1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Exercícios com o Arduino

Prof. Simão

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Dúvidas

Se tiver dúvidas do que foi desenvolvido até o momento, tire-as agora, antes de prosseguirmos.

Caso contrário, vamos em frente...

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Veja a documentação de referência da linguagem em:
https://www.arduino.cc/reference/pt/

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Objetivo – ambientar-se com o monitor serial e com o plotter serial e interagir com a placa/ programa por meio de interações no teclado.

simao@ufpr.br - 2025

Compile o seguinte programa:

```
void setup(){
  Serial.begin(9600);
}
int x = 0;
void loop() {
  Serial.print("Numero: ");
  Serial.println(x);
  delay(300);
  x++;
}
```

Ligue o **Monitor Serial** (Ferramentas/ Monitor Serial) e veja o resultado.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10100011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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

simao@ufpr.br - 2025

```
 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

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

```
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)

simao@ufpr.br - 2025

Compile o seguinte programa:

```
void setup(){
  Serial.begin(9600);
}
int x = 0;
void loop() {
  Serial.print("Numero: ");
  Serial.println(x);
  Serial.print(" DEC: ");
  Serial.print(x);
  Serial.print(" HEX: ");
  Serial.print(x, HEX);
  Serial.print(" BIN: ");
  Serial.println(x, BIN);
  delay(300);
  x++;
}
```

Note o elemento (`DEC, HEX, BIN`) junto à informação de impressão do valor (`print`), de forma a obtermos, respectivamente, a visualização nos sistemas Decimal, Hexadecimal e Binário (também foi usado o '`print`', ao invés de '`println`', no qual o segundo comando acrescenta uma quebra de linha – '`ln`' – após a impressão).

Ligue o **Monitor Serial** (Ferramentas/ Monitor Serial) e veja o resultado.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

Compile o seguinte programa:

```
long aleatorio;

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

void loop() {
  Serial.print("\nEntre 0 e 99:");
  aleatorio = random(100);
  Serial.println(aleatorio);
  Serial.print("\nEntre 5 e 14: ");
  aleatorio = random(5, 15);
  Serial.println(aleatorio);
  delay(300);
}
```

O comando '`\n`' inserido dentro da instrução print faz como que seja pulada uma linha ('`\n = new line`'). A impressão será algo como: `Entre 0 e 99:7; Entre 5 e 14: 14; Entre 0 e 99:73 ;` ...

Se você compilar o programa novamente e executá-lo novamente os valores sorteados deverão ser os mesmos. Isto porque o gerador de números aleatórios não foi inicializado.

Ligue o **Monitor Serial** (Ferramentas/ Monitor Serial) e veja o resultado.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111
simao@ufpr.br - 2025

Uma forma de inicializar o gerador é por meio do uso da função `ramdomSeed()` (semente aleatória):

```
long aleatorio;

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

  // inicializa o gerador de números aleatórios.
  // um pino analógico desconectado irá retornar um
  // valor aleatório de tensão em analogRead()
  randomSeed(analogRead(0));
}

void loop() {
  Serial.print("\nEntre 0 e 99:");
  aleatorio = random(100);
  Serial.println(aleatorio);
  Serial.print("\nEntre 5 e 14: ");
  aleatorio = random(5, 15);
  Serial.println(aleatorio);
  delay(300);
}
```

Se você compilar o programa novamente e executá-lo novamente os valores sorteados deverão ser diferentes a cada momento. Compare a execução do programa usando / não usando o comando `randomSeed()`.

Ligue o **Monitor Serial** (Ferramentas/ Monitor Serial) e veja o resultado.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111
simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Compile o seguinte programa:

```
long aleatorio;
void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(0));
}
void loop() {
  aleatorio = random(50);
  Serial.println(aleatorio);
}
```

Ligue o **Monitor Serial** (Ferramentas/ Monitor Serial) e veja o resultado. Depois, feche a tela do monitor serial e ligue o **Plotter Serial** (Ferramentas/ Plotter Serial) e veja o resultado.

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Vamos criar mais uma variável de exibição. Compile o seguinte programa:

```
long aleatorio1, aleatorio2;
void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(0));
}
void loop() {
  aleatorio1 = random(50);
  Serial.println(aleatorio1);
  aleatorio2 = random(50);
  Serial.println(aleatorio2);
}
```

Ligue o **Plotter serial**. Viu alguma diferença? Não são duas variáveis? O plotter simplesmente continuou a mostrar os valores continuamente.

simao@ufpr.br - 2025

Agora, teste assim: compile o programa que segue.

```
long aleatorio1, aleatorio2;
void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(0));
}
void loop() {
  aleatorio1 = random(50);
  Serial.print(aleatorio1);
  Serial.print(",");
  aleatorio2 = random(50);
  Serial.print(aleatorio2);
  Serial.print("\n");
}
```

O uso do separador (‘,’) entre os valores passados ao comando print (e não é o `println`), faz com que o plotter interprete a mudança de variável. Para uma nova série de valores, usamos o terminador de nova linha (‘`\n`’) – poderia ter sido utilizado `Serial.prinln(aleatorio2)` ao invés de primeiro mostrar o valor e depois pular uma linha, o resultado seria a mesma interpretação por parte do plotter serial.

Ligue o **Plotter serial** e veja a diferença.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

Mas a legenda ficou confusa, não é mesmo? Vamos acertá-la? Compile o programa que segue:

```
long aleatorio1, aleatorio2;
void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(0));
}
void loop() {
  aleatorio1 = random(50);
  Serial.print("Aleatorio1: ");
  Serial.print(aleatorio1);
  Serial.print(",");
  Serial.print("Aleatorio2: ");
  aleatorio2 = random(50);
  Serial.print(aleatorio2);
  Serial.print("\n");
}
```

Ligue o **Plotter serial** Agora a legenda deverá estar ok.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10100011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Trabalhar lendo um valor na interface serial e, em consequência, testar algo e/ ou agir em uma saída é o básico do uso em controles do tipo

Entrada → Processamento → Saída.

E, este processo de interação será utilizado quando formos trabalhar com *BlueTooth*.

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Parabéns, você está avançando!

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Dúvidas

Se tiver dúvidas do que foi desenvolvido até o momento, tire-as agora, antes de prosseguirmos.

Caso contrário, vamos em frente...

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Leitura de valores analógicos

→ algum valor entre ‘0’ e ‘1’

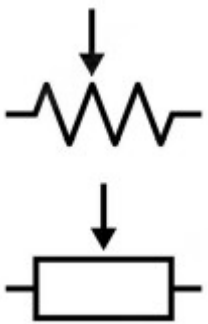
→ uma faixa de valores entre um mínimo e um máximo

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

O potenciômetro

Aspecto físico



Símbolos usuais

Trata-se de um resistor variável, ou ajustável.

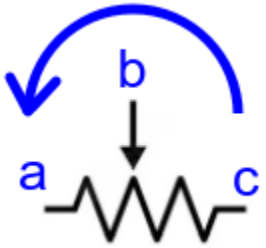
O terminal ‘do meio’, corresponde ao ajuste. A resistência será ajustada entre um ponto e outro à medida em que eixo é deslocado.

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).

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Valores proporcionais à posição

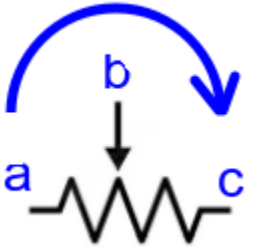


Quando o cursor for levado totalmente em direção a esta extremidade, a resistência será nula (0 ohm).

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

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

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



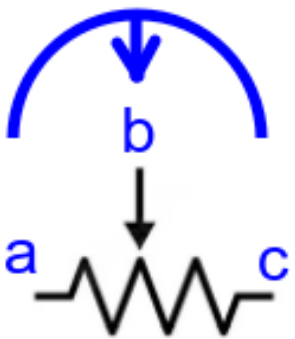
simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Valores proporcionais à posição

Quando o cursor for levado extamente 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

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Divisor de tensão

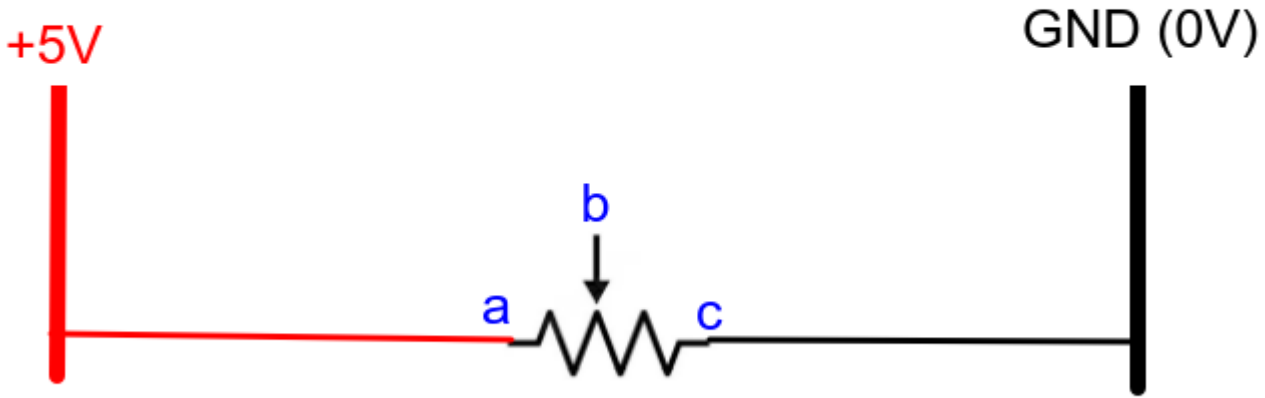
- Podemos utilizar um potenciômetro como divisor de tensão:
  - por exemplo, podemos ligar uma extremidade no `+5V` e outra no `GND` (0V); no cursor, em função de sua posição, obteremos tensões proporcionais, entre o mínimo (0V) e o máximo (5V, para este exemplo)

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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'

simao@ufpr.br - 2025

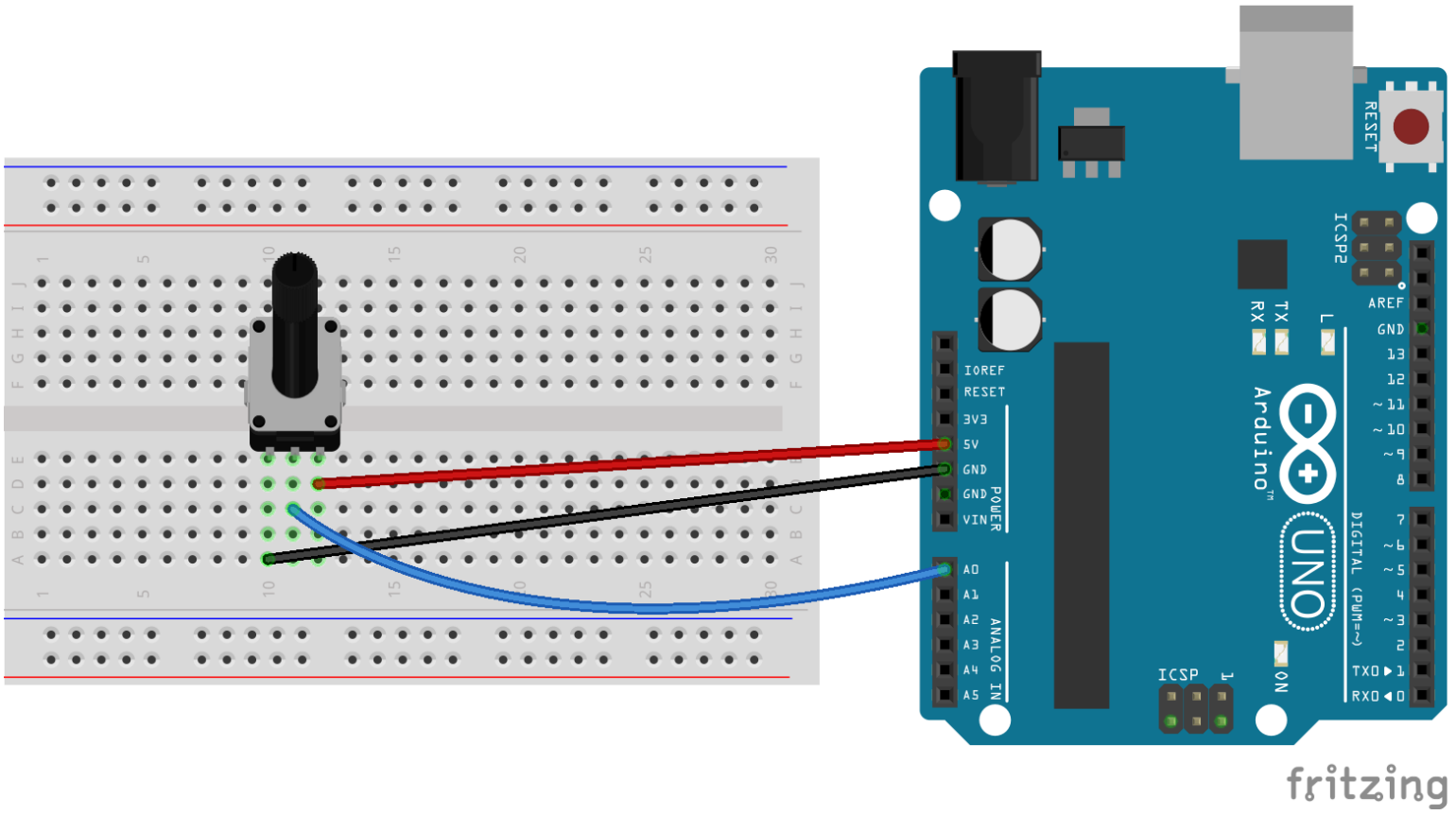
1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Vamos praticar

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Monte um circuito como abaixo



simao@ufpr.br - 2025

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 }
```

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Monitor serial

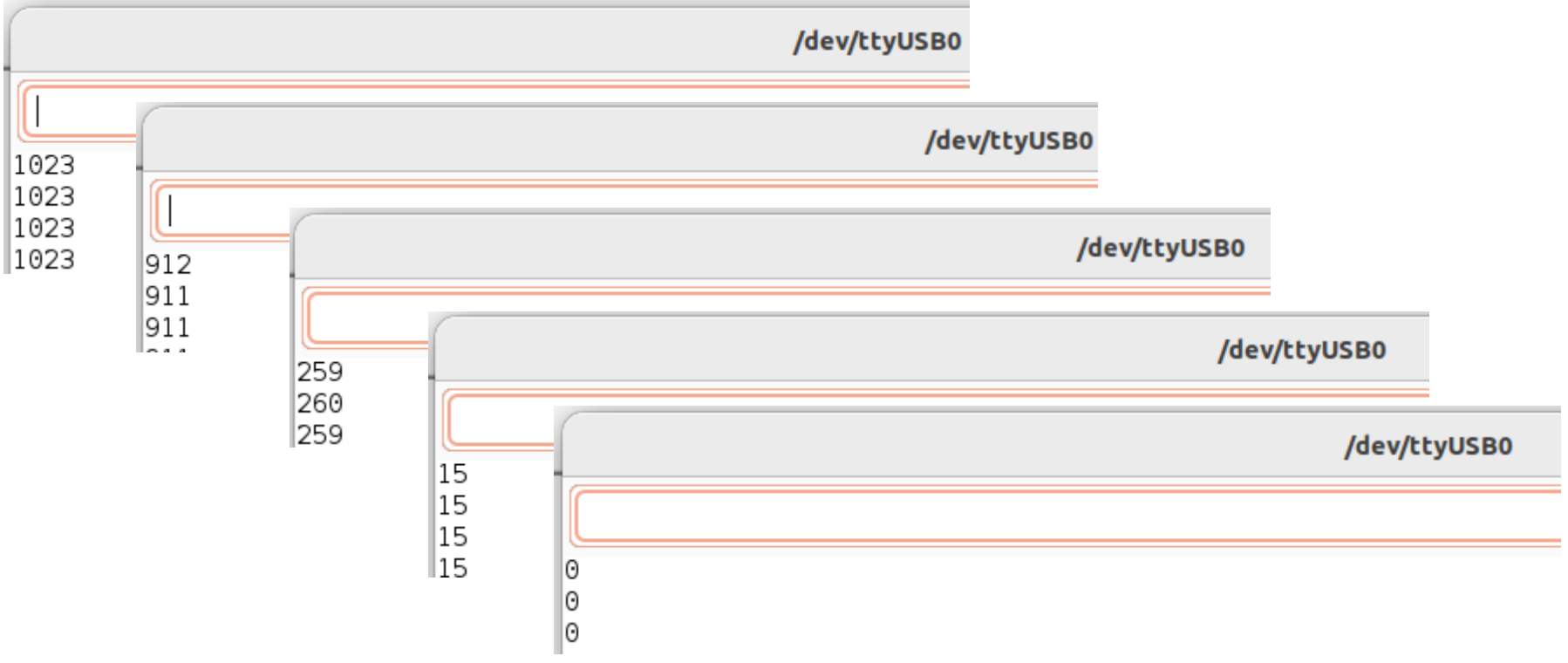
Ligue o monitor serial: Ferramentas/ Monitor serial

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Monitor serial

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



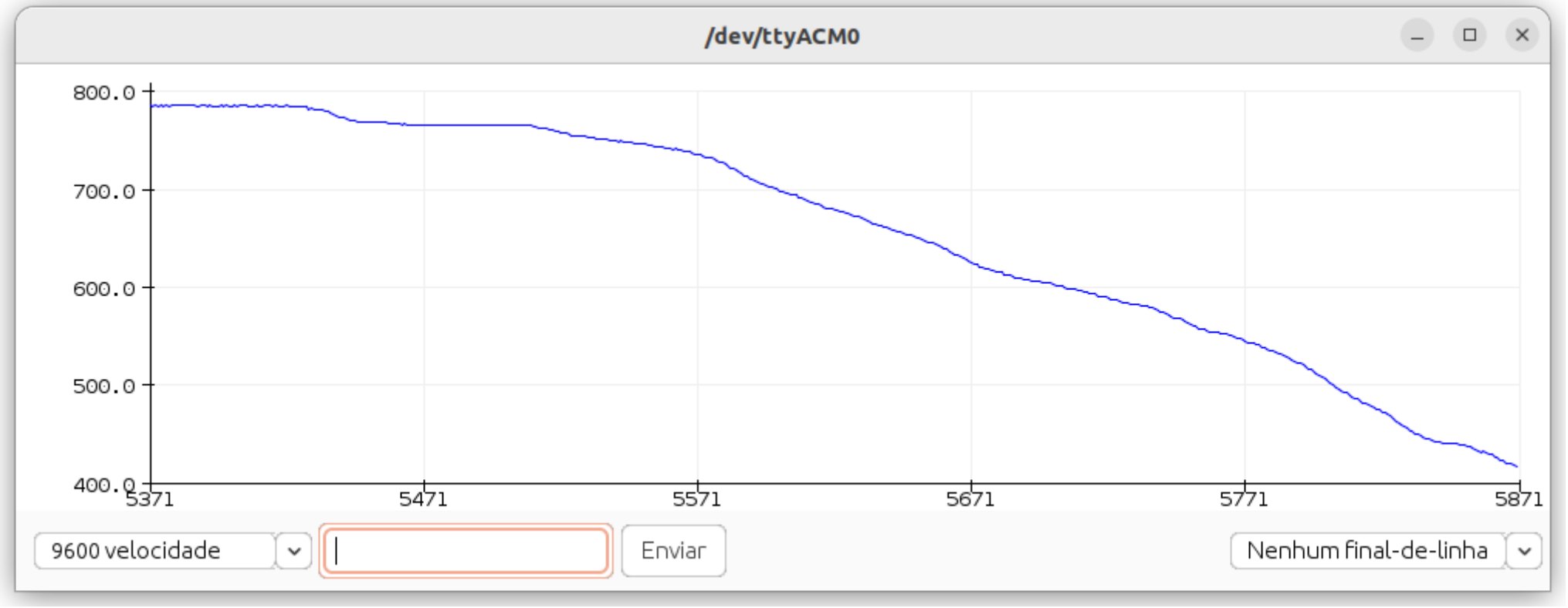
simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10100011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Feche a tela do monitor serial.

Ligue o Plotter Serial (Ferramentas/ Plotter Serial)

Ajuste o potenciômetro para atualizar o gráfico



1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111
simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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));
}
```

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

Entendendo o código

```
1  #define potenciometro A0
2
3  voi
4     S
5     p
6  }
7
8  voi
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...)

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

Entendendo o código

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

Funções básicas:
`setup()` e `loop()`

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

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

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

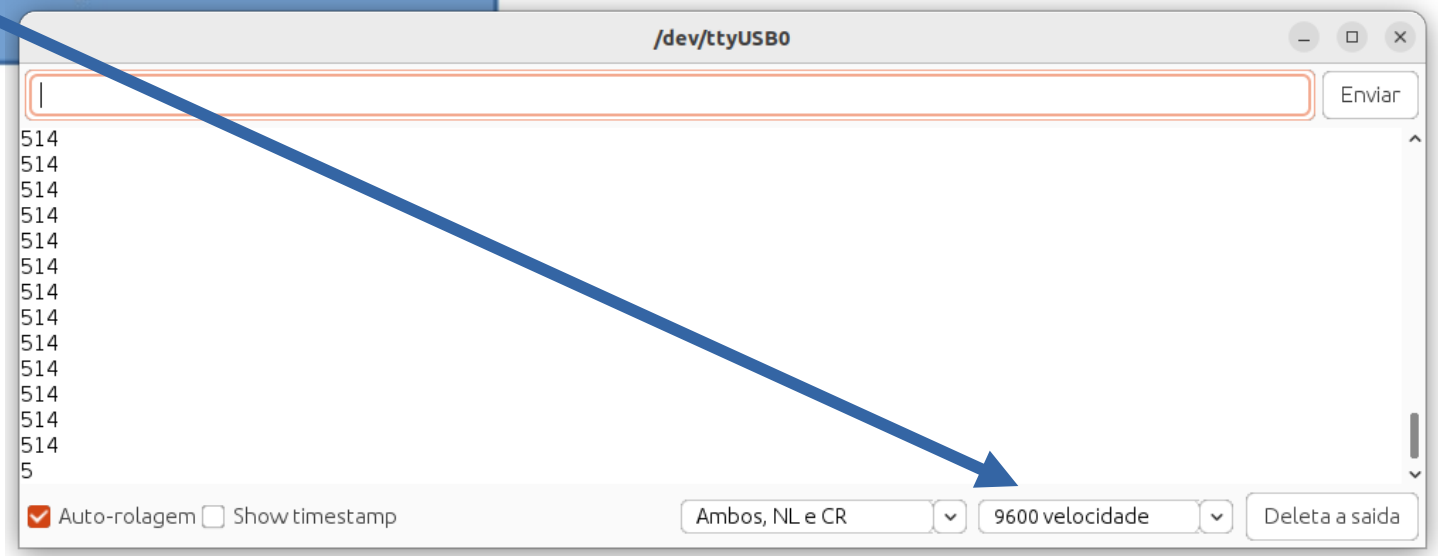
simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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



simao@ufpr.br - 2025

Entendendo o código

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

```
1  #def
2
3  void
4    Se
5    pinMode(potenciometro, INPUT);
6  }
7
8  void loop() {
9    Serial.println(analogRead(potenciometro));
10 }
```

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

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`.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10100011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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...

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Por quê entre 0 e 1023?

- Existem infinitos valores analógicos
- Por exemplo, entre 0 e 1 inteiros, temos:
  - 0,0000000000001
  - 0,000053
  - 0,008
  - 0,03
  - ...

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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.

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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`

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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)

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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)

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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.

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10100011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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?

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Resolução

- A resolução da medida pode ser calculada por:

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

- O que, para nosso caso, será:

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

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

simao@ufpr.br - 2025

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.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100001 1101111

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10100011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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.

simao@ufpr.br - 2025

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 }
```

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

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.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111
simao@ufpr.br - 2025

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 }
```

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111
simao@ufpr.br - 2025

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`)

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10100011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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.

simao@ufpr.br - 2025

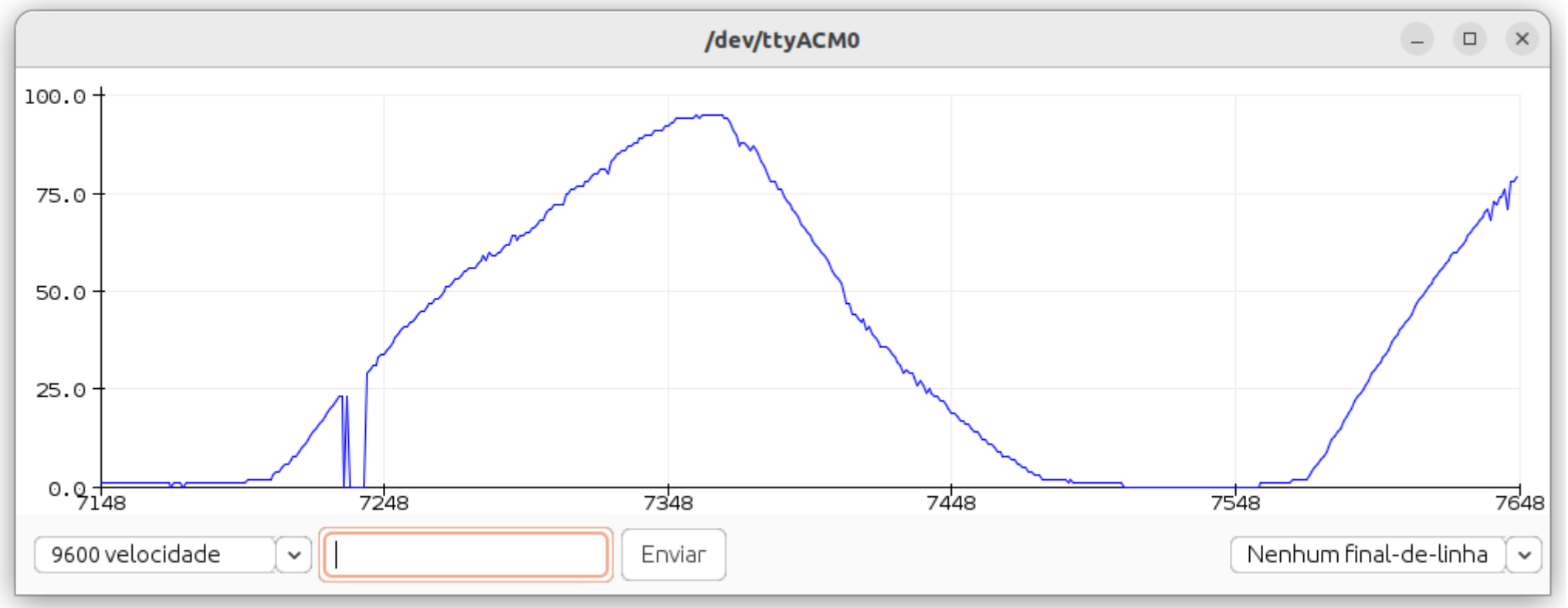
Função 'map'

```
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)

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111
simao@ufpr.br - 2025

Usando map – ajuste o potenciômetro



1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111
simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Teste diferentes valores para a função ‘`map`’

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Dúvidas

Se tiver dúvidas do que foi desenvolvido até o momento, tire-as agora, antes de prosseguirmos.

Caso contrário, vamos em frente...

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

EXPERIMENTOS

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Monitor serial

- O monitor serial é um programa de comunicação serial, capaz de receber e enviar comandos.
- Vamos experimentar

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Monitor serial 1

```
1  void setup() {
2    Serial.begin(9600);
3  }
4
5  void loop() {
6     if (Serial.available())
7     {
8      byte lido;
9      lido = Serial.read();
10     Serial.print("Recebi ");
11     Serial.println((char)lido);
12     }
13 }
```

simao@ufpr.br - 2025

Monitor serial 1

```
 1 void setup() {
 2   Serial.begin(9600);
 3 }
 4
 5 void loop() {
 6    if (Serial.available())
 7    {
 8     byte lido;
 9     lido = Serial.read();
10     Serial.print("Recebi ");
11     Serial.println((char)lido);
12     }
13 }
```

Na linha 6, SE houver algum *byte* a ser lido na entrada do monitor serial, o *status* `available` estará em nível alto (verdadeiro, *true*, `HIGH`, `1`).

SE este for o caso, será executado o bloco de comandos existente entre as linhas 7 e 12.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

Monitor serial 1

```
 1 void setup() {
 2   Serial.begin(9600);
 3 }
 4
 5 void loop() {
 6    if (Serial.available())
 7    {
 8     byte lido;
 9     lido = Serial.read();
10     Serial.print("Recebi ");
11     Serial.println((char)lido);
12     }
13 }
```

Na linha 8 é declarada a variável ‘lido’, do tipo `byte`.

Na linha 9, o *byte* lido no monitor serial é colocado na variável ‘lido’.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

Monitor serial 1

```
 1 void setup() {
 2   Serial.begin(9600);
 3 }
 4
 5 void loop() {
 6    if (Serial.available())
 7    {
 8     byte lido;
 9     lido = Serial.read();
10     Serial.print("Recebi ");
11     Serial.println((char)lido);
12     }
13 }
```

Na linha 10 será enviado ao monitor serial o texto “Recebi “ e permanecerá na mesma linha.

Na linha 11, o valor da variável ‘lido’ é convertido para um valor de caracter, por meio do *typecast* `(char)` colocado antes da variável.

Ainda na linha 11, o valor convertido da variável ‘lido’, agora um caracter, será enviado para o monitor serial, assim como um comando de quebra de linha (o ‘`ln`’ ao final do comando ‘print’ = *newline*, nova linha).

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111
simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Monitor serial 2

```
1  void setup() {
2    Serial.begin(9600);
3    pinMode(13, OUTPUT);
4  }
5
6  void loop() {
7     if (Serial.available())
8     {
9      byte lido;
10     lido = Serial.read();
11     if((char)lido == 'L' || (char)lido == 'l')
12       digitalWrite(13, 1);
13     if((char)lido == 'D' || (char)lido == 'd')
14       digitalWrite(13, 0);
15     }
16 }
```

simao@ufpr.br - 2025

Monitor serial 2

```
1  void setup() {
2    Serial.begin(9600);
3    pinMode(13, OUTPUT);
4  }
5
6  void loop() {
7     if (Serial.available())
8     {
9      byte lido;
10     lido = Serial.read();
11     if((char)lido == 'L' || (char)lido == 'l')
12       digitalWrite(13, 1);
13     if((char)lido == 'D' || (char)lido == 'd')
14       digitalWrite(13, 0);
15     }
16 }
```

Na linha 7, SE houver algum byte a ser lido na entrada do monitor serial, o *status* `available` estará em nível alto (verdadeiro, *true*, `HIGH`, `1`).

SE for este o caso será executado o bloco de comandos existente entre as linhas 8 e 15.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

Monitor serial 2

```
 1 void setup() {
 2   Serial.begin(9600);
 3   pinMode(13, OUTPUT);
 4 }
 5
 6 void loop() {
 7    if (Serial.available())
 8    {
 9     byte lido;
10     lido = Serial.read();
11     if((char)lido == 'L' || (char)lido == 'l')
12       digitalWrite(13, 1);
13     if((char)lido == 'D' || (char)lido == 'd')
14       digitalWrite(13, 0);
15     }
16 }
```

Na linha 9 é declarada uma variável do tipo `byte` a qual recebe o nome de ‘lido’.

Na linha 10 o valor apresentado pela entrada serial é lido (`Serial.read`), e colocado na variável ‘lido’.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Monitor serial 2

```
1 void setup() {
2   Serial.begin(9600);
3   pinMode(13, OUTPUT);
4 }
5
6 void loop() {
7    if (Serial.available())
8    {
9     byte lido;
10    lido = Serial.read();
11    if((char)lido == 'L' || (char)lido == 'l')
12      digitalWrite(13, 1);
13    if((char)lido == 'D' || (char)lido == 'd')
14      digitalWrite(13, 0);
15    }
16 }
```

Na linha 11 o valor da variável ‘lido’ é testado.

Para isso, ele é convertido para um valor de caracter, por meio do *typecast* `(char)` colocado antes da variável.

Caso o valor seja ‘L’, ou (`||`) caso o valor seja ‘l’, o led da placa (pino 13) será ativado. Lembrando que, em C/C++, minúsculas são diferentes de maiúsculas – por isso são consideradas as duas para o comando desejado.

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Monitor serial 2

```
1 void setup() {
2   Serial.begin(9600);
3   pinMode(13, OUTPUT);
4 }
5
6 void loop() {
7    if (Serial.available())
8    {
9     byte lido;
10    lido = Serial.read();
11    if((char)lido == 'L' || (char)lido == 'l')
12      digitalWrite(13, 1);
13    if((char)lido == 'D' || (char)lido == 'd')
14      digitalWrite(13, 0);
15    }
16 }
```

Na linha 13 o valor da variável ‘lido’ é testado.

Para isso, ele é convertido para um valor de caracter, por meio do *typecast* `(char)` colocado antes da variável.

Caso o valor seja ‘D’, ou (`||`) caso o valor seja ‘d’, o led da placa (pino 13) será desativado. Lembrando que, em C/ C++, minúsculas são diferentes de maiúsculas – por isso são consideradas as duas para o comando desejado.

simao@ufpr.br - 2025

Monitor serial 3

```
1  void setup() {
2    Serial.begin(9600);
3  }
4
5  void loop() {
6     if (Serial.available())
7     {
8      switch(Serial.read())
9      {
10       case 'L':
11         Serial.println("Recebi um L");
12       break;
13     }
14   }
15 }
```

Na linha 8 o valor lido do monitor serial é testado dendtro do comando `switch`.

A linha 10 testa caso (case) a leitura seja ‘L’ e, na linha 11, imprime uma mensagem no monitor serial.

A linha 12, `break`, para a execução do `case`.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111
simao@ufpr.br - 2025

Monitor serial 4

```
 1 int codigoRecebido = 0;
 2
 3 void setup() {
 4   Serial.begin(9600);
 5 }
 6
 7 void loop() {
 8   if (Serial.available() > 0) {
 9     codigoRecebido = Serial.read();
10     Serial.print("O que o Arduino leu: ");
11     Serial.println(codigoRecebido, DEC);
12   }
13 }
```

Outra forma de interagir.

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111
simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Dúvidas

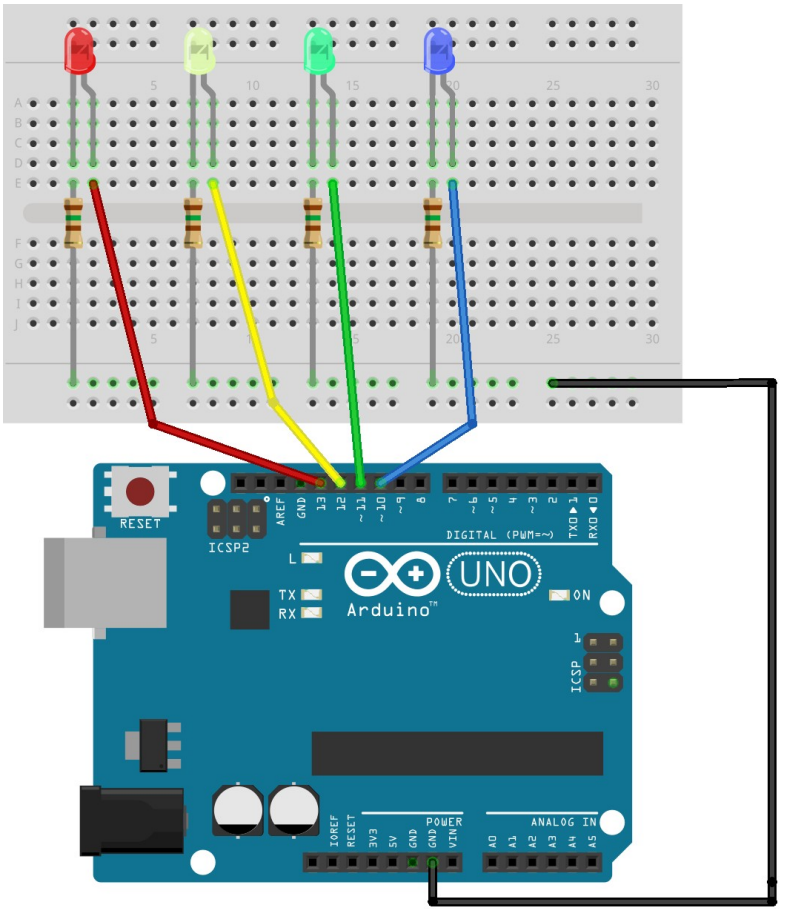
Se tiver dúvidas do que foi desenvolvido até o momento, tire-as agora, antes de prosseguirmos.

Caso contrário, vamos em frente...

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

4 leds, 4 resistores



No exemplo, os resistores são de 150 ohms. Utilize valores entre 100 ohms e 1000 ohms.

simao@ufpr.br - 2025

```
void loop(){
  if (Serial.available())
  {
    switch(Serial.read())
    {
    case 't':
      digitalWrite(ledAmarelo, 0);
      digitalWrite(ledVermelho, 0);
      digitalWrite(ledVerde, 0);
      digitalWrite(ledAzul, 0);
      Serial.println("Todos apagados");
      break;
    case 'T':
      digitalWrite(ledAmarelo, 1);
      digitalWrite(ledVermelho, 1);
      digitalWrite(ledVerde, 1);
      digitalWrite(ledAzul, 1);
      Serial.println("Todos acesos");
      break;
      }
    }
  }
```

Use o monitor serial para interagir com seu circuito (os comandos poderiam também vir via Bluetooth ou outra conexão).

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Plotter Serial

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Imagem: fotos do autor

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

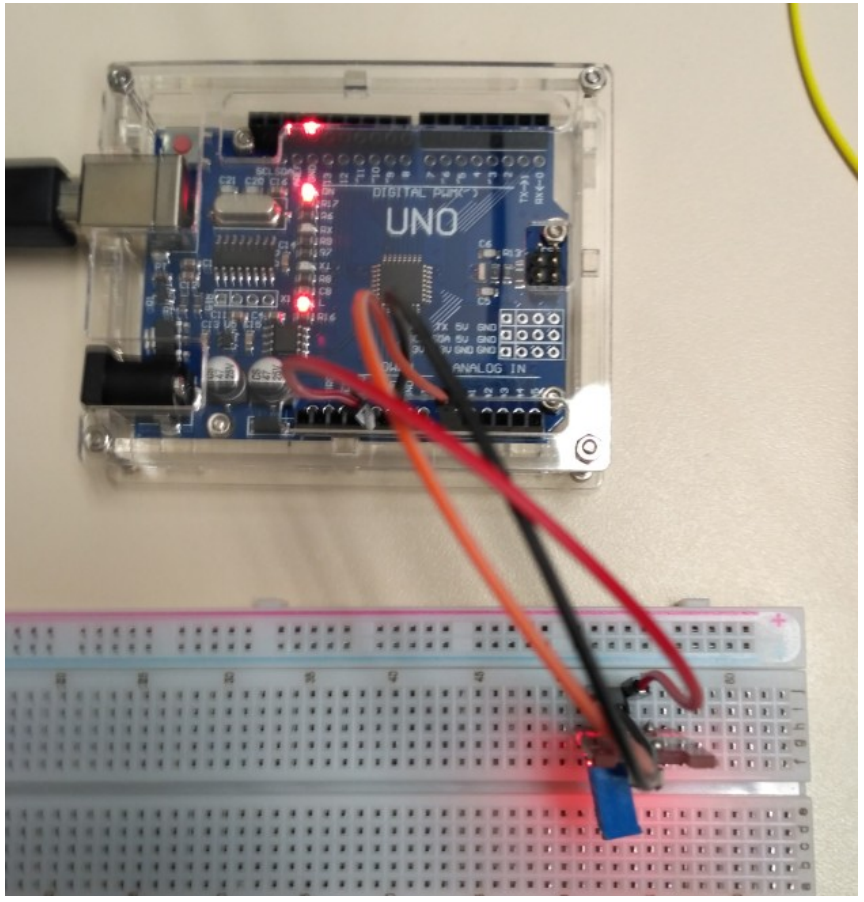
O que você precisará?

- Arduino configurado na IDE
- Uma placa de experimentação e fios
- Um sensor de som KY-037

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Ligação



O pino A0 do KY-037 é ligado na entrada analógica A0 da placa do Arduino.

O pino G do KY-037 é ligado ao GND da placa do Arduino.

O pino + do KY-037 é ligado ao 5V da placa do Arduino.

Imagem: fotos do autor

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Sequência

- Desligue o cabo USB
- Ligue os componentes na placa
- Conecte os fios à placa do Arduino; anote as portas utilizadas
- Elabore o código

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Código

Para ver o resultado, vamos ligar o Plotter Serial.

Ferramentas / **Plotter Serial**

Imagem: cópia de tela realizada pelo autor

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Ferramentas / Plotter Serial

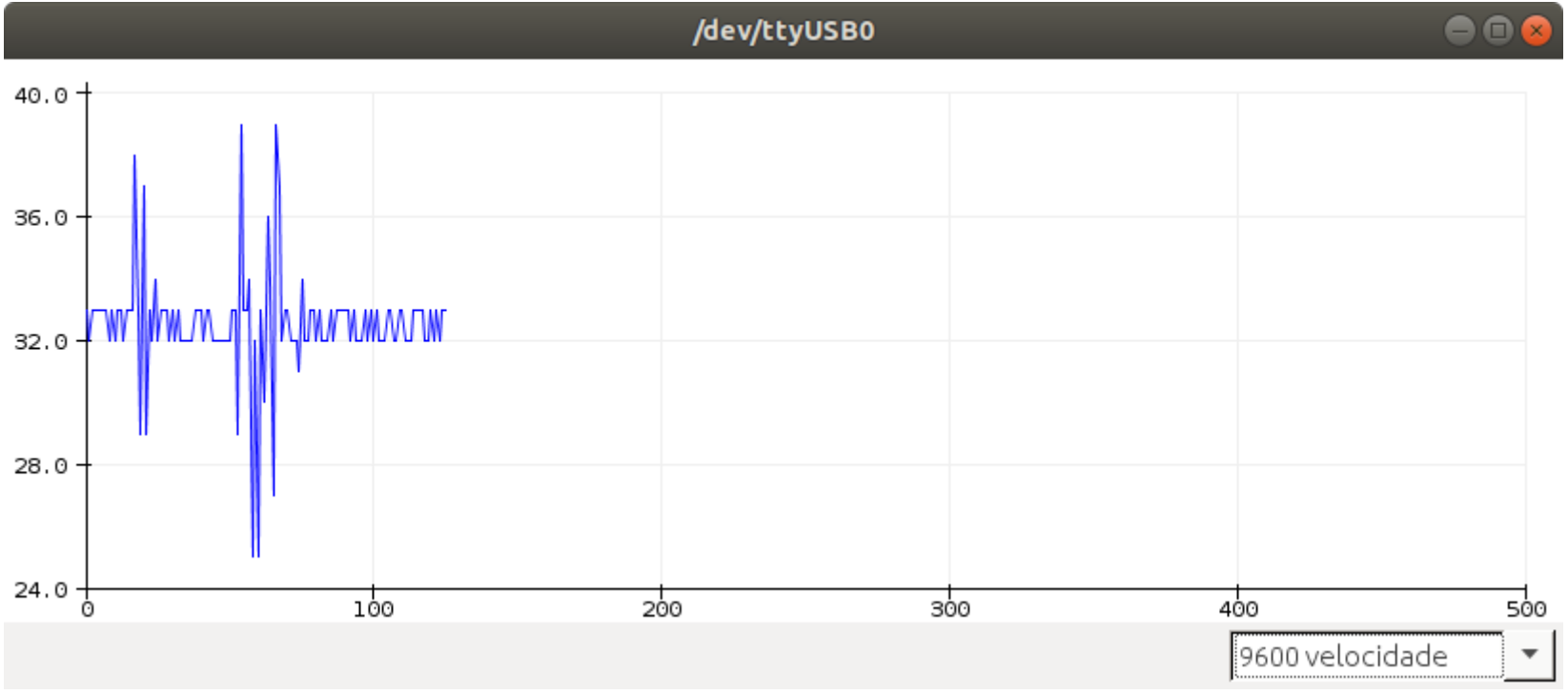


Imagem: cópia de tela realizada pelo autor

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Resistor dependente da luz

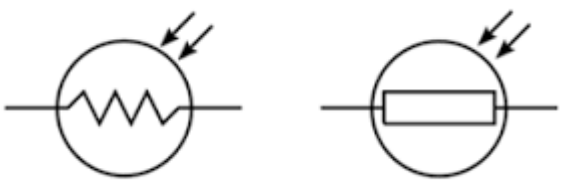
simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10100011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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

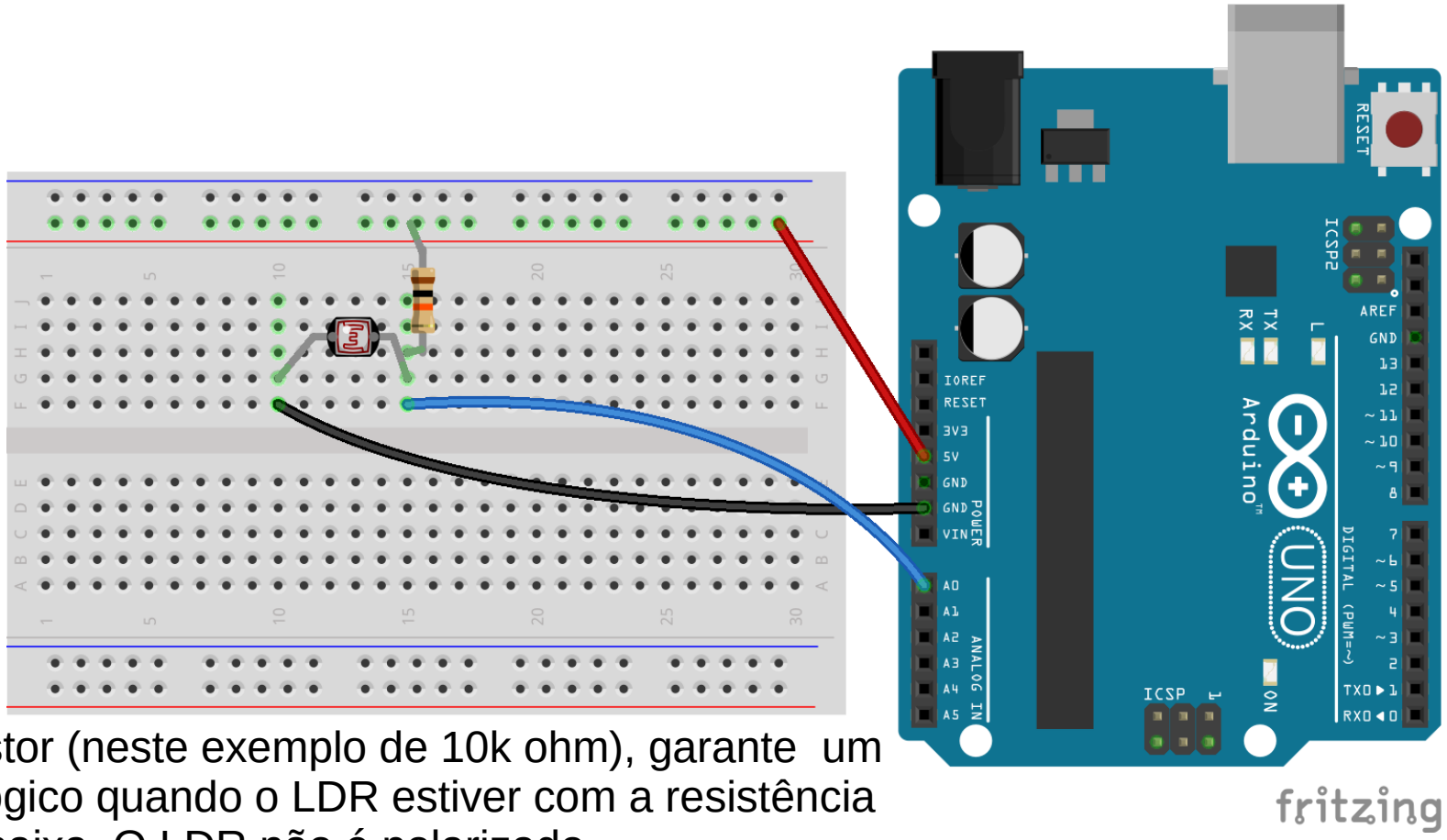
Aspecto físico



Representação

simao@ufpr.br - 2025

Monte o circuito



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

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

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 }
```

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10100011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Teste!

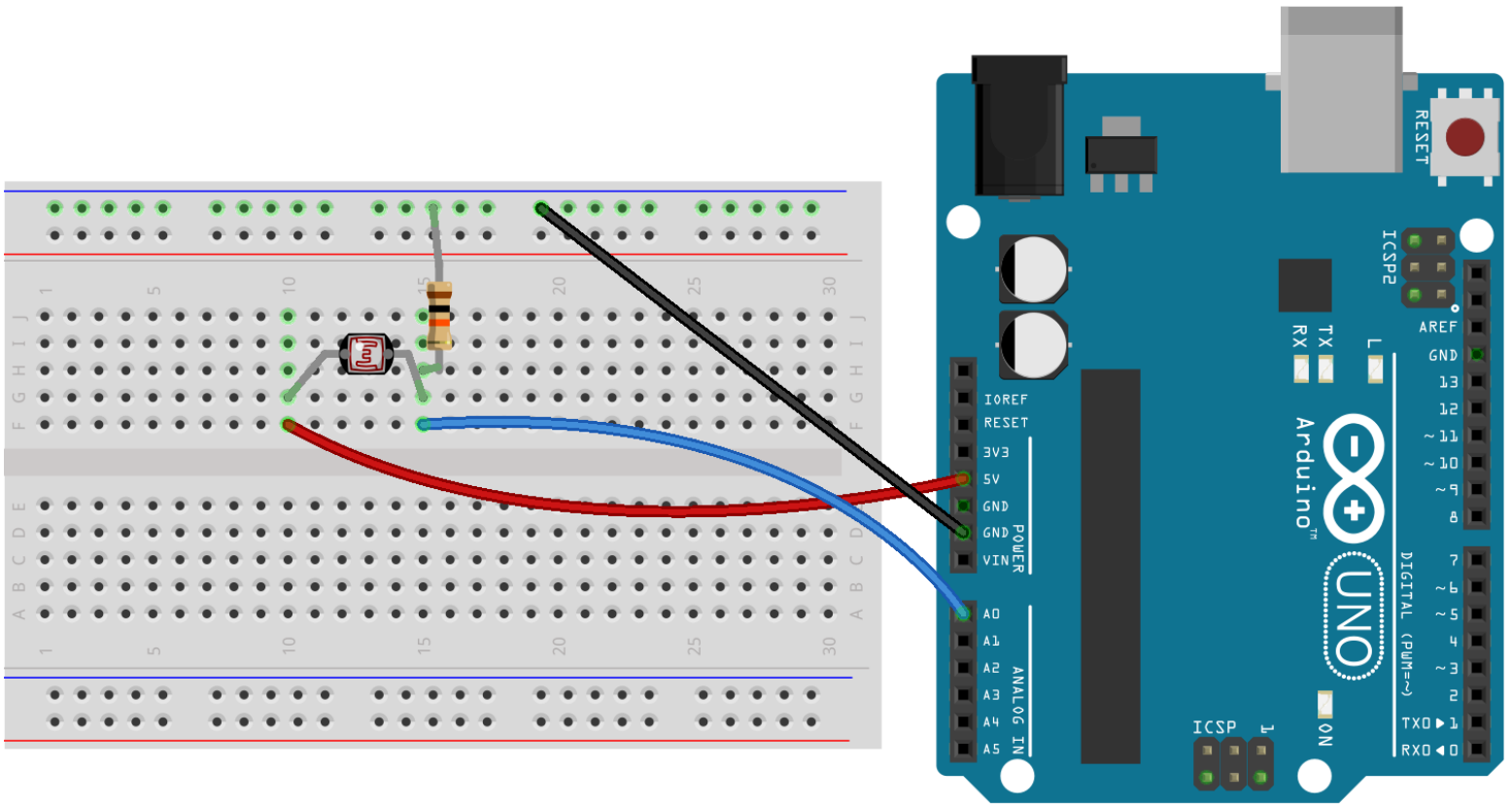
Ligue o monitor serial e tente fazer com que o LDR receba muita, nenhuma ou alguma luz. Veja o resultado

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

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Será que assim funciona?



Desligue o Arduino antes de inverter os fios!

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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 }
```

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10100011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

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

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Dúvidas

Se tiver dúvidas do que foi desenvolvido até o momento, tire-as agora, antes de prosseguirmos.

Caso contrário, vamos em frente...

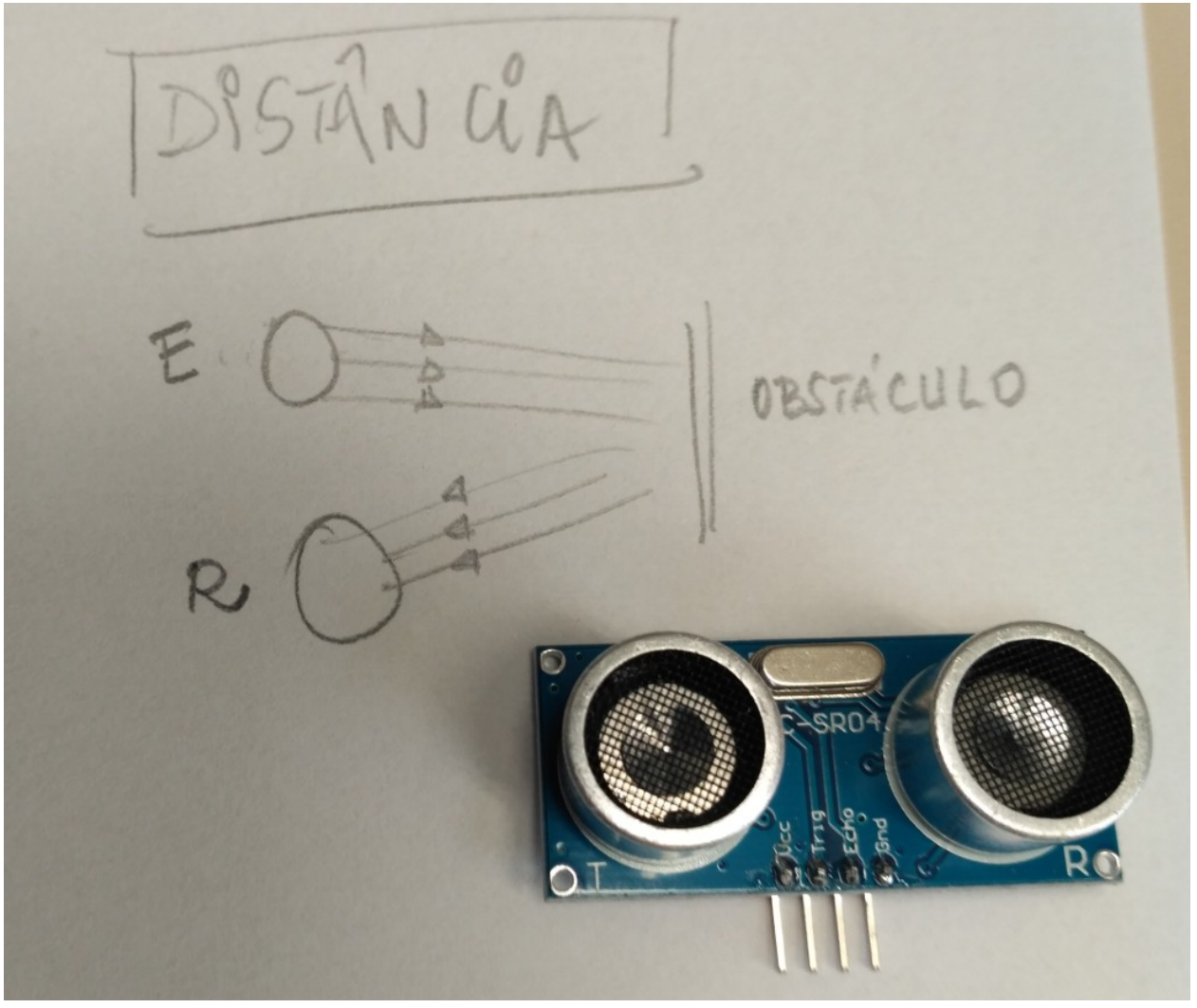
simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Vamos trabalhar com o sensor de ultrassom

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111



Atenção à polaridade

Imagem: fotos do autor

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Medidor de distância

- Baseado em seus experimentos anteriores, crie um medidor de distância

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

O que você precisará?

- Arduino configurado na IDE
- Uma placa de experimentação e fios
- Um emissor – receptor de ultrassom SR-04

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

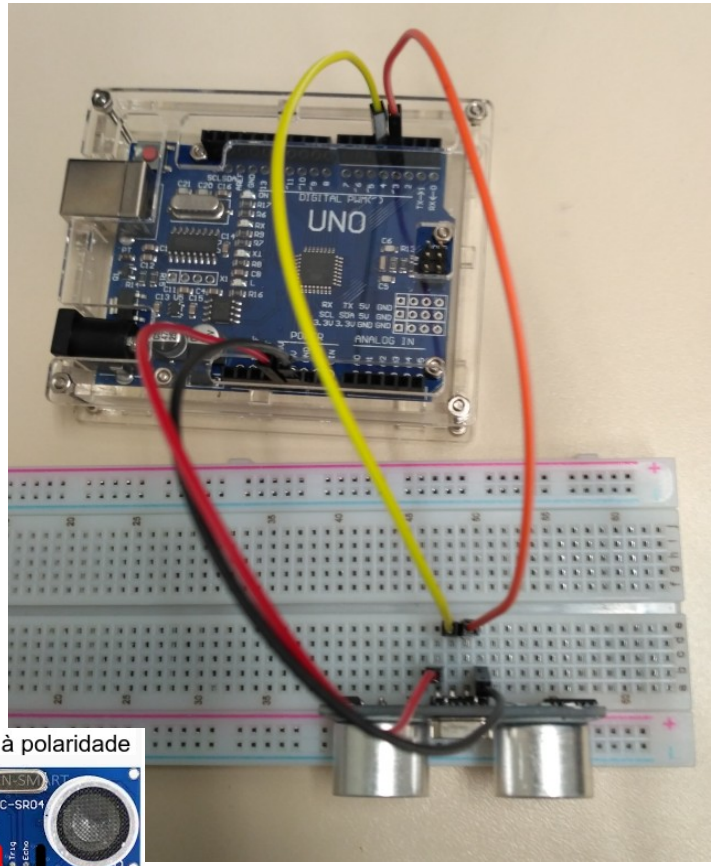
Sequência

- Desligue o cabo USB
- Ligue os componentes na placa
- Conecte os fios à placa do Arduino; anote as portas utilizadas
- Elabore o código

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1101111 0100000 1101001 0100000 1101011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Ligação



O pino VCC do SR-04 é ligado ao 5V da placa do Arduino.

O pino GND do SR-04 é ligado ao GND da placa do Arduino.

O pino TRIG do SR-04 é ligado na digital 4 da placa do Arduino.

O pino ECHO do SR-04 é ligado na digital 3 da placa do Arduino.

(Se usar outras portas troque no código exemplo a seguir)

Imagem: fotos do autor

simao@ufpr.br - 2025

Código

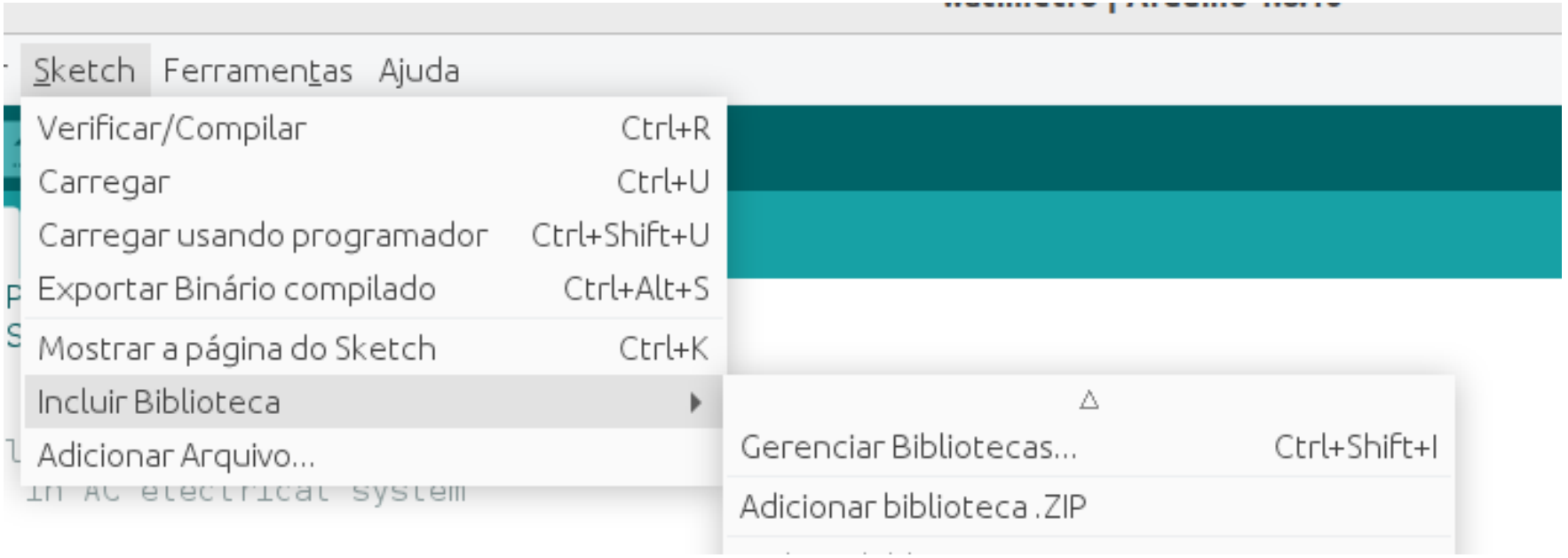
Quem faz todo o trabalho é a biblioteca incluída no código, “Ultrasonic.h”.

SE ocorrer um erro de compilação indicando que a biblioteca não está instalada, veja o slide seguinte.

Imagem: cópia de tela realizada pelo autor

simao@ufpr.br - 2025
1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

SE for necessário incluir a biblioteca



Além da óbvia pesquisa na internet, você poderá encontrar bibliotecas em:
https://www.arduino.cc/reference/en/libraries/

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111
simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Código

Arquivo Editar Sketch Ferramentas Ajuda

Salvar

medeDistancia

```
#include <Ultrasonic.h>

#define trig 4
#define echo 3

Ultrasonic ultrasonic(trig, echo);

void setup()
{
  pinMode(echo, INPUT);
  pinMode(trig, OUTPUT);
  Serial.begin(9600);
  Serial.println("Iniciando...\n\n");
}

void loop()
{
digitalWrite(trig, LOW);
delayMicroseconds(2);
digitalWrite(trig, HIGH);
delayMicroseconds(10);
digitalWrite(trig, LOW);
int distancia = (ultrasonic.Ranging(CM));
Serial.print("Distancia em CM: ");
Serial.println(distancia);
delay(2000);
}
```

Quem faz todo o trabalho é a biblioteca incluída no código, “Ultrasonic.h”.

O valor da distância está sendo enviado à saída Serial.

Para vê-lo, vamos ligar o Monitor Serial.

Basta clicar no ícone à esquerda da tela:

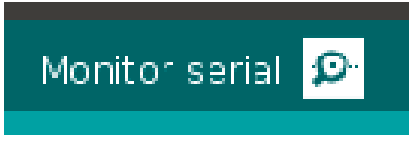


Imagem: cópia de tela realizada pelo autor

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Ferramentas / Monitor serial

Imagem: cópia de tela realizada pelo autor

simao@ufpr.br - 2025

Melhorando

medeDistanciaCMPol

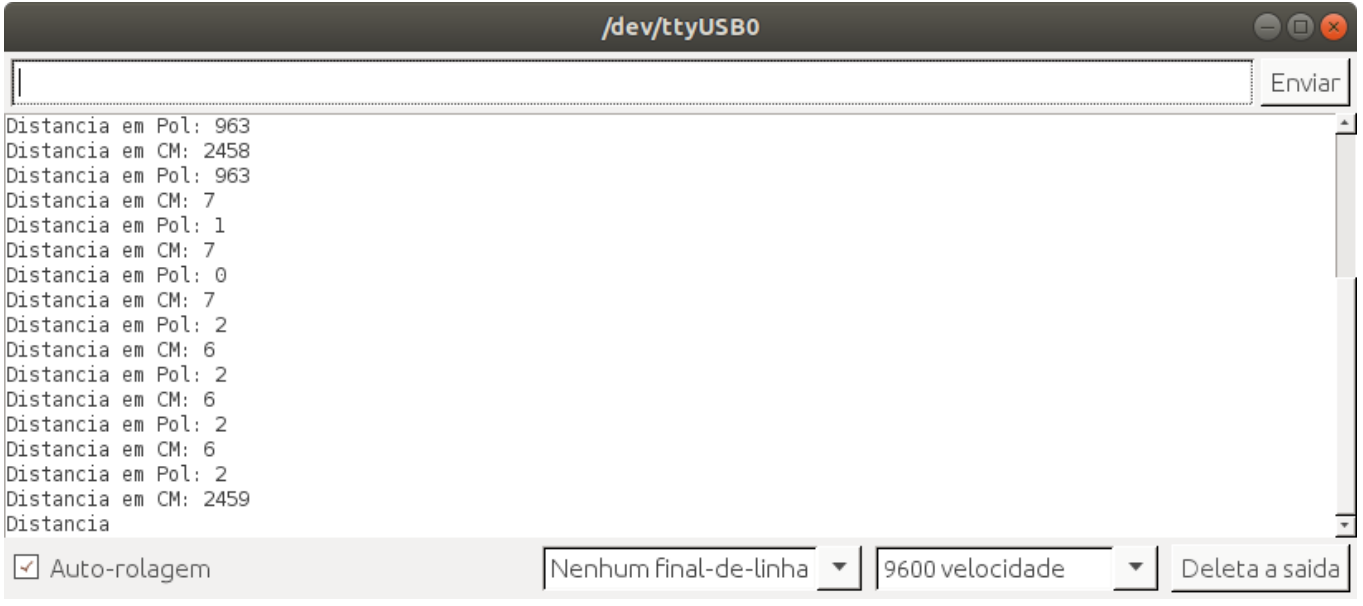
```
#include <Ultrasonic.h>

#define trig 4
#define echo 3

Ultrasonic ultrasonic(trig, echo);

void setup()
{
  pinMode(echo, INPUT);
  pinMode(trig, OUTPUT);
  Serial.begin(9600);
  Serial.println("Iniciando...\n\n");
}

void loop()
{
digitalWrite(trig, LOW);
delayMicroseconds(2);
digitalWrite(trig, HIGH);
delayMicroseconds(10);
digitalWrite(trig, LOW);
int centimetros = (ultrasonic.Ranging(CM));
int polegadas = (ultrasonic.Ranging(INC));
Serial.print("Distancia em CM: ");
Serial.println(centimetros);
Serial.print("Distancia em Pol: ");
Serial.println(polegadas);
delay(2000);
}
```



Imagens: cópia de tela realizada pelo autor

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111
simao@ufpr.br - 2025

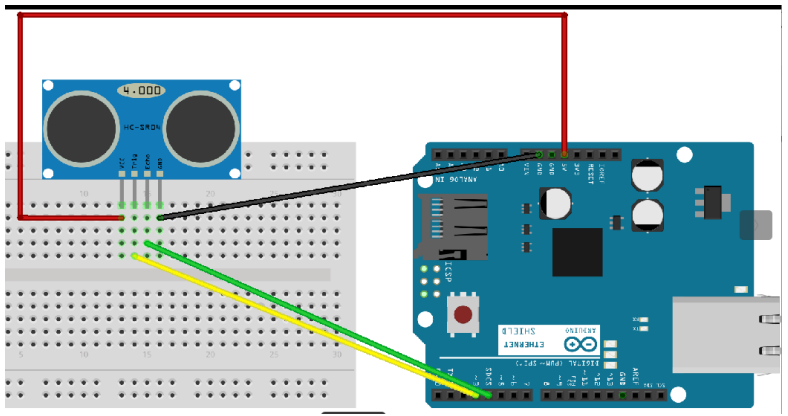
1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Sem biblioteca

- Se você tiver informações a respeito do módulo que está utilizando (ou se você desenvolveu o módulo…), poderá controlar suas características diretamente, sem utilizar uma biblioteca.
  - Em geral as características do módulo ou de seu circuito integrado de controle estão disponíveis na forma de *datasheets* (folhas de dados)

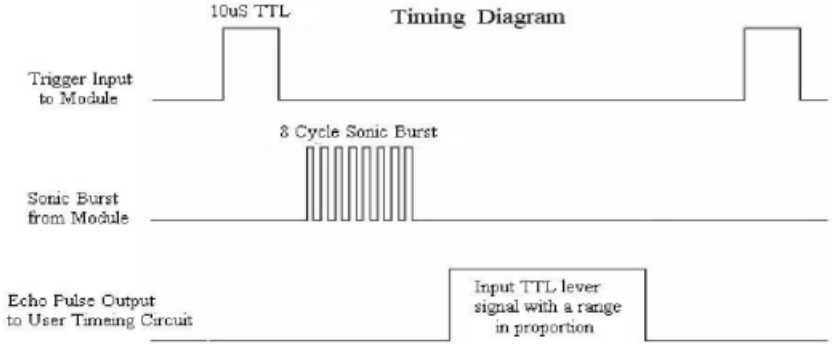
simao@ufpr.br - 2025

Ultrassom



```
1 int pinoTrigger = 3;
2 int pinoEcho = 4;
3
4 void setup()
5 {
6   Serial.begin(9600);
7   pinMode(pinoTrigger, OUTPUT);
8   pinMode(pinoEcho, INPUT);
9 }
```

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

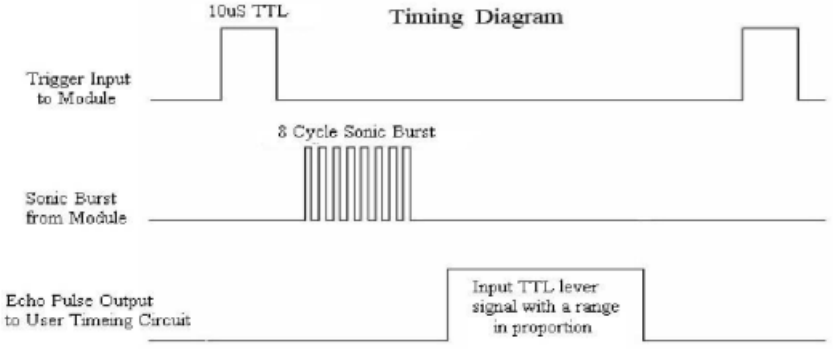
simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Ultrassom

```
11 float mede()
12 {
13   digitalWrite(pinoTrigger,LOW);
14   delayMicroseconds(3);
15   digitalWrite(pinoTrigger,HIGH);
16   delayMicroseconds(10);
17   digitalWrite(pinoTrigger,LOW);
18   float tempoUs = pulseIn(pinoEcho, HIGH);
19   return (tempoUs / 58);
20 }
21
22 void loop()
23 {
24   float distancia = mede();
25   Serial.print("Distancia medida: ");
26   Serial.print(distancia);
27   Serial.println(" centimetros");
28   delay(500);
29 }
```

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

E se ao invés de um medidor de distânica você quisesse um medidor de altura de uma pessoa?

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

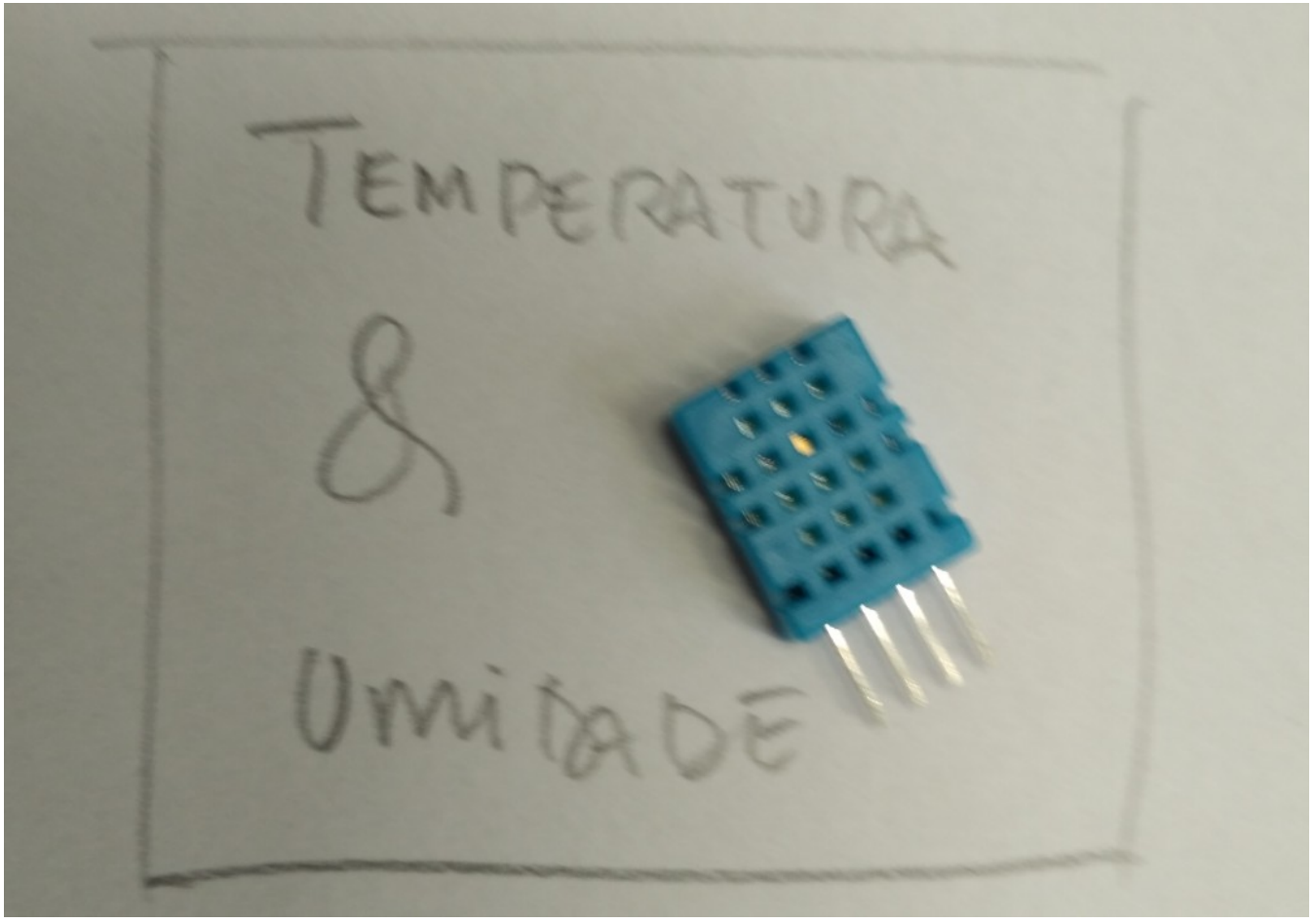


Imagem: fotos do autor

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

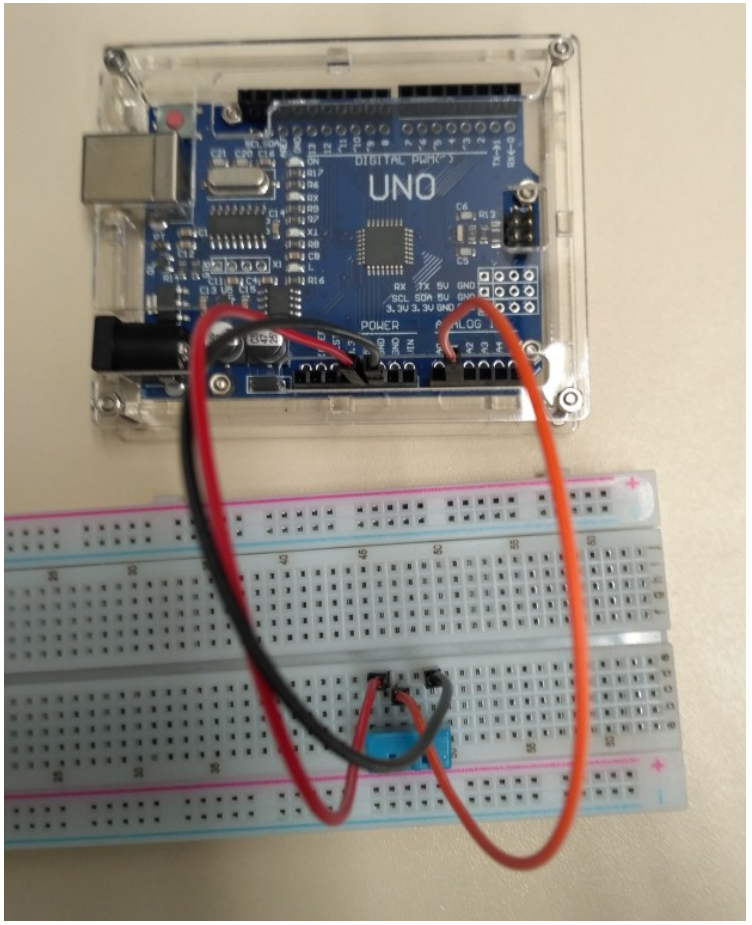
O que você precisará?

- Arduino configurado na IDE
- Uma placa de experimentação e fios
- Um sensor de temperatura e umidade DHT-11

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

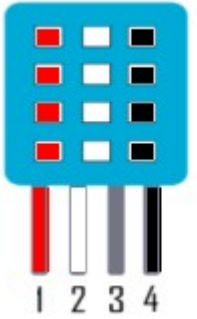
Ligação



O pino VCC do DHT11 é o pino 1, e é ligado ao 5V da placa do Arduino.

O pino GND do DHT11 é o pino 4, e é ligado ao GND da placa do Arduino.

O pino de saída do DHT11 é ligado na entrada analógica A1 da placa do Arduino.



Pinos do Sensor DHT-11

1 = VCC
2 = Saída
3 = sem uso
4 = GND

Imagem: fotos do autor

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Atenção

Imagem: fotos do autor

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Sequência

- Desligue o cabo USB
- Ligue os componentes na placa
- Conecte os fios à placa do Arduino; anote as portas utilizadas
- Elabore o código

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

tempUmidadeDHT11

```
#include "DHT.h"

#define sensor A1
#define tipo DHT11

DHT dht(sensor, tipo);

void setup()
{
  Serial.begin(9600);
  Serial.println("Iniciando...\n\n");
  dht.begin();
}

void loop()
{

  float temperatura = dht.readTemperature();
  float umidade = dht.readHumidity();

  if (isnan(temperatura) || isnan(umidade))
  {
    Serial.println("Comunicação falhou!\n");
  }
  else
  {
    Serial.print("Temperatura = ");
    Serial.print(temperatura);
    Serial.print(" C, e umidade = ");
    Serial.print(umidade);
    Serial.println(" %");

    delay(2000);
  }
}
```

Código

Quem faz todo o trabalho agora é a biblioteca incluída no código, “DHT.h”.

Para ver o resultado, vamos ligar o Monitor Serial.

Basta clicar no ícone à esquerda da tela:

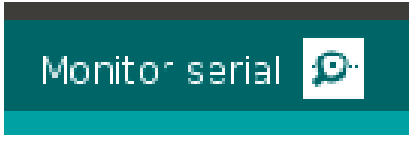


Imagem: cópia de tela realizada pelo autor

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Ferramentas / Monitor serial

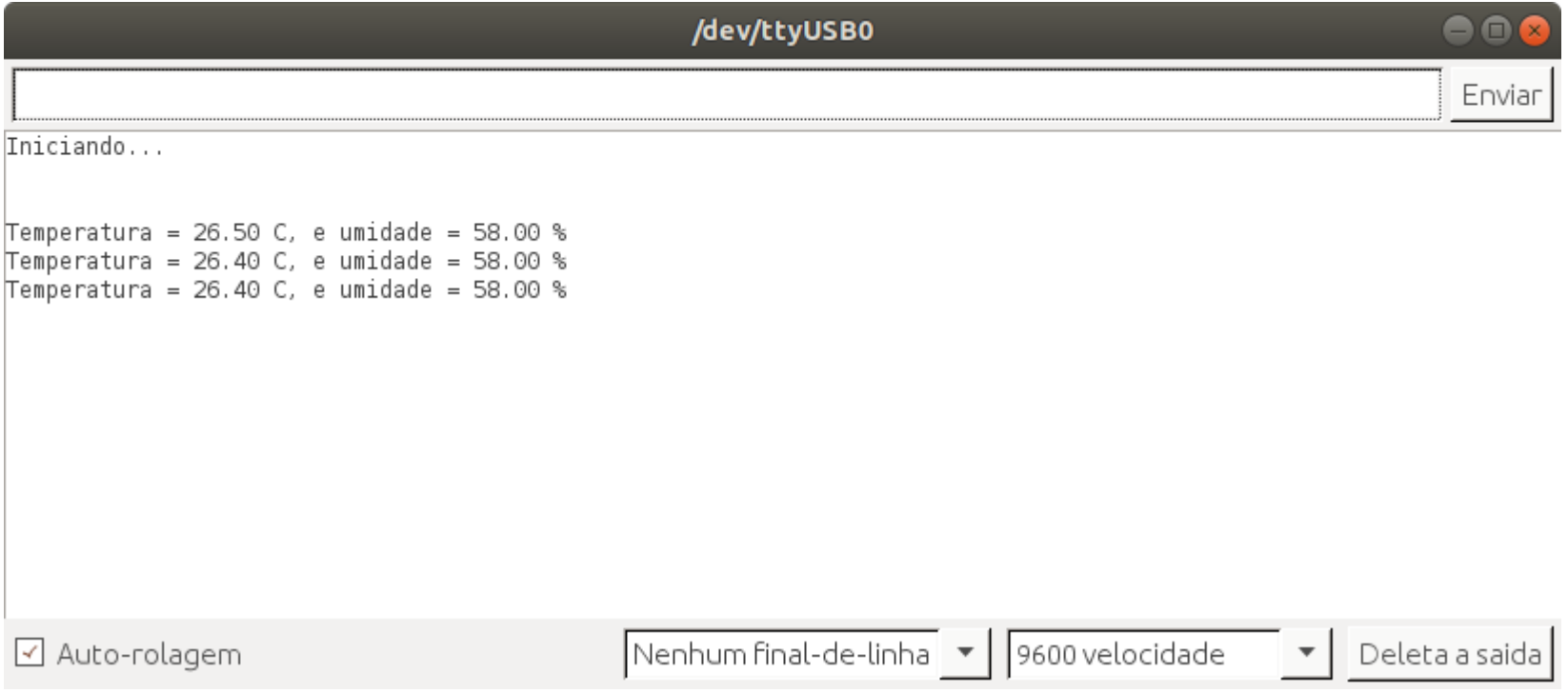


Imagem: cópia de tela realizada pelo autor

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Sem biblioteca

- Também é possível ligar este sensor sem usar uma biblioteca.
  - Pesquise…

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 1010011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Dúvidas

Se tiver dúvidas do que foi desenvolvido até o momento, tire-as agora, antes de prosseguirmos.

Caso contrário, vamos em frente...

simao@ufpr.br - 2025

1010000 0100000 1110010 0100000 1101111 0100000 1100110 0100000 10110011 0100000 1101001 0100000 1101101 0100000 1100001 0100000 1101111

Parabéns!

simao@ufpr.br - 2025