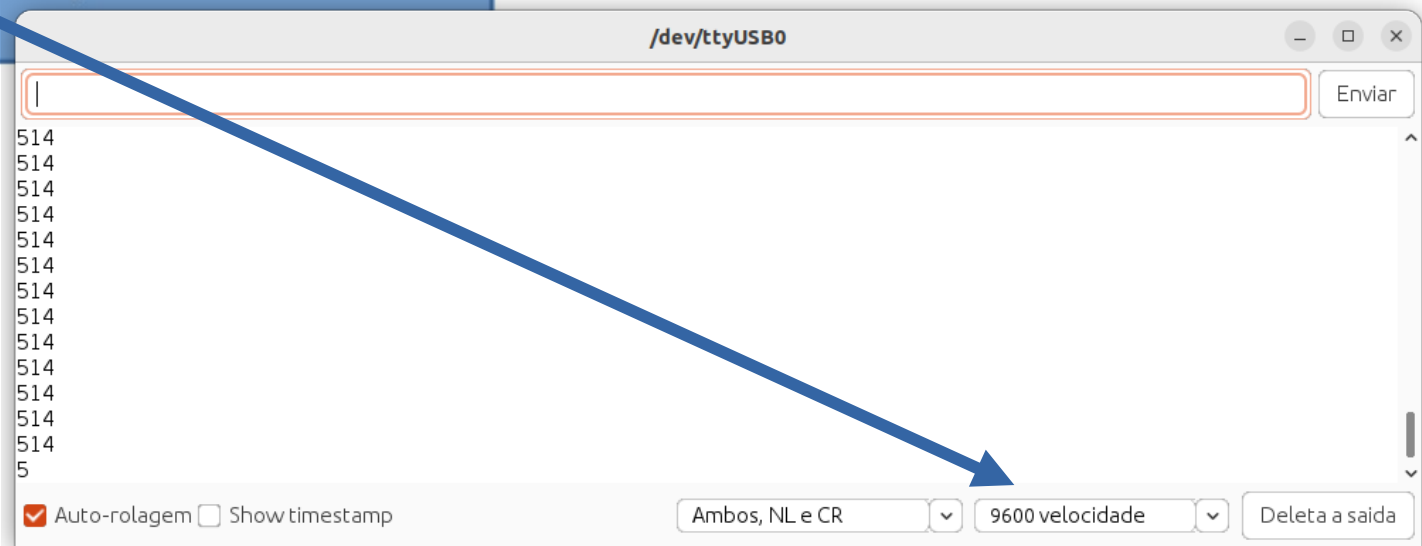
```
1 #define potenciometro A0
2
3 void setup() {
4     Serial.begin(9600);
5     pinMode(potenciometro, INPUT);
6 }
7
8 void
9 Ser
10 }
```

Inicializa o Monitor Serial, com uma velocidade de 9600bps

Entendendo o código

```
1 #define potenciometro A0
2
3 void setup() {
4   Serial.begin(9600);
5   pinMode(potenciometro, INPUT);
6 }
7
8 void loop() {
9   Ser
10 }
```

Inicializa o Monitor Serial, com uma velocidade de 9600bps



Entendendo o código

```
1 #define pot 0  
2  
3 void setup() {  
4   Serial.begin(9600);  
5   pinMode(potenciometro, INPUT);  
6 }  
7  
8 void loop() {  
9   Serial.println(analogRead(potenciometro));  
10 }
```

Coloca a porta analógica '0' (A0, identificada pela constante 'potenciometro') em modo de entrada (INPUT)

Entendendo o código

```
1 #defi
2
3 void
4   Ser
5   pir
6 }
7
8 void
9   Serial.println(analogRead(potenciometro));
10 }
```

Escreve, pulando uma linha (`print` – escreve, `ln` – pula uma linha → `println`), na saída serial (`Serial`), o valor analógico que é lido (`analogRead`) na porta identificada como `potenciometro`.

Lembre-se de que o Arduino executa o último código sempre que houver energia.

Desta forma, se você fechar a janela do monitor serial não quer dizer que não estará sendo realizada a leitura e a escrita dos valores relativos à posição do potenciômetro. Você só não estará vendo.

O Arduino estará trabalhando enquanto houver energia...

Por quê entre 0 e 1023?

- Existem infinitos valores analógicos
- Por exemplo, entre 0 e 1 inteiros, temos:
 - 0,0000000000000001
 - 0,000053
 - 0,008
 - 0,03
 - ...

Por quê entre 0 e 1023?

- Os circuitos eletrônicos digitais necessitam realizar uma conversão, de valor analógico para valor digital. Daí podem utilizá-los.
- Obviamente, por serem os valores analógicos infinitos, nem todos poderão ser convertidos.
 - Haverá limites para a conversão.

Por quê entre 0 e 1023?

- O circuito interno do Arduino UNO, que estamos utilizando em nossos experimentos, trabalha com conversores que utilizam **10 bits** de precisão.
- 10 bits significa a possibilidade de termos uma combinação de 'zeros e uns' entre
0000000000 e 1111111111

Por quê entre 0 e 1023?

- Na prática, utilizamos potências de '2', por ser a numeração binária (com '2' valores possíveis: '0' e '1')

- Assim:

$2^0 = 0$, nenhuma combinação

$2^1 = 2$, duas combinações (0 e 1)

$2^2 = 4$, quatro combinações (00, 01, 10, 11)

$2^3 = 8$, oito combinações (000, 001, 010, 011,
100, 101, 110, 111)

Por quê entre 0 e 1023?

- Das potências de 2, teremos:

$2^{\text{elevado a 10 bits}}$

$2^{10} = 1024$ combinações diferentes

de 0000000000	- que é nosso número 0
até 1111111111	- que é nosso número 1023

(1024 valores, iniciando em 0, vai de 0 a 1023)

E 'quanto' vale cada valor?

- Para os valores possíveis (entre 0 e 1023), estamos aplicando uma tensão elétrica na entrada analógica, obtida por meio de um divisor de tensão formado por um potenciômetro.
- Sabemos que entre o cursor e as extremidades (às quais aplicamos GND e +5V), teremos que ter tensões proporcionais ao deslocamento do eixo.

E 'quanto' vale cada valor?

- Cada um dos valores apresentados (entre 0 e 1023) corresponde a uma pequena fração de tensão.
- Vem à tona o conceito de **resolução**.
 - De quanto é a diferença entre os valores lidos?

Resolução

- A resolução da medida pode ser calculada por:

$$\text{Resolução} = \frac{\text{Tensão de referência}}{\text{Número de combinações (bits)}}$$

- O que, para nosso caso, será:

$$\text{Resolução} = \frac{5\text{ V}}{1024} = 0,004882813\text{ V}$$

- Ou seja, teremos, para cada valor indicado, uma fração correspondente a cerca de 4,88mV

Convertendo...

```
1 #define potenciometro A0
2
3 void setup() {
4     Serial.begin(9600);
5     pinMode(potenciometro, INPUT);
6 }
7
8 void loop() {
9     Serial.println(analogRead(potenciometro)*5/1024);
10 }
```

Estamos multiplicando o valor lido pela relação $5V / 1024$ bits antes de mostrá-lo. Compile e envie a modificação ao Arduino, ligue o monitor serial e note a diferença.

Estão sendo utilizados comandos ‘sempre na mesma linha’, para facilitar as explicações iniciais e testes.

Isto não é o mais adequado por tornar o código um pouco confuso, difícil de ler.

No próximo exemplo vamos melhorar a legibilidade do código.

Apresentar o valor lido

```
1 #define potenciometro A0
2
3 void setup() {
4     Serial.begin(9600);
5     pinMode(potenciometro, INPUT);
6 }
7
8 void loop() {
9     int valorLido = 0;
10    float valorTensao = 0.0;
11    valorLido = analogRead(potenciometro);
12    valorTensao = ((float)valorLido) * 5 / 1024;
13    Serial.println(valorTensao);
14 }
```

Apresentar o valor lido

```
1 #define potenciometro A0
2
3 void setup() {
4   Serial.begin(9600);
5   pinMode(potenciometro, INPUT);
6 }
7
8 void loop() {
9   int valorLido = 0;
10  float valorTensao = 0.0;
11  valorLido = analogRead(potenciometro);
12  valorTensao = ((float)valorLido) * 5 / 1024;
13  Serial.println(valorTensao);
14 }
```

Na linha 9 foi declarada uma variável (`valorLido`) do tipo inteira (`int`) – ela poderá trabalhar com números inteiros. A variável foi inicializada com o valor 0.

Na linha 10 foi declarada uma variável (`valorTensao`) do tipo `float`, que trabalhará com números em ponto flutuante (decimais). Ela foi inicializada com o valor 0.0.

Na linha 11 a variável `valorLido` recebe o valor da leitura da porta `potenciometro` (que é um inteiro entre 0 e 1023).

Na linha 12 a variável `valorTensao` recebe o resultado da conversão de `valorLido` para `float`, multiplicado por $5/1024$, que é a nossa resolução.

Melhorando a apresentação

```
- 1 #define potenciometro A0
  2
  3 void setup() {
  4     Serial.begin(9600);
  5     pinMode(potenciometro, INPUT);
  6 }
  7
  8 void loop() {
  9     int valorLido = 0;
 10     float valorTensao = 0.0;
 11     valorLido = analogRead(potenciometro);
 12     valorTensao = ((float)valorLido) * 5 / 1024;
 13     Serial.print("O valor lido foi: ");
 14     Serial.print(valorLido);
 15     Serial.print(" que corresponde a : ");
 16     Serial.print(valorTensao);
 17     Serial.println(" Volts\n");
 18 }
```

Melhorando a apresentação

```
1 #define potenciometro A0
2
3 void setup() {
4   Serial.begin(9600);
5   pinMode(potenciometro, INPUT);
6 }
7
8 void loop() {
9   int valorLido = 0;
10  float valorTensao = 0.0;
11  valorLido = analogRead(potenciometro);
12  valorTensao = ((float)valorLido) * 5 / 1024;
13  Serial.print("O valor lido foi: ");
14  Serial.print(valorLido);
15  Serial.print(" que corresponde a : ");
16  Serial.print(valorTensao);
17  Serial.println(" Volts\n");
18 }
```

Só print – não pula linha

println – pula linha

" \n " – pula linha
(newline)

Entre 0 e 4

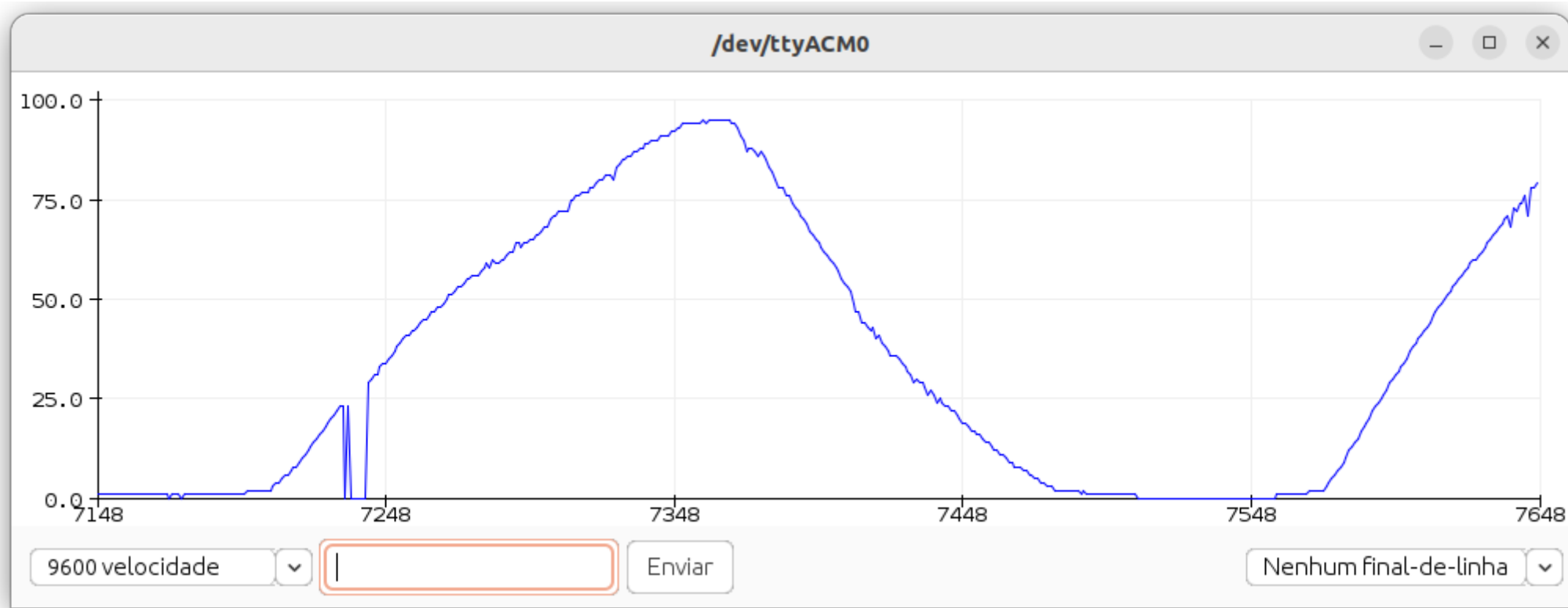
- Em função do tipo de dados utilizado para a conversão 'automática' utilizada, os valores não apresentação casas decimais... e, portanto, ficarão entre 0 e 4. Veremos posteriormente como mostrar os valores com casas decimais.
- Mas, antes, vamos usar uma outra função.

Função 'map'

```
1 #define potenciometro A0
2
3 void setup() {
4   Serial.begin(9600);
5   pinMode(potenciometro, INPUT);
6 }
7
8 void loop() {
9   Serial.println(map(analogRead(potenciometro), 0, 1023, 0, 100));
10 }
```

A função `map` recebe os valores originais (mínimo e máximo, no nosso caso 0 e 1023) e os novos valores para os quais os lidos serão transformados (no exemplo, 0 e 100)

Usando map – ajuste o potenciômetro



Teste diferentes valores para a função 'map'

Números

- Conforme a aplicação, talvez seja necessário trabalhar com números em bases diferentes da decimal
- O pequeno programa a seguir exibe um conjunto de números em diferentes bases de numeração

```
1 int numero;  
2  
3 void setup() {  
4   Serial.begin(9600);  
5  
6 }  
7  
8 void loop() {  
9  
10  for (numero = 0; numero < 33; numero++) {  
11    Serial.print(numero, DEC);  
12    Serial.print("\t");  
13    Serial.print(numero, BIN);  
14    Serial.print("\t");  
15    Serial.println(numero, HEX);  
16    delay(500);  
17  }  
18  while(1){  
19  
20  }  
21 }
```

**Para ver o
resultado da
execução, clique
Ferramentas /
Monitor Serial – e,
SE precisar, ajuste
a velocidade de
comunicação para
9600 bps**

```
1 int numero;  
2  
3 void setup() {  
4   Serial.begin(9600);  
5  
6 }  
7  
8 void loop() {  
9  
10  for (numero = 0; numero < 33; numero++) {  
11    Serial.print(numero, DEC);  
12    Serial.print("\t");  
13    Serial.print(numero, BIN);  
14    Serial.print("\t");  
15    Serial.println(numero, HEX);  
16    delay(500);  
17  }  
18  while(1){  
19  
20  }  
21 }
```

DEC - Números decimais

'\t' é um código de tabulação

BIN – Números binários

HEX – Números hexadecimais

`while(1)` é um 'loop infinito'. Como 1 é sempre 1, sempre verdadeiro para o teste efetuado pelo comando `while` (enquanto), a execução será contínua (e neste caso, não fará nada... - experimente apagar as linhas 18, 19 e 20 e rodar o programa)

Está ficando ótimo! Já conseguimos ler valores e apresentá-los no Monitor Serial.

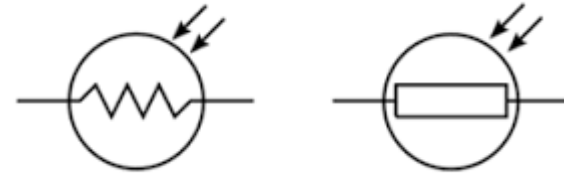
Agora, vamos ver um componente que permite fazer 'leituras' de luz.

LDR

- *Light Dependent Resistor*
 - Resistor cujo valor ôhmico de sua resistência depende da quantidade de luz que ele recebe
 - Às vezes chamado de foto resistor

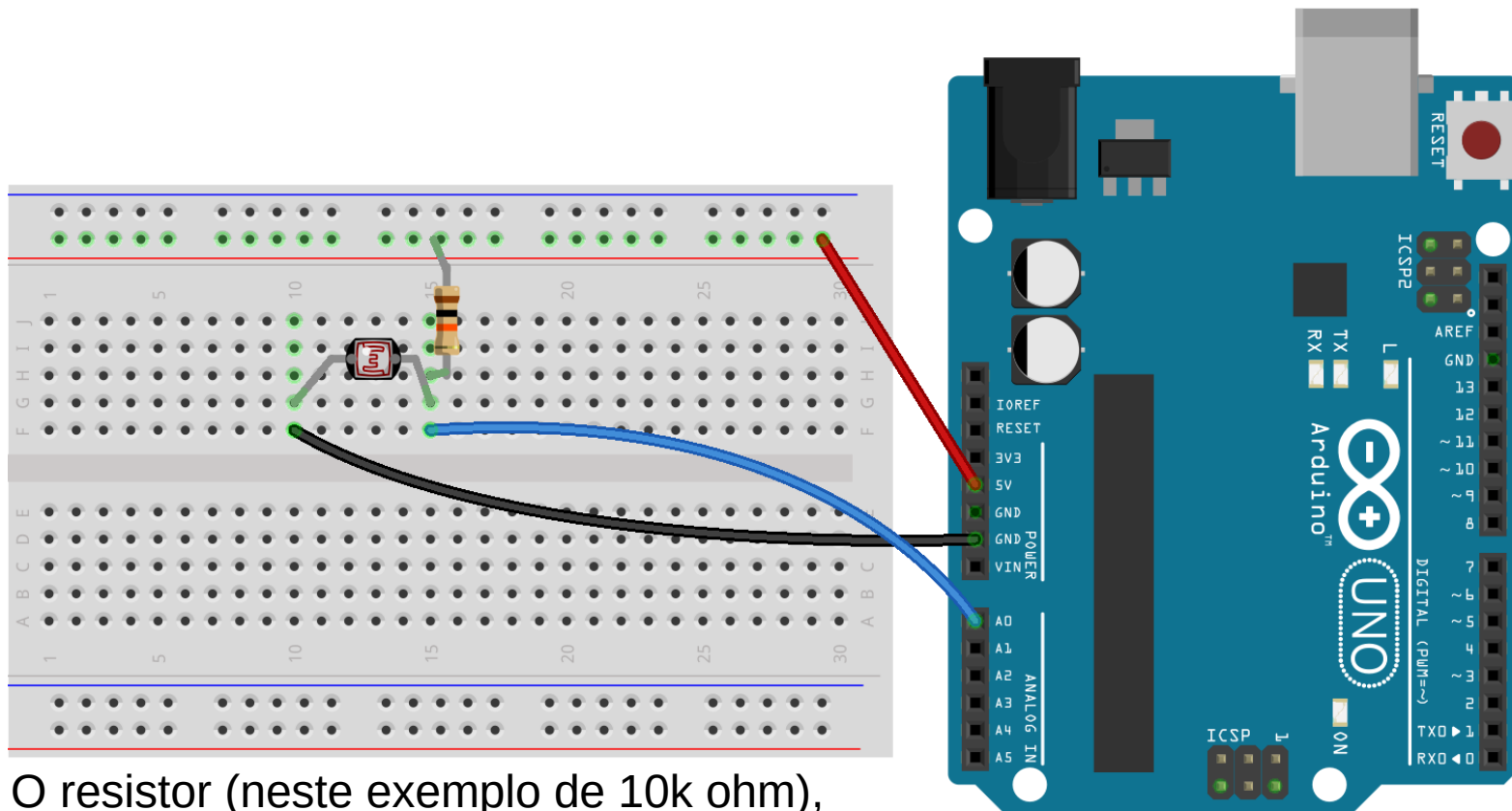


Aspecto
físico



Representação

Monte o circuito



O resistor (neste exemplo de 10k ohm), garante um nível lógico quando o LDR estiver com a resistência muito baixa

fritzing

Código

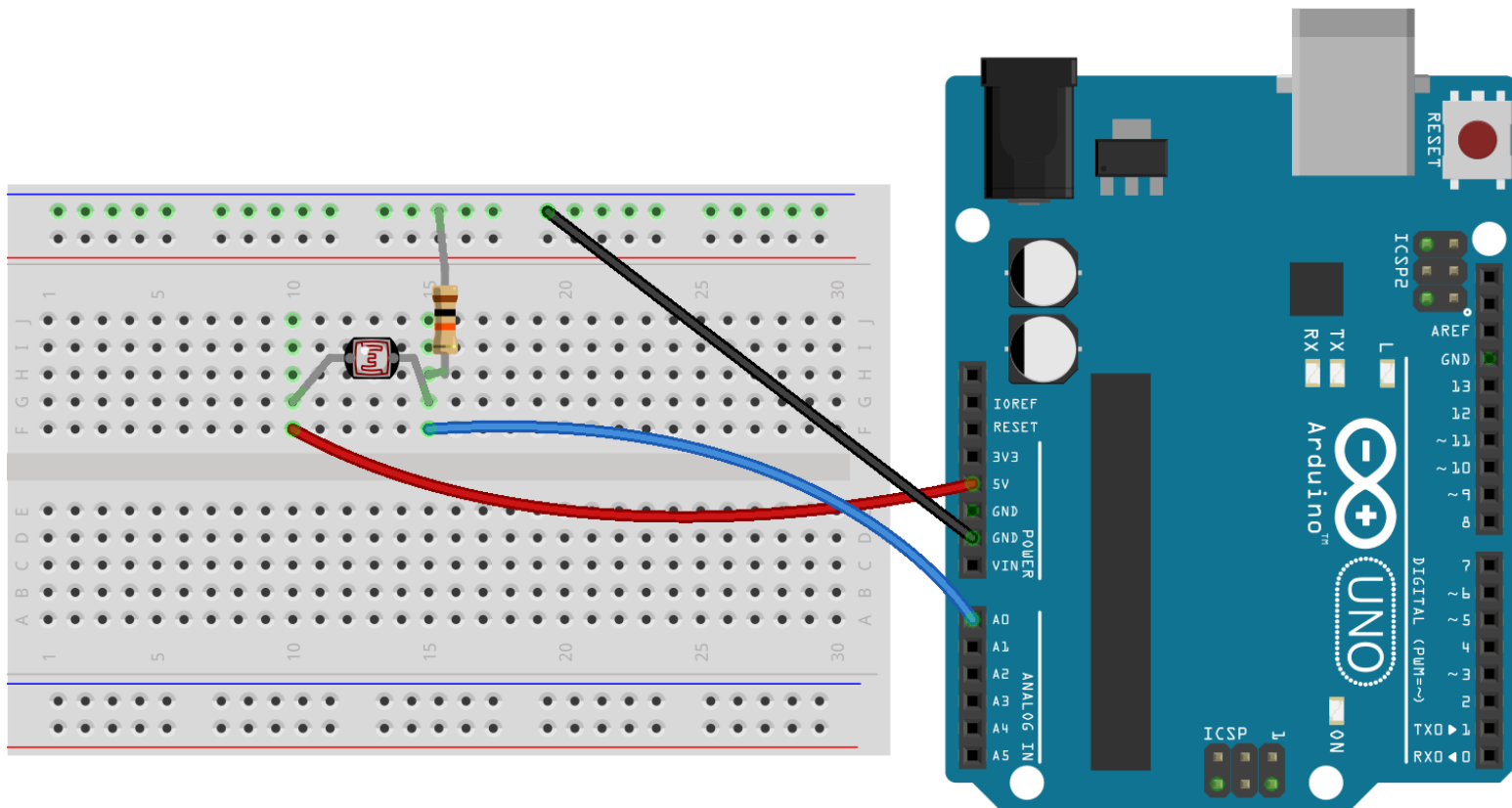
```
1 #define LDR A0
2
3 void setup() {
4     pinMode(LDR, INPUT);
5     Serial.begin(9600);
6 }
7
8 void loop() {
9     Serial.println(analogRead(LDR));
10    delay(300);
11 }
```


Teste!

Ligue o monitor serial e tente fazer com que o LDR receba muita, nenhuma ou alguma luz. Veja o resultado

Depois, se quiser, ajuste o código, como realizado com o potenciômetro, para melhorar a visualização

Será que assim funciona?



Desligue o Arduino antes de inverter os fios!

fritzing

O código é o mesmo

```
1 #define LDR A0
2
3 void setup() {
4     pinMode(LDR, INPUT);
5     Serial.begin(9600);
6 }
7
8 void loop() {
9     Serial.println(analogRead(LDR));
10    delay(300);
11 }
```

Comentários

- Funciona de ambas as formas, o que estamos mudando é a **referência**
 - Em um caso, quando a luz diminui o valor lido aumenta, e
 - No outro caso, quando a luz aumenta o valor lido aumenta
- Cada situação exigirá uma resposta diferente

O que você aprendeu:

1. Ler valores analógicos
2. Transformar os valores de 0-1023 em 0-100 ou em seu valor 'real', dentro da resolução possível
3. Apresentar os valores lidos utilizando o Monitor Serial
4. Apresentar os valores lidos utilizando o Plotter Serial

Parabéns!

