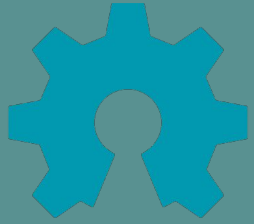
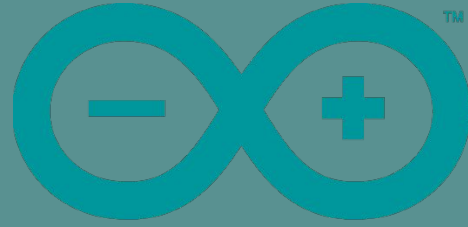


03 de junho de 2019



open source  
hardware



ARDUINO



open source

# CURSO ARDUÍNO PRÁTICO

Instrutores: Rafael e Ruben

# Simplificando o processo de um computador...



Entrada



Processo

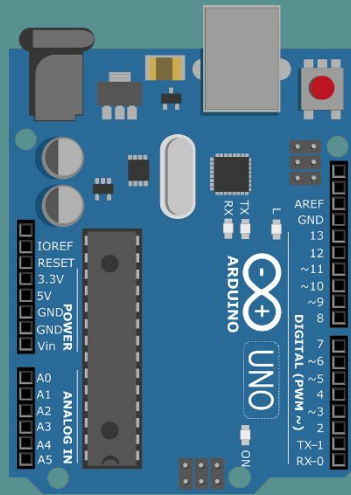


Saída

... para o arduino.



Entrada



Processo



Saída

Sensor

>

Processo

>

Atuador

# Portas e Componentes

Impede que a USB do computador seja danificada em caso de sobrecorrente. (acima de 500 mA)

Regula tensão DC para 5V.

Conector USB tipo B

Botão de Reset

Conjunto microcontrolador e cristal que faz a interface USB com o computador

Conector para gravação ICSP, do ATMEGA16U2

Led conectado ao pino 13 do arduino

Leds de status da comunicação serial Entre placa e computador

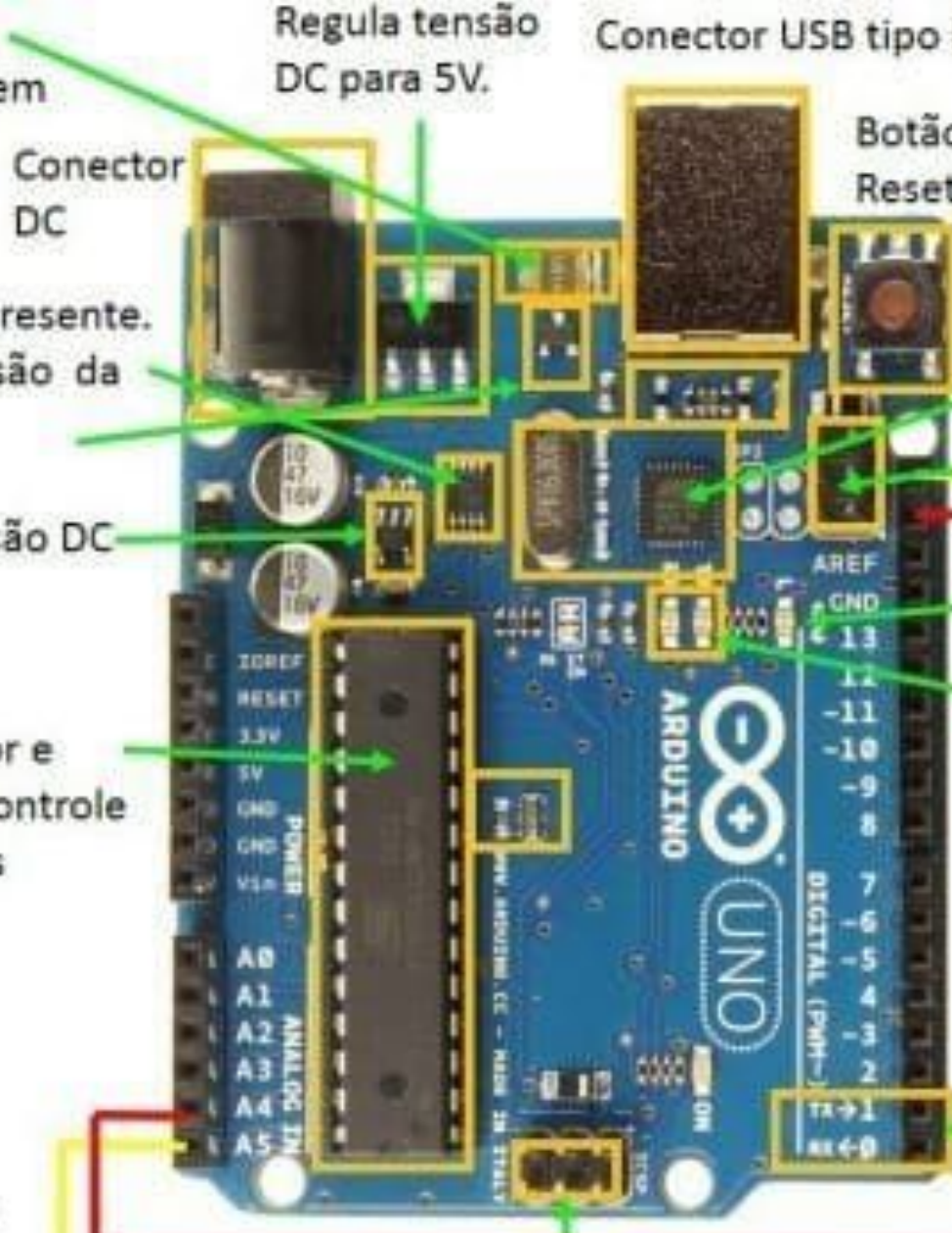
Os sinais em amarelo e vermelho Indicam dois pinos que estão em curto

Caso utilize esses sinais no projeto, tome cuidado pois estão conectados ao outro microcontrolador para

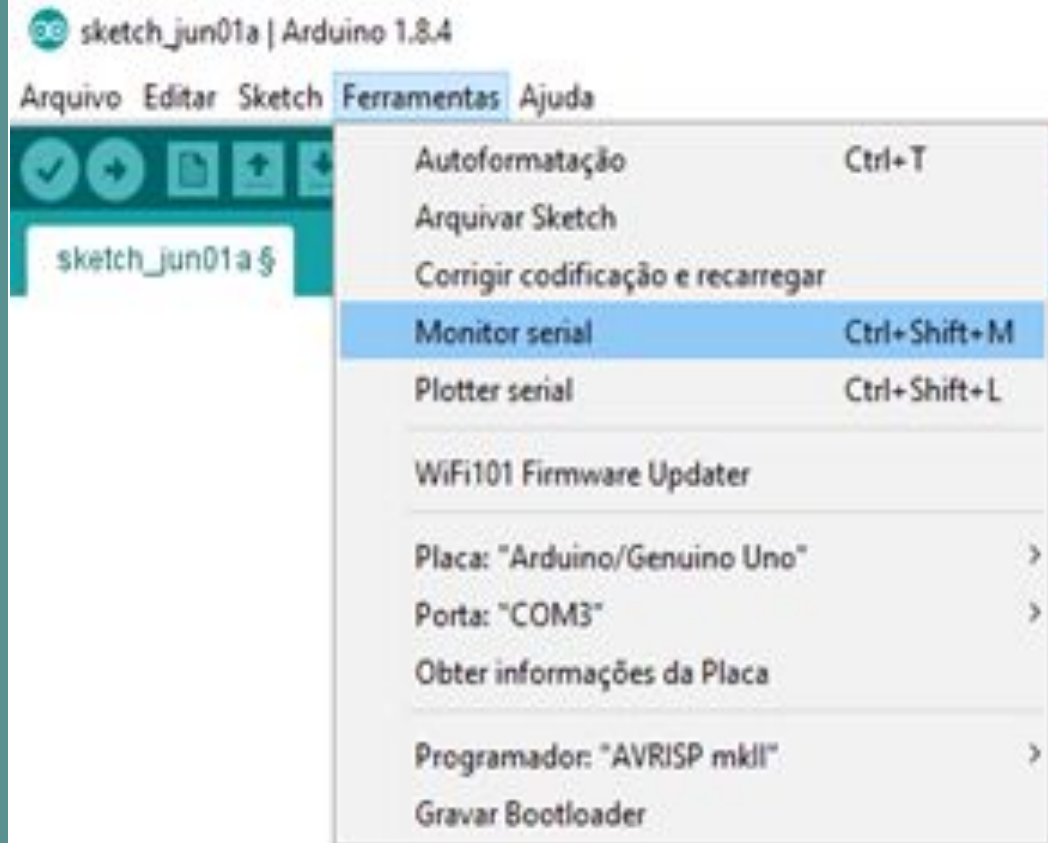
Conector DC

Regula a tensão DC para 3,3 V.

Conjunto microcontrolador e cristal, responsável pelo controle e leitura de todos os pinos da placa.



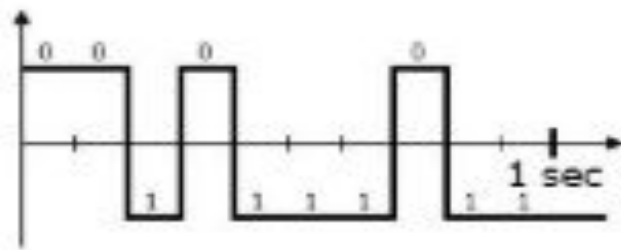
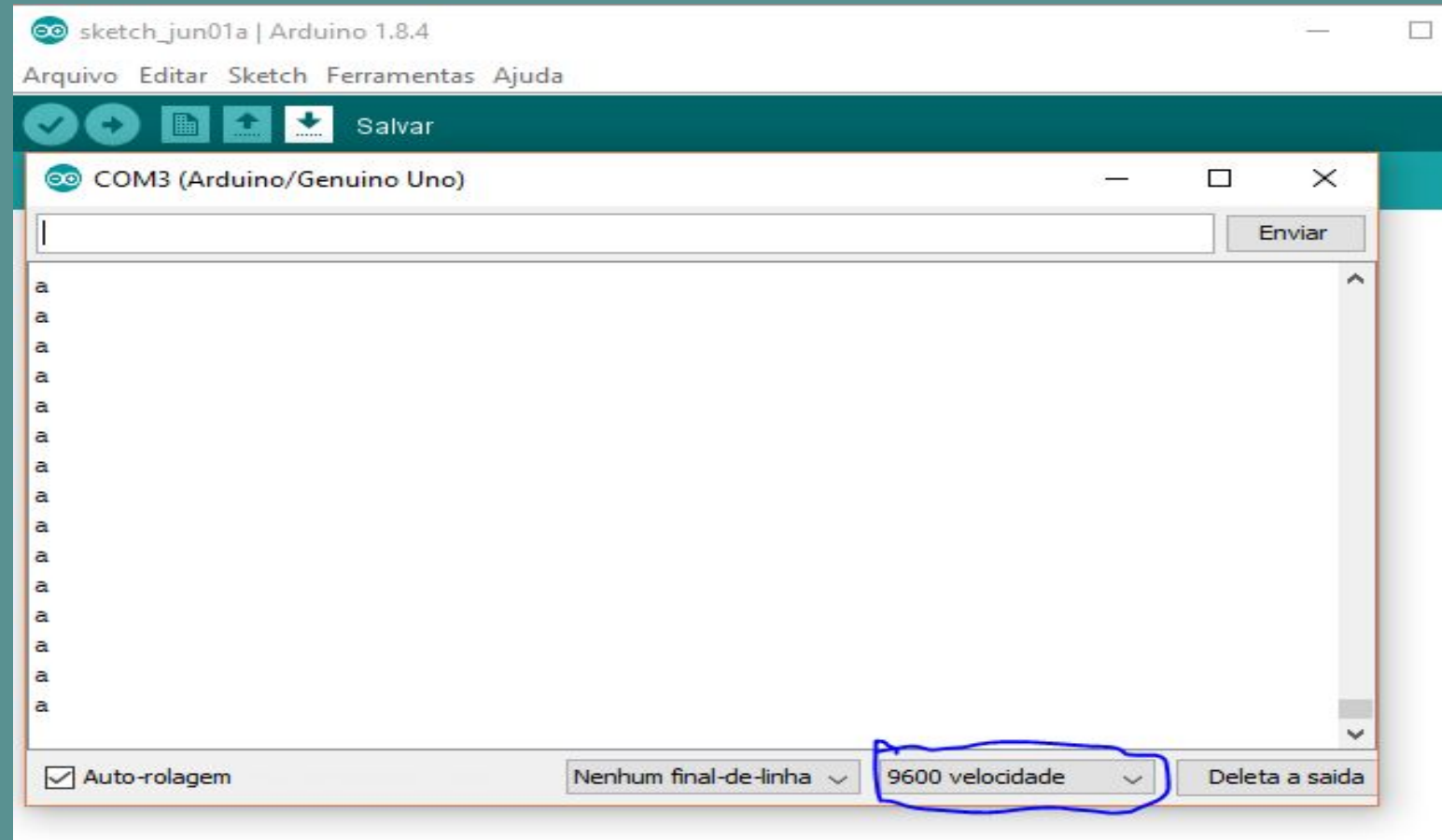
# Monitor Serial



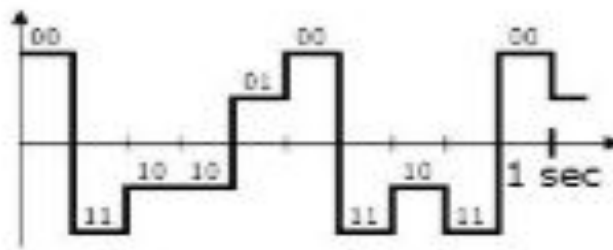


# Monitor Serial

Baud rate  
vs  
Bit rate



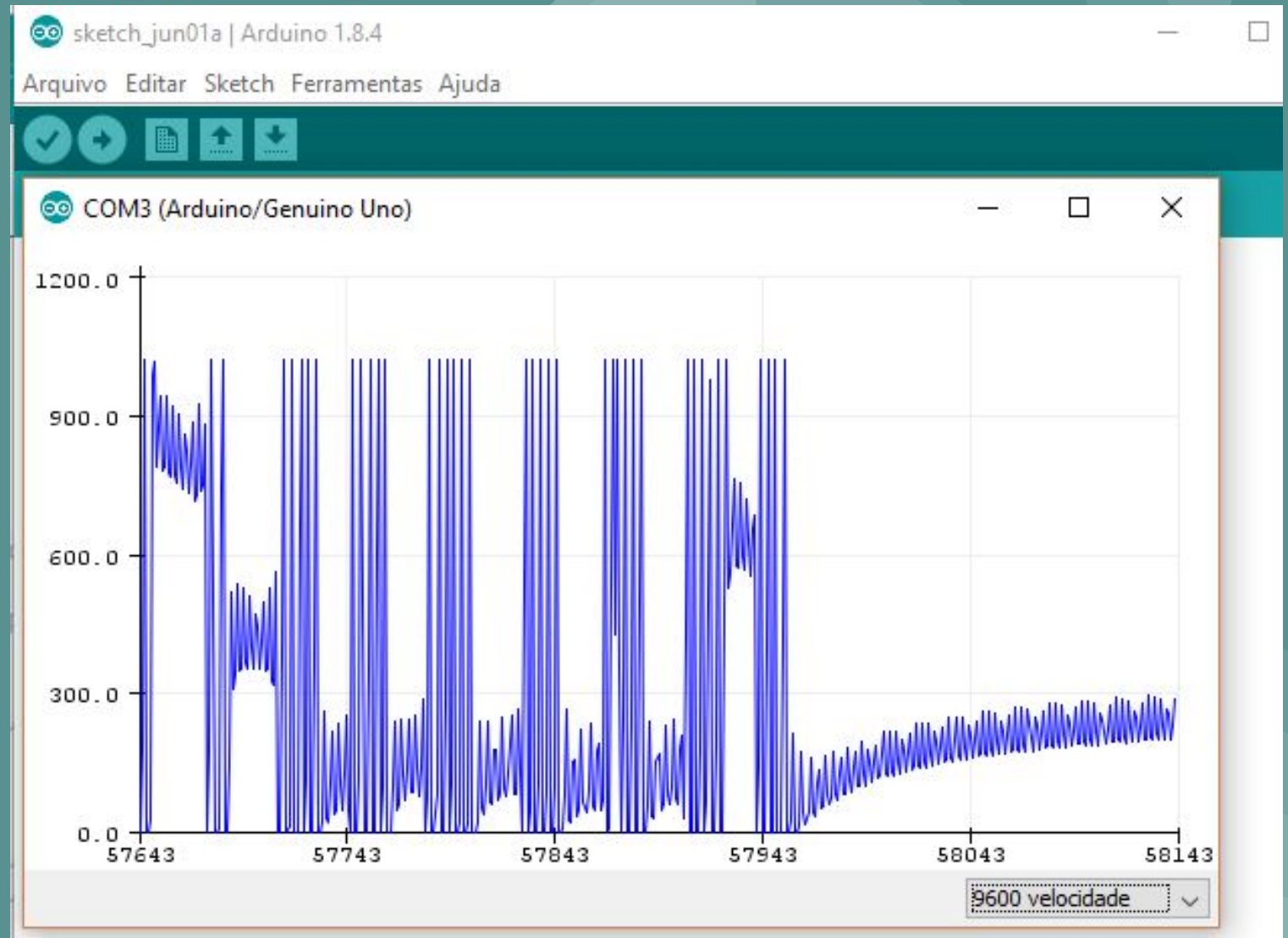
Baud = 10  
Bit rate = 10 bps



Baud = 10  
Bit rate = 20 bps

# Monitor Serial

## Plotter Serial



# Monitor Serial

Vejamos na prática!



The screenshot shows the Arduino IDE interface for a sketch named 'sketch\_jun01a' using Arduino 1.8.4. The menu bar includes 'Arquivo', 'Editar', 'Sketch', 'Ferramentas', and 'Ajuda'. Below the menu bar is a toolbar with icons for checking, running, opening a file, uploading, and downloading. The main text area contains the following C++ code:

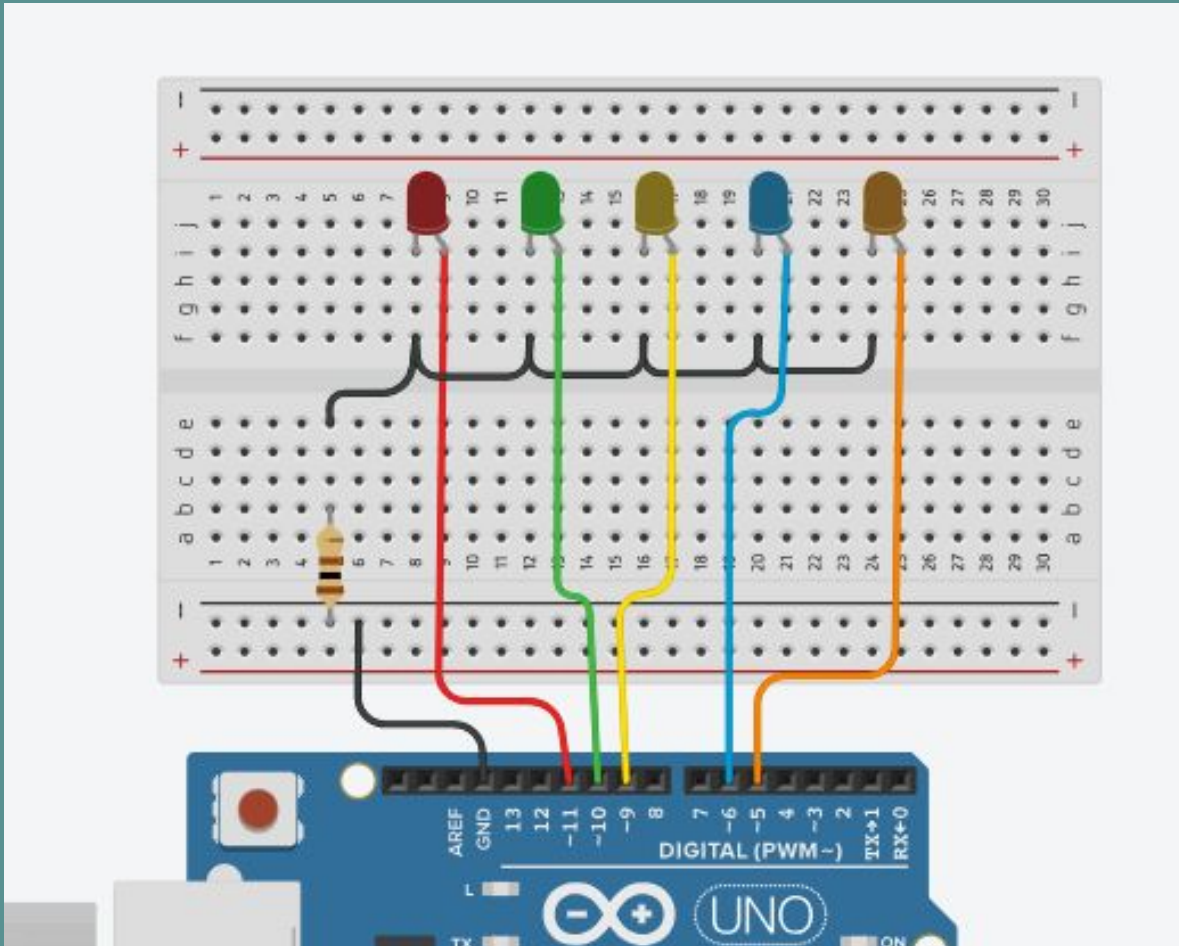
```
sketch_jun01a $  
  
char entrada;  
  
void setup()  
{  
  //Definindo a taxa de transmissão como 9600 bps  
  Serial.begin(9600);  
}  
  
void loop()  
{  
  //Apresente os resultados enquanto houver dados para leitura  
  while (Serial.available() > 0)  
  {  
    // Lê byte da serial  
    entrada = Serial.read();  
    Serial.println(entrada);  
  }  
}
```



Deixando os monitores Serial em “stand by”, vamo para um Show de Luzes!

**Projeto 01 – Show de luzes**

# Monte o esquema:



# Componentes:

- 1 – resistor 100 Ohms
- 5 - LEDS

```
int arrayLed[] = {11, 10, 9, 6, 5};
int tamanhoArray = sizeof(arrayLed) / sizeof(arrayLed[0]);
```

```
void apaga()
{
    for (int i = 0; i < tamanhoArray; i++)
    {
        digitalWrite(arrayLed[i], LOW);
    }
}
```

```
void acende()
{
    for (int i = 0; i < tamanhoArray; i++)
    {
        digitalWrite(arrayLed[i], HIGH);
    }
}
```

```
void trocarEstadoLed(char estado)
{
    switch (estado)
    {
        case '0':
            apaga();
            break;
        case '1':
            acende();
            break;
        case '2':
            sequencial();
            break;
    }
}
```

```
void sequencial()
{
    for (int i = 0; i < tamanhoArray; i++)
    {
        digitalWrite(arrayLed[i], HIGH);
        delay (150);
        digitalWrite(arrayLed[i], LOW);
        delay (150);
    }
}
```

```
void setup()
{
    for (int i = 0; i < tamanhoArray; i++)
    {
        pinMode(arrayLed[i], OUTPUT);
    }
}
```

```
void loop()
{
    trocarEstadoLed(1);
}
```

**setup()** – Parte do código dedicado para iniciar o programa. Dizer para o arduino com quais portas ele deve trabalhar, além de iniciar algumas funções e modos de trabalho. Esta parte é lida uma única vez.

**loop()** – Parte do código dedicado ao trabalho repetitivo. Esta parte roda infinitas vezes.

# Função pinMode()

- ▶ Sintaxe : `pinMode(PORTA, FORMATO);`
- ▶ As portas podem ser as portas analógicas A0,A1,A2..A7, ou digital 0,1,2,3...13.
- ▶ O formato pode ser INPUT para sinais que chegam ao arduino, e OUTPUT para sinais enviados pelo arduino.



# Função digitalWrite()

- ▶ Serve para enviar um sinal digital, verdadeiro ou falso...5v ou 0v.
- ▶ Sintaxe : digitalWrite(PORTA, FORMATO);
- ▶ As portas podem ser as portas analógicas A0,A1,A2..A7, ou digital 0,1,2,3...13.
- ▶ O formato pode ser HIGH (ou 1) para enviar 5V, e LOW (ou 0) para enviar 0V.

# Função delay()

- ▶ Serve para espera um tempo em milisegundos.
- ▶ Sintaxe : `delay(tempo);`
- ▶ O tempo é dado em milissegundos.

**Vamos voltar para o Monitor  
Serial?**

**Projeto 1.1 – Controlando o  
Show de luzes pela Serial.**

# Adicionando alguns códigos ao anterior

- Variável global: `char entrada;`

Atualizando a loop():

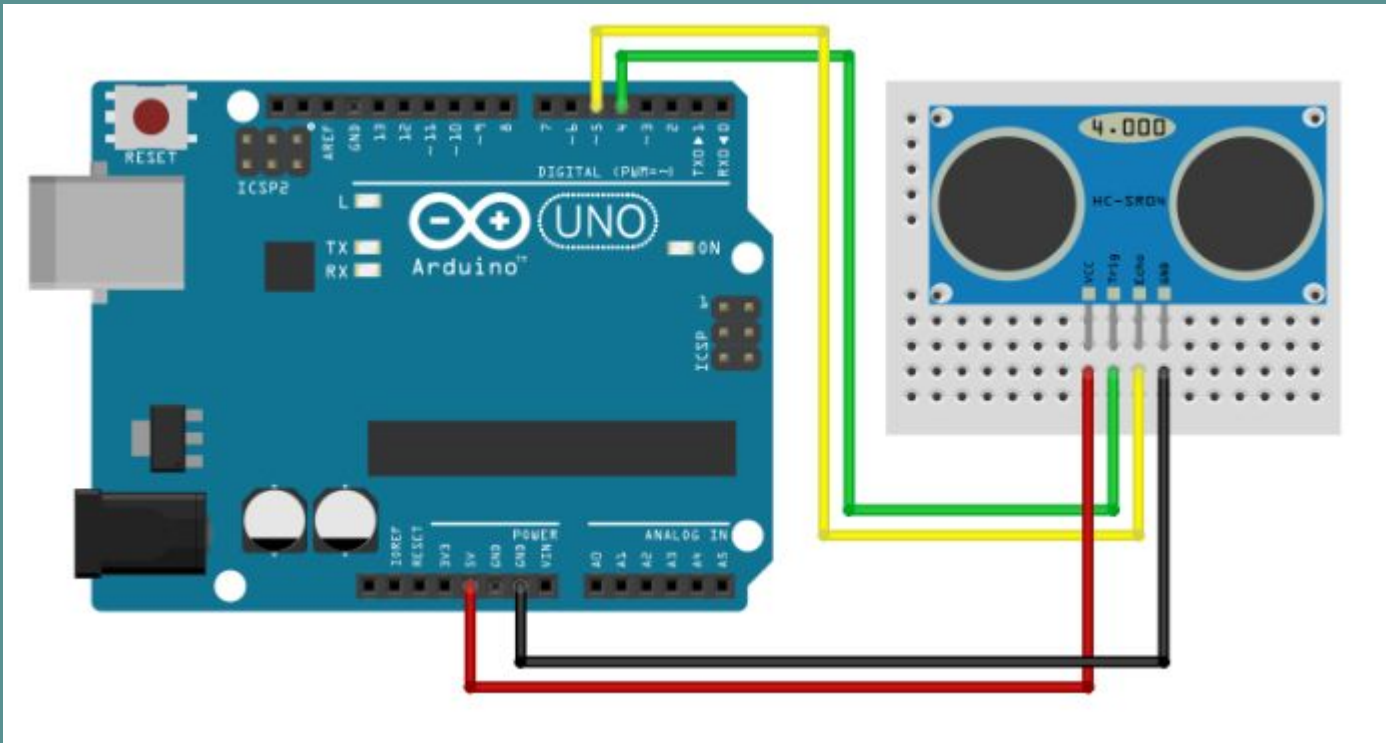
```
void loop()
{
    if (Serial.available() > 0)
    {
        // Lê byte da serial
        entrada = Serial.read();
        Serial.println(entrada);
    }
    trocarEstadoLed(entrada);
}
```

**MELHOR FORMA DE  
APRENDER É PRATICANDO.**

## **Projeto 02 – Sensor ultrassônico**



# Monte o esquema:



# Componentes:

- 1 – Sensor Ultrassônico
- Library: <https://github.com/ErickSimoes/Ultrasonic>

# DIGITE O CÓDIGO NA IDE

```
1 /**
2  * Leitura de distância com o sensor HC - SR04
3  * Library used: https://github.com/ErickSimoes/Ultrasonic
4  */
5 #include <Ultrasonic.h>
6
7 /*
8  * Definindo o nome do sensor (ultrassom) - um objeto e
9  * especificando as portas de conexão trig(8) e echo(7)
10  * respectivamente.
11  */
12 Ultrasonic ultrassom(8,7);
13
14 //Função setup() roda apenas 1 vez quando a placa é ligada ou resetada
15 void setup()
16 {
17     //Habilitando a comunicação serial a uma taxa de 9600 bauds.
18     Serial.begin(9600);
19 }
20 //Definindo a variável (global) que receberá a distância
21 long distancia;
22
23 //Função loop(): executa um loop infinitamente enquanto a placa estiver ligada.
24 void loop()
25 {
26     distancia = ultrassom.read(CM); //ultrassom.read(CM) retorna a distância em
27                                     //centímetros (CM)
28     Serial.print(distancia); //imprime o valor da variável distancia
29     Serial.println(" cm");
30
31     delay(100); //Aguarda 100ms antes de executar o loop novamente
32 }
```

Lembra do  
Plotter?

Filtros são usados para reduzir ruídos vindos de sinais.

Vamos praticar?!

## PROJETO 3 – FILTRO LINEAR

# MODELO DE CÓDIGO

```
#define N_INTERACAO 700
int filtroLinear(int porta)
{
    unsigned long sinalFiltrado = 0;

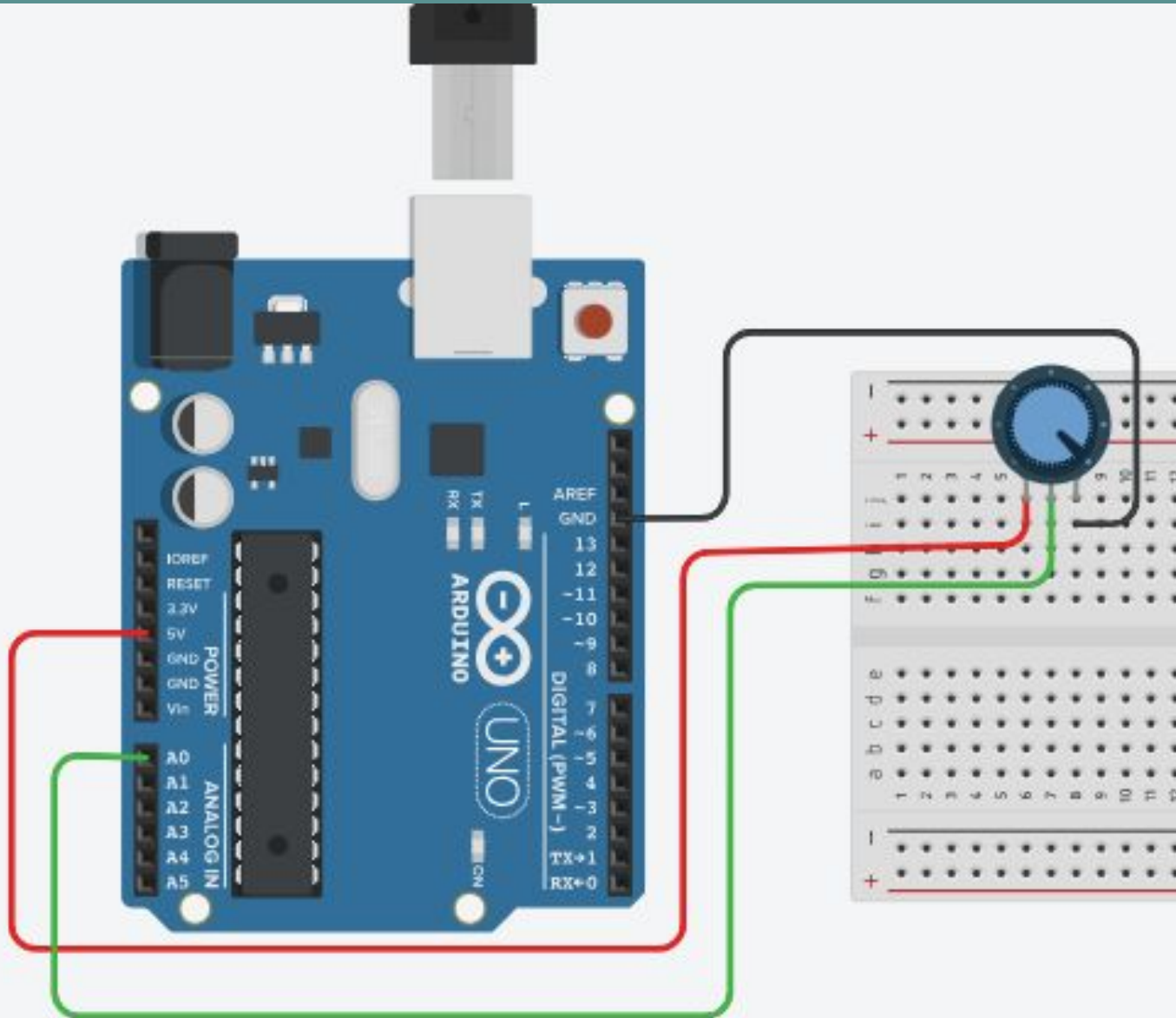
    for (int i = 0; i < N_INTERACAO; i++)
    {
        sinalFiltrado += analogRead(porta);
    }
    return sinalFiltrado / N_INTERACAO;
}
```

Experimente passar o filtro  
linear no sinal lido por um  
potenciômetro

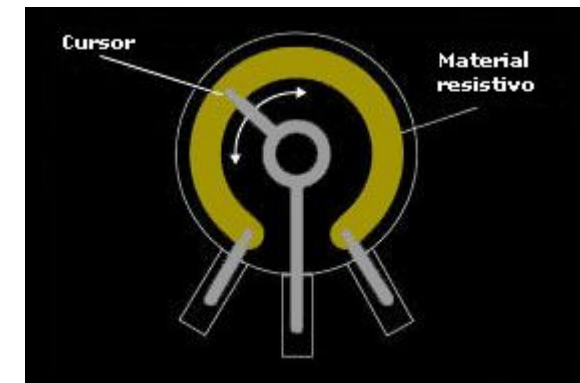
## PROJETO 3.1 – FILTRO LINEAR COM POTENCIÔMETRO



# SIGA O ESQUEMA:



```
1  #define POT A0
2
3  void setup()
4  {
5      Serial.begin(9600);
6      pinMode(POT, INPUT);
7  }
8
9
10 void loop()
11 {
12     int valor = 0;
13     valor = analogRead(POT);
14     Serial.print("valor : ");
15     Serial.print(valor);
16 }
```



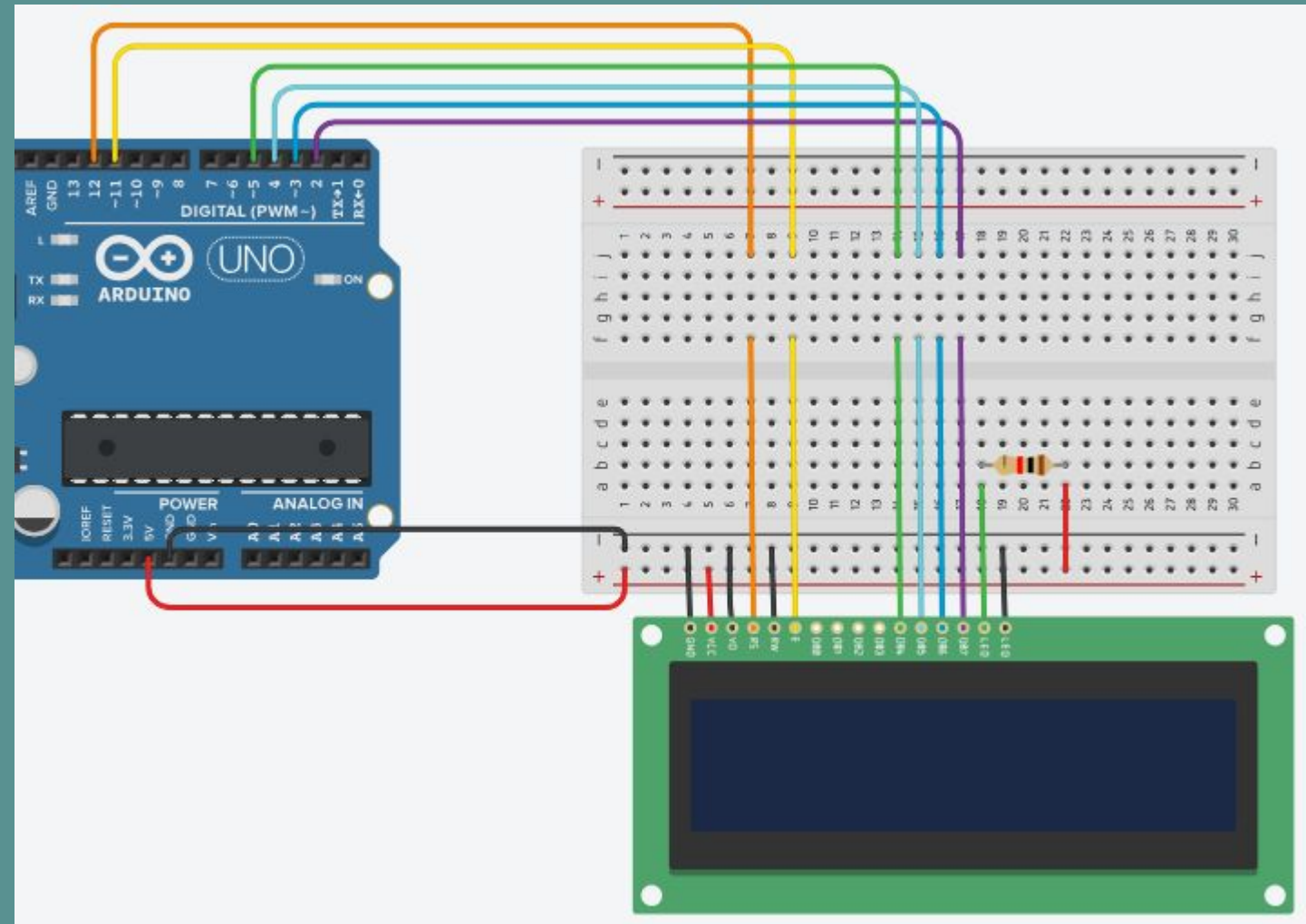
Vamos aprender a utilizar o  
LCD (Light Cristal Display)



PROJETO 4 – LCD

# Monte o esquema:

## Descrição dos pinos



Pino	Nome	Função
1	Vss	Terra
2	Vdd	Positivo (normalmente 5V)
3	Vo	Contraste do LCD. Às vezes também é chamado de Vee
4	RS	Register Select
5	R/W	Read/Write
6	E	Enable
7	D0	Bit 0 do dado a ser escrito no LCD (ou lido dele).
8	D1	Bit 1 do dado a ser escrito no LCD (ou lido dele).
9	D2	Bit 2 do dado a ser escrito no LCD (ou lido dele).
10	D3	Bit 3 do dado a ser escrito no LCD (ou lido dele).
11	D4	Bit 4 do dado a ser escrito no LCD (ou lido dele).
12	D5	Bit 5 do dado a ser escrito no LCD (ou lido dele).
13	D6	Bit 6 do dado a ser escrito no LCD (ou lido dele).
14	D7	Bit 7 do dado a ser escrito no LCD (ou lido dele).
15	A	Anodo do back-light (se existir back-light).
16	K	Catodo do back-light (se existir back-light).

Tabela 1: Descrição das funções dos pinos do LCD

# DIGITE O CÓDIGO :

```
1 // Inclue a biblioteca:
2 #include <LiquidCrystal.h>
3
4 // Inicializa o LCD
5 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
6
7 void setup() {
8     // Inicia o LCD com suas dimensões
9     lcd.begin(16, 2);
10    // Imprime uma mensagem no LCD
11    lcd.print("hello, world!");
12 }
13
14 void loop() {
15     //Coloca o cursor na coluna 1 e linha 0
16     lcd.setCursor(0, 1);
17     // Imprime o tempo de operação
18     lcd.print(millis() / 1000);
19 }
20
```

Algo de uso mais prático...



PROJETO 4.1 – LETREIRO  
ELETRÔNICO



# NO CÓDIGO ANTERIOR, INCLUA E FAÇA USO DESTA FUNÇÃO:

```
String lerSerial ()
{
    String conteudo1 = "";
    char entrada1;

    // Se receber algo pela serial
    while (Serial.available() > 0) {
        //Segue fazendo a leitura de cada byte enviado
        entrada1 = Serial.read();

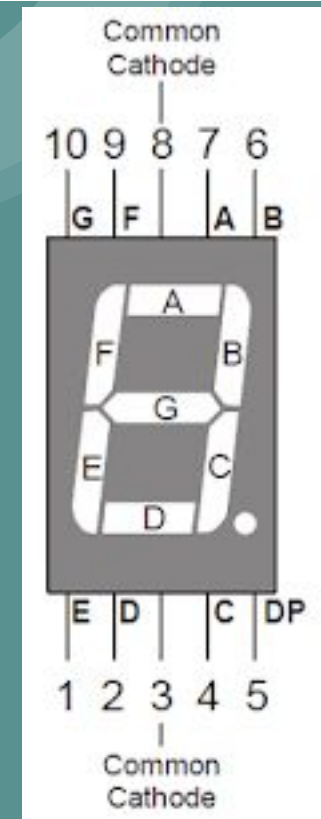
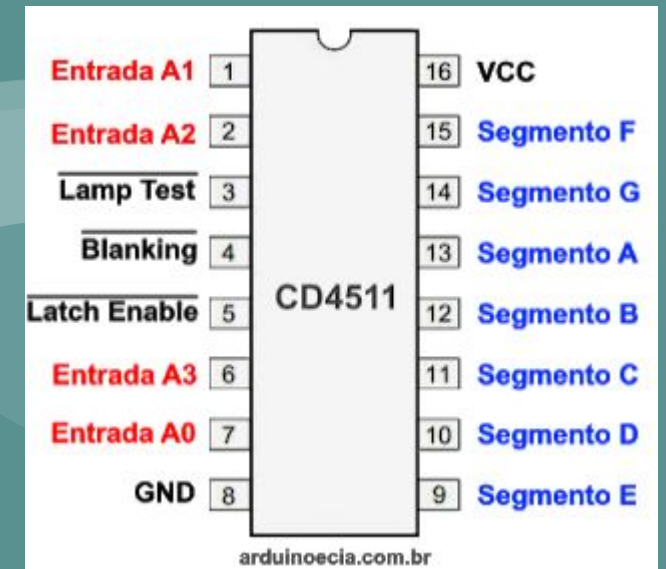
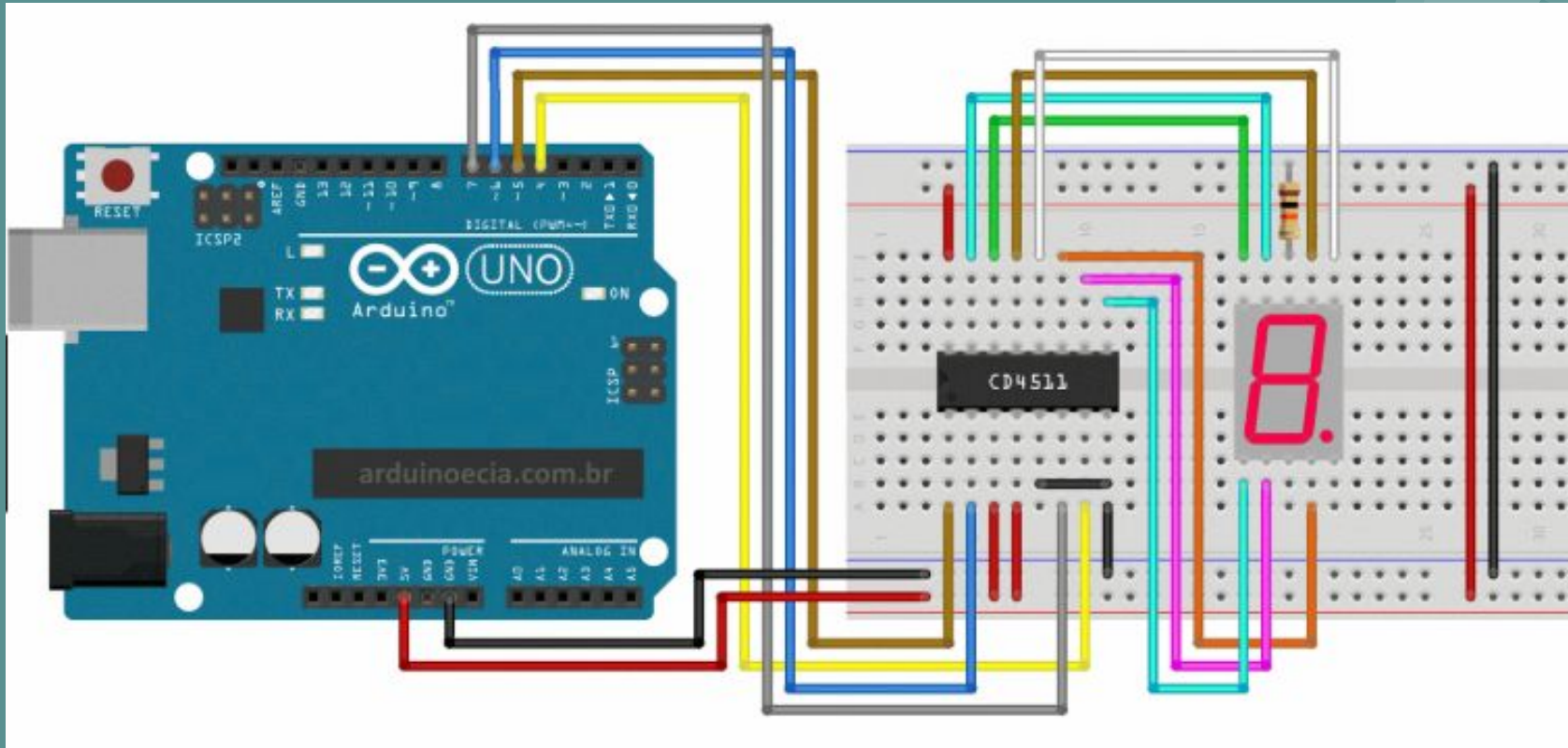
        //Ignora sinal/caractere de ENTER
        if (entrada1 != '\n') {
            conteudo1.concat(entrada1);
        }
        // Aguarda buffer serial ler próximo caractere
        delay(10);
    }
    //Quando não houver mais bytes a ler...
    if (Serial.available() <= 0) {
        return conteudo1;
    }
}
```

## USANDO CIRCUITO INTEGRADO



PROJETO 5 – Led de 7  
segmentos com CI  
Decodificador

# SEGUIE O ESQUEMA



# O CÓDIGO

```
//Definicao dos pinos de entrada me relação ao CI
#define PinoA0 4
#define PinoA1 5
#define PinoA2 6
#define PinoA3 7
void setup()
{
    Serial.begin(9600);
    //Define os pinos como saída
    pinMode(PinoA0, OUTPUT);
    pinMode(PinoA1, OUTPUT);
    pinMode(PinoA2, OUTPUT);
    pinMode(PinoA3, OUTPUT);
}
void loop()
{
    Serial.print("Numero: 0 ");
    digitalWrite(PinoA0, LOW);
    digitalWrite(PinoA1, LOW);
    digitalWrite(PinoA2, LOW);
    digitalWrite(PinoA3, LOW);
    delay(1000);

    Serial.print("1 ");
    digitalWrite(PinoA0, HIGH);
    digitalWrite(PinoA1, LOW);
    digitalWrite(PinoA2, LOW);
    digitalWrite(PinoA3, LOW);
    delay(1000);
}
```

UMA CURIOSIDADE...



“RESETANDO” A  
PROGRAMAÇÃO VIA  
CÓDIGO

# O CÓDIGO

```
void (*reset)(void) = 0;
void setup() {
  Serial.begin(9600);
  Serial.println("Comando 1 Executado!");

  delay(1000);
  reset();
  Serial.println("Comando 2 Executado!");
}
void loop() {
}
```

Instrução que levará o controlador a ler a posição inicial da memória onde está o código a ser executado.

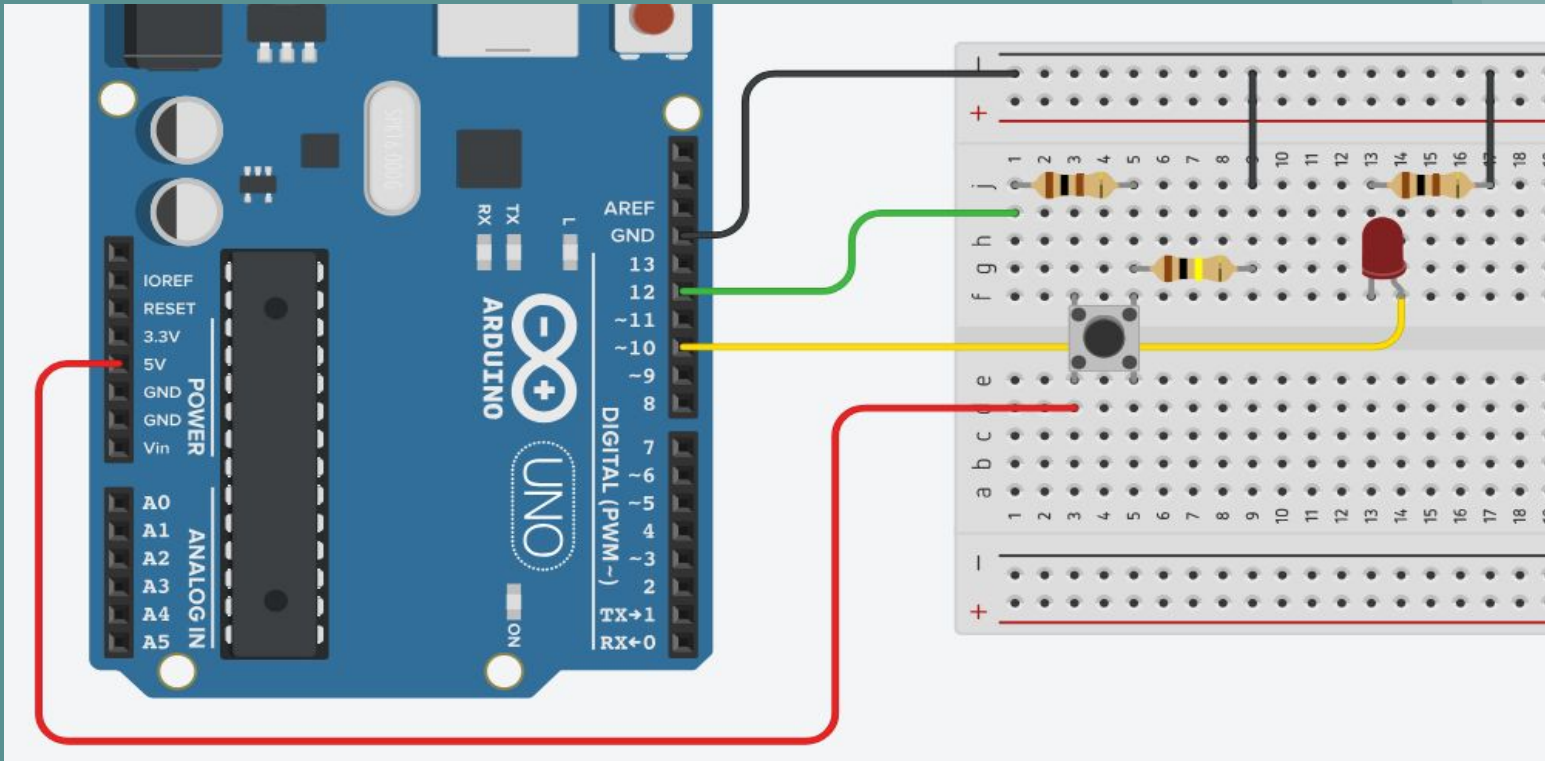
Vamos montar um esquema pull-down de um botão, e ele vai determinar se o nosso LED vai está ligado ou não.



## PROJETO 5 – BOTÃO



# ESQUEMA



# COMPONENTES

1 – BOTÃO (PUSHBUTTON)

2 – RESISTOR 100 OHM

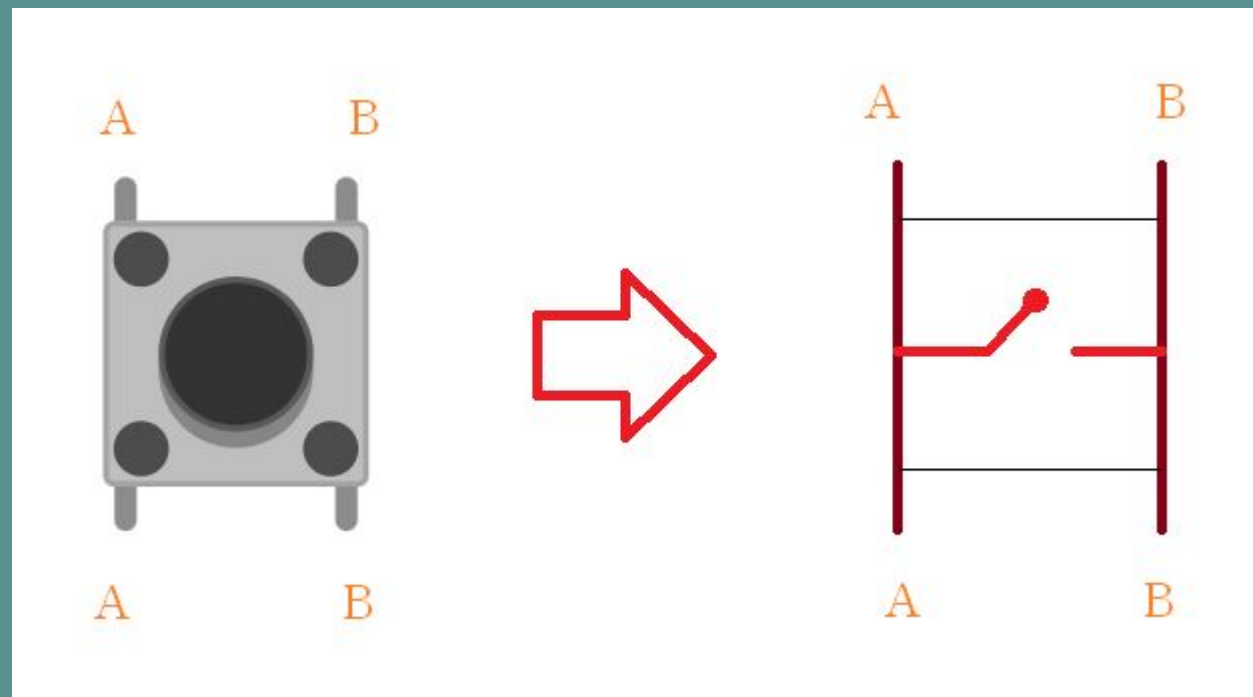
1- RESISTOR >10KOHMS



# DIGITE O CÓDIGO:

```
1  int botao = 12;
2  int led = 10;
3
4  void setup()
5  {
6      pinMode(botao, INPUT);
7      pinMode(led, OUTPUT);
8  }
9
10 void loop()
11 {
12     if(digitalRead(botao) == 1)
13     {
14         digitalWrite(led,HIGH);
15     }
16     else
17     {
18         digitalWrite(led,LOW);
19     }
20 }
```

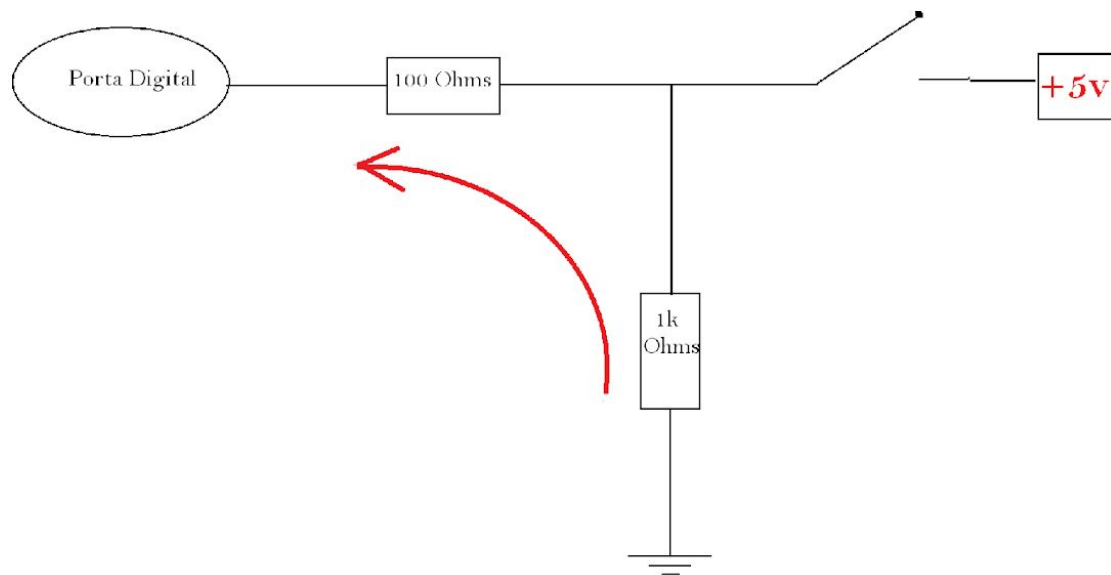
# DETALHES DO BOTÃO (PUSHBUTTON)



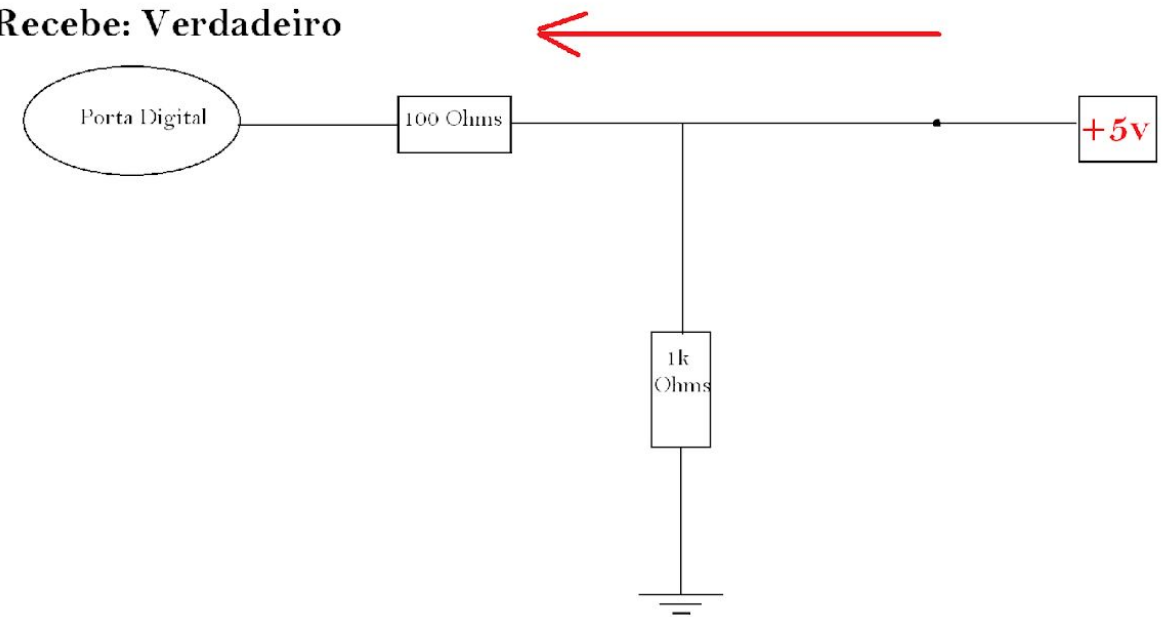
O botão é uma chave na qual quando é pressionado liga os pontos A e B.

# ESQUEMA PULL-DOWN

**Recebe: Falso**



**Recebe: Verdadeiro**



# Função digitalRead()

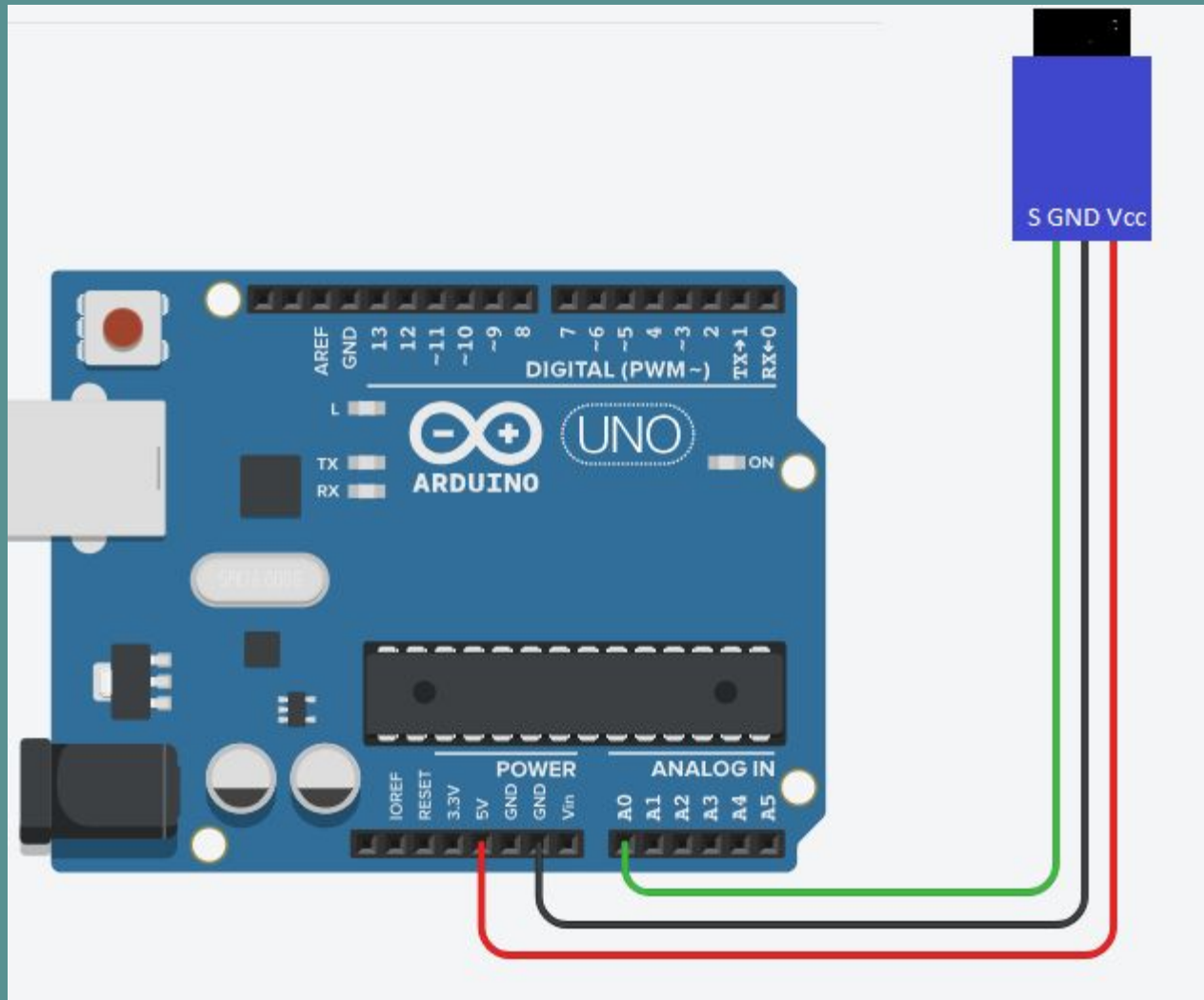
- ▶ Sintaxe : `digitalRead(PORTA);`
- ▶ Retorno: 1 caso tenha sinal e 0 caso não tenha sinal
- ▶ As portas podem ser as portas analógicas A0,A1,A2..A7, ou digital 0,1,2,3...13.
- ▶ O formato pode ser INPUT para sinais que chegam ao arduino, e OUTPUT para sinais enviados pelo arduino.

**Vamos aprender a trabalhar  
com um poderoso sensor de  
som.**



**PROJETO 6 – SENSOR DE SOM**

# ESQUEMA



# COMPONENTES

1 – SENSOR DE SOM

```
#define N_INTERACOES 400

void setup() {
  pinMode(A0, INPUT);
  Serial.begin(9600);
}

void loop()
{
  int valor = 0;

  valor = filtro(A0);
  Serial.print("Valor =");
  Serial.println(valor);
}

int filtro(int porta)
{
  unsigned long valorfiltrado = 0;
  for(int i =0; i< N_INTERACOES; i++)
  {
    valorfiltrado += analogRead(A0);
  }
  return valorfiltrado/N_INTERACOES;
}
```

# DIGITE O CÓDIGO:



A seguir, slides anteriores...



# DIGITE O CÓDIGO NA IDE

```
1 int pin[5] = {11,10,9,6,5};
2 void setup()
3 {
4     //Define os pinos como saídas
5     for(int i=0; i < 5; i++)
6     {
7         pinMode(pin[i], OUTPUT);
8     }
9 }
10 void loop()
11 {
12     apaga();
13     delay(1000);
14     acende();
15     delay(1000);
16 }
17 void apaga()
18 {
19     for(int i=0; i< 5 ;i++)
20     {
21         digitalWrite(pin[i], LOW);
22     }
23 }
24 void acende()
25 {
26     for(int i=0; i< 5 ;i++)
27     {
28         digitalWrite(pin[i], HIGH);
29     }
30 }
31
```

Outra opção:

```
for (int i=0; i< sizeof(pin) / sizeof(pin[0]); i++)
```