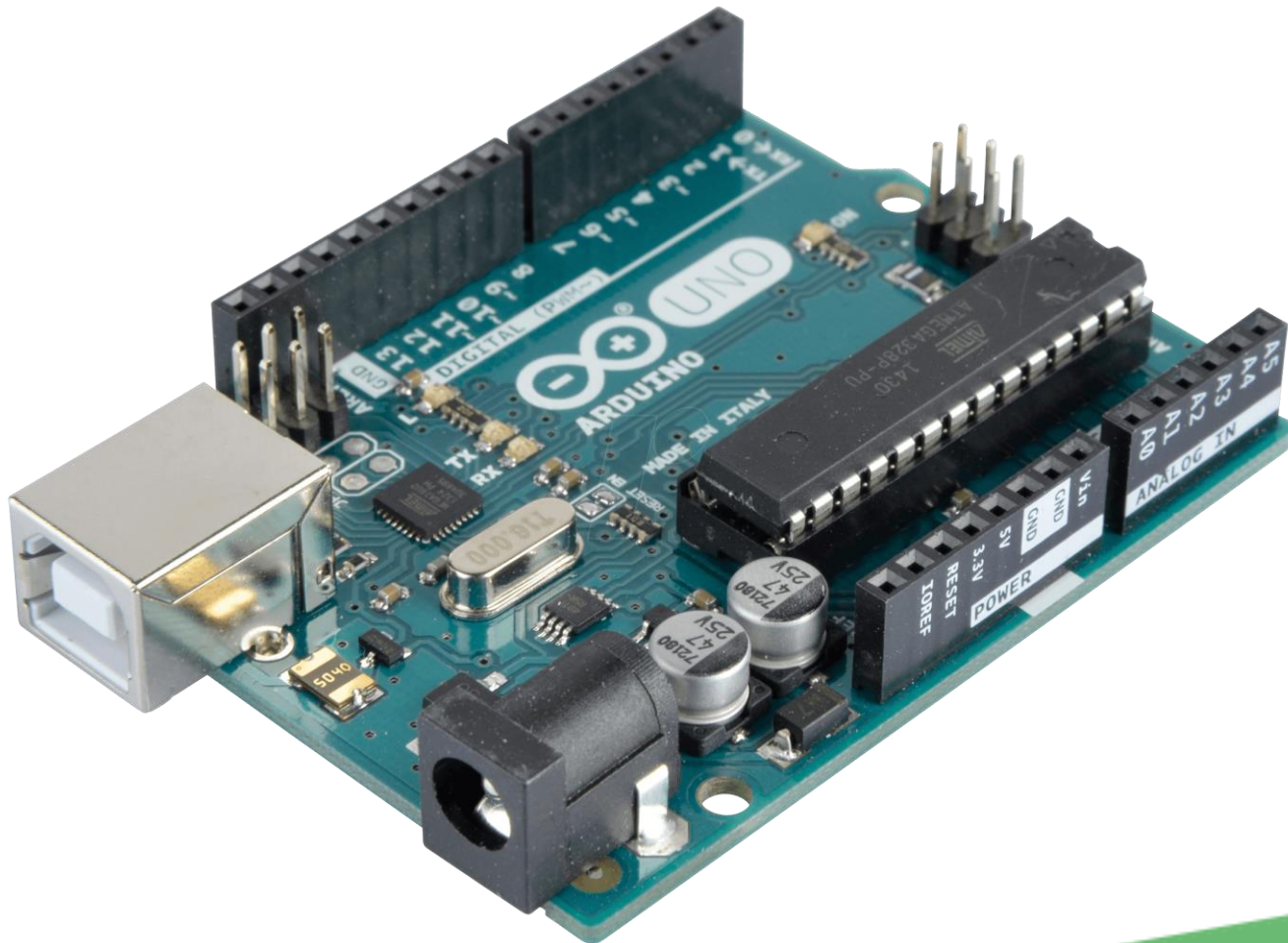




Workshop Arduino

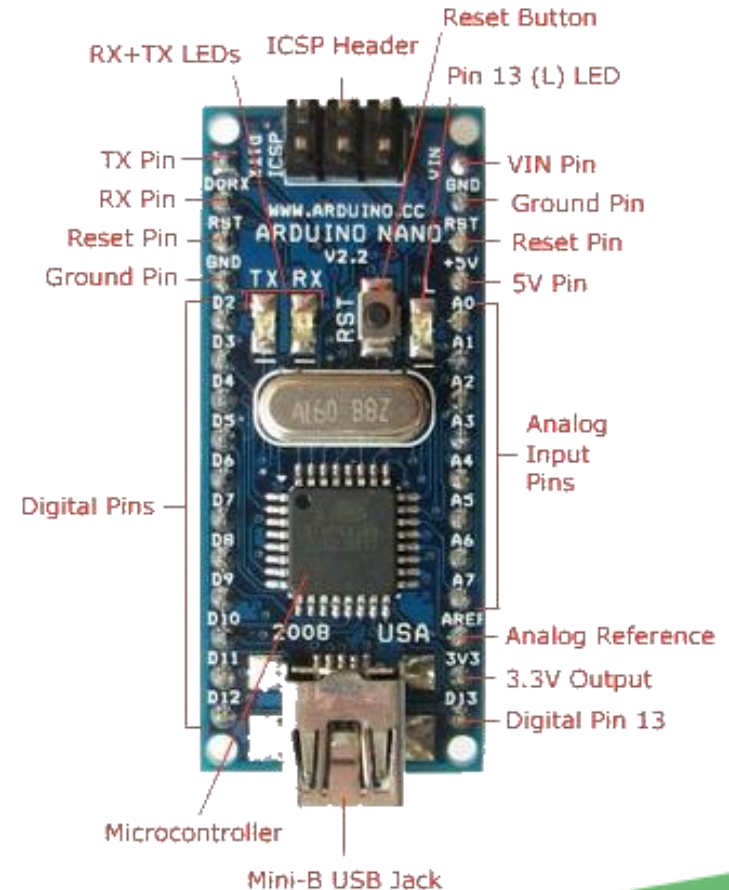


O que é o Arduino?



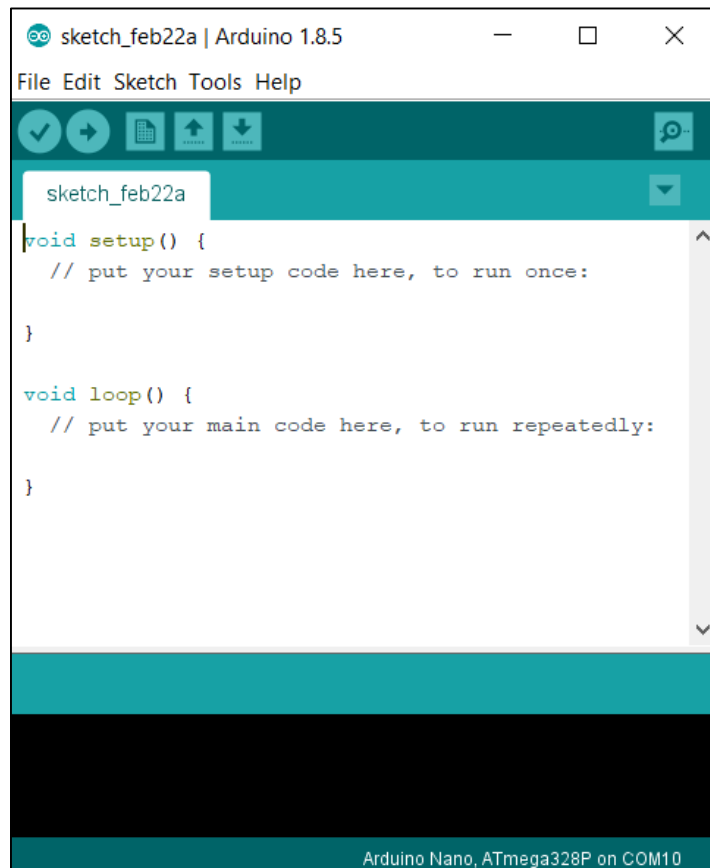
Especificações

Microcontrolador	ATmega328
Flash Memory	32KB
SRAM	2KB
EEPROM	1KB
Frequência do CLK	16MHz
Entradas analógicas	8
Entradas e saídas digitais	22 (6 PWM)
Tensão de entrada	7-12 V
Corrente máxima por saída	40mA





Arduino IDE



- Editar o código
- Compilar o código
- Programar o Arduino
- Serial Monitor
- Serial Plotter



Estrutura do código

- Inclusão de bibliotecas
- Declaração de MACROS
- void **setup()**
- void **loop()**

```
#include <exemplo.h>
#define SAIDA1 5
#define ENTRADA1 2

void setup() {
    // put your setup code here, to run once:
}

void loop() {
    // put your main code here, to run repeatedly:
}
```



Controlo de Fluxo e Variáveis

- If... else
- while
- do... while
- switch... case
- for
- continue
- break

int	16 bits
float	32 bits
long	32 bits
char	8 bits
byte	8 bits
bool	8 bits

<https://www.arduino.cc/reference/en/>



Função pinMode()

```
pinMode (pin, mode) ;
```

Argumentos:

- pin – numero do pin que queremos configurar
- mode – Existem três configurações possíveis:
 - INPUT (definido por defeito)
 - OUTPUT
 - INPUT_PULLUP



Função digitalWrite()

`digitalRead(pin) ;`

Argumentos:

- pin – numero do pin onde queremos medir uma tensão “ler”

Retorno:

- HIGH – valor na entrada aproximadamente 5V
- LOW – valor na entrada aproximadamente 0V



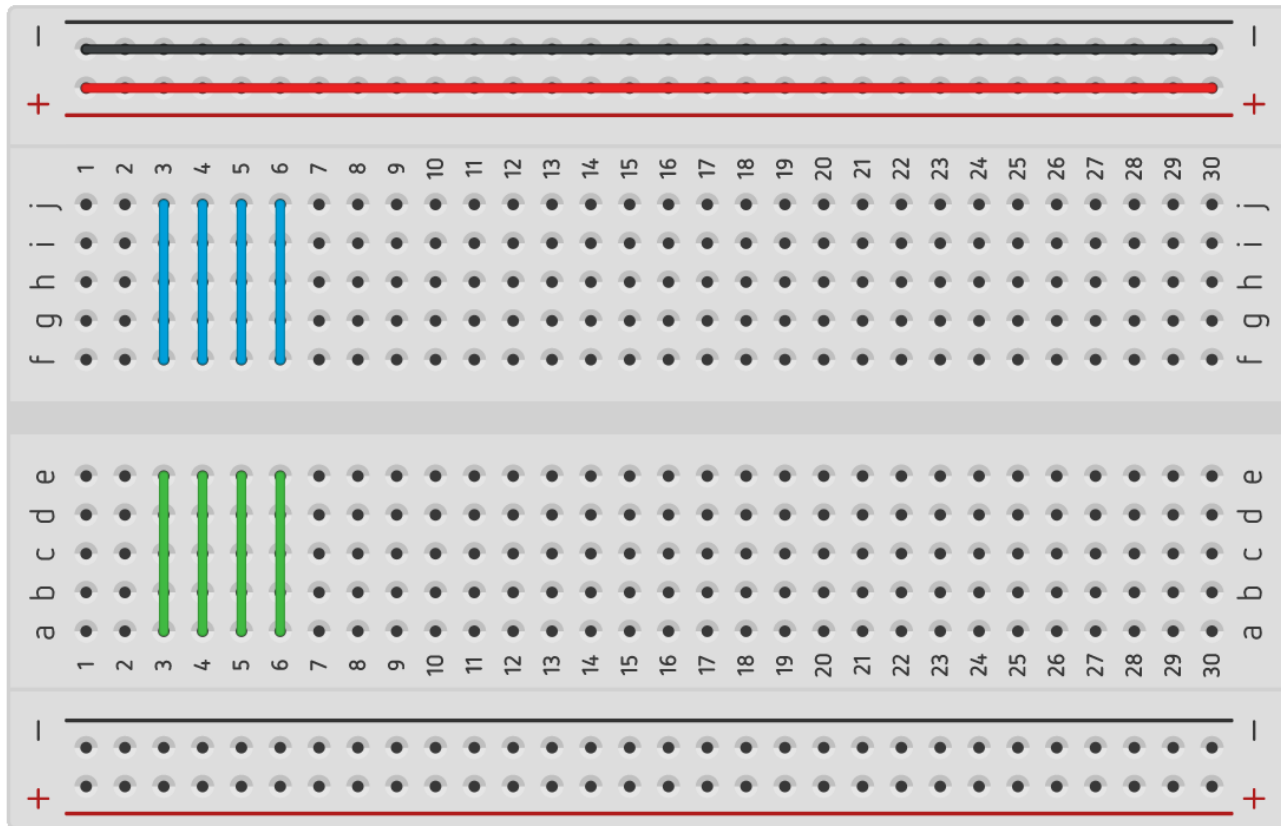
Função digitalWrite()

```
digitalWrite(pin, value);
```

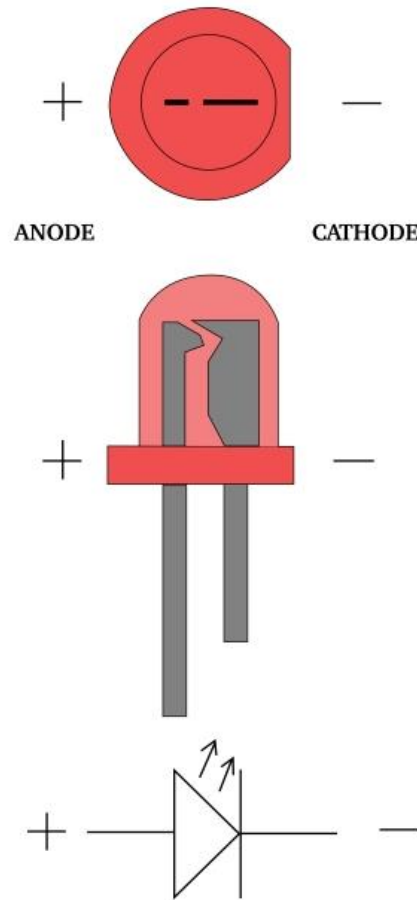
Argumentos:

- pin – numero do pin onde queremos colocar um valor de tensão “escrever”
- value:
 - HIGH – Coloca 5V na saída
 - LOW – Coloca 0V na saída

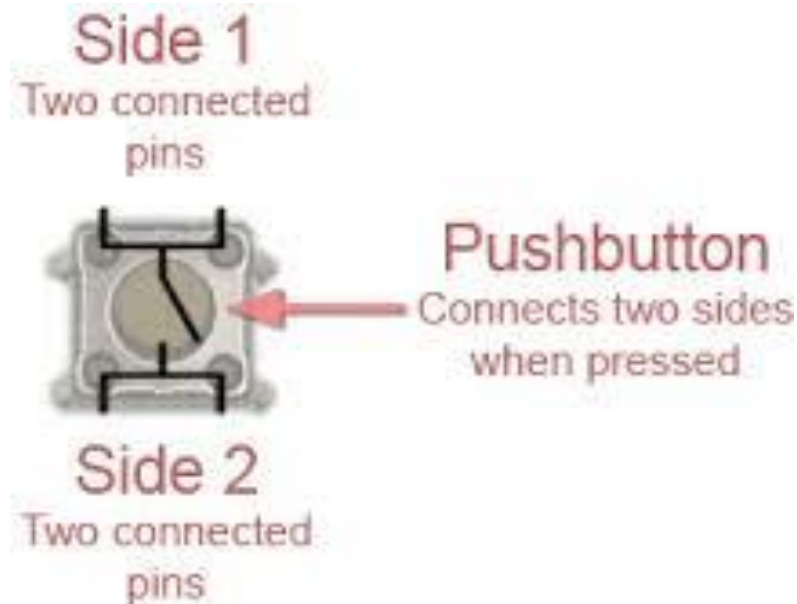
Como funciona a Breadboard?



Como funciona o LED?



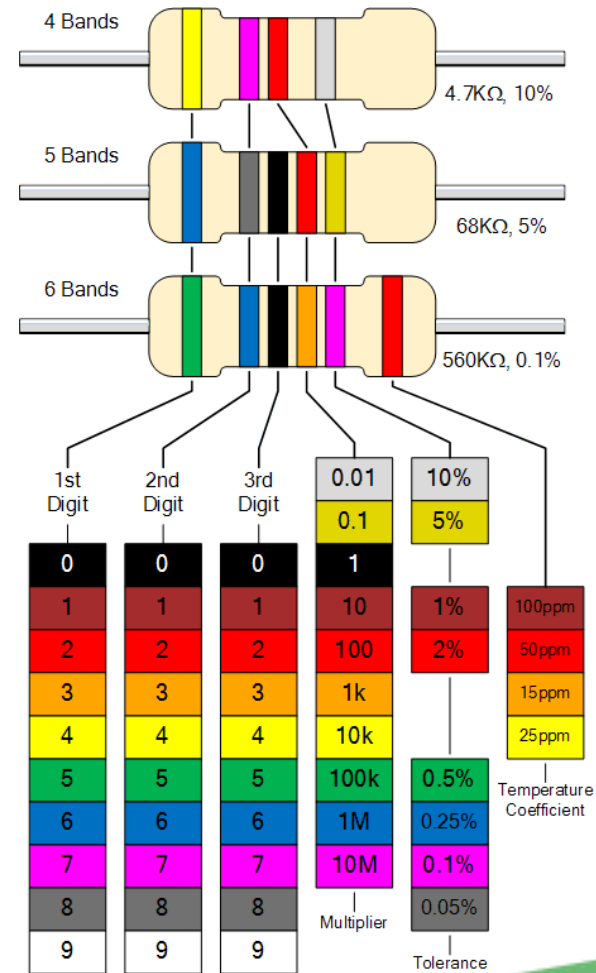
Como funciona o botão?



Resistência

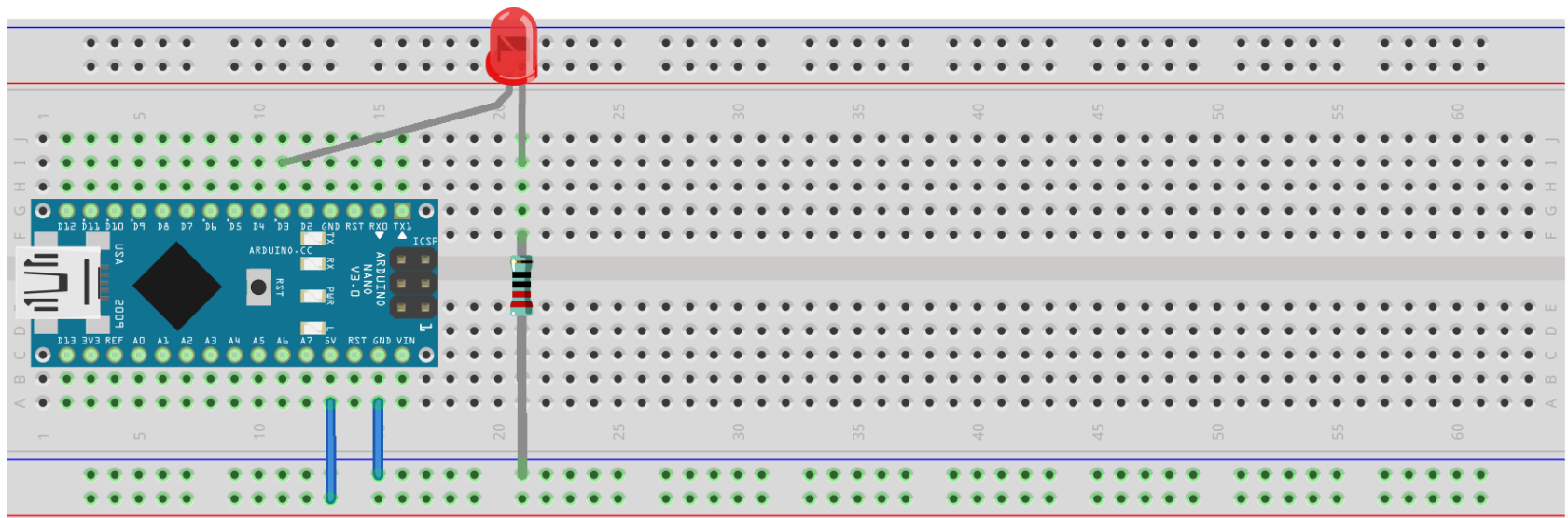
$$V = I * R$$

<https://www.eeweb.com/tools/4-band-resistor-calculator>



Exercício 1

- Configurar uma saída
- Acender o LED



fritzing



Solução exercício 1

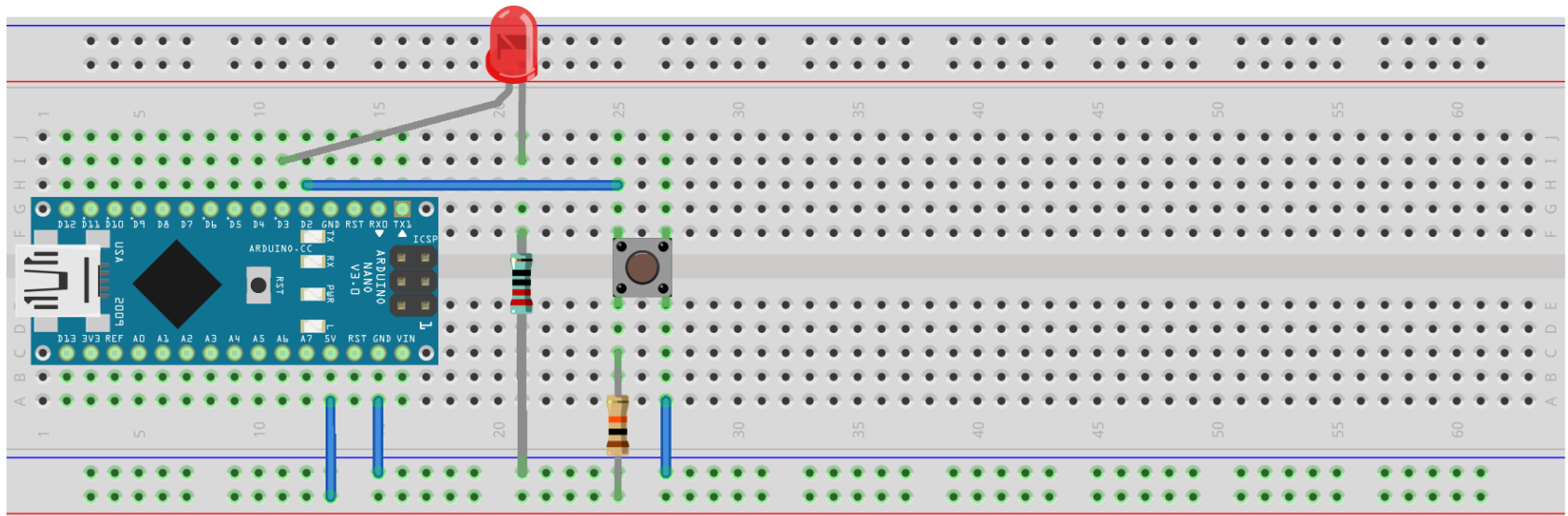
```
#define LED 3

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

void loop() {
    digitalWrite(LED, HIGH);
}
```

Exercício 2

- Configurar uma saída e uma entrada
- Acender o LED quando o botão está pressionado



fritzing



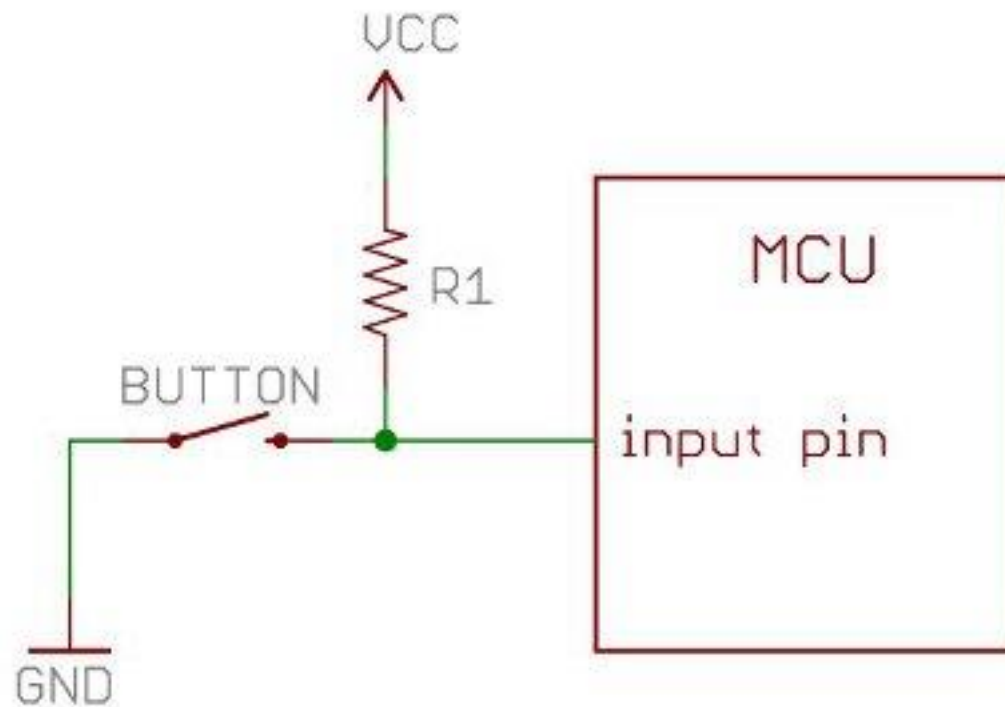
Solução exercício 2

```
#define BOTAO 2
#define LED 3

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

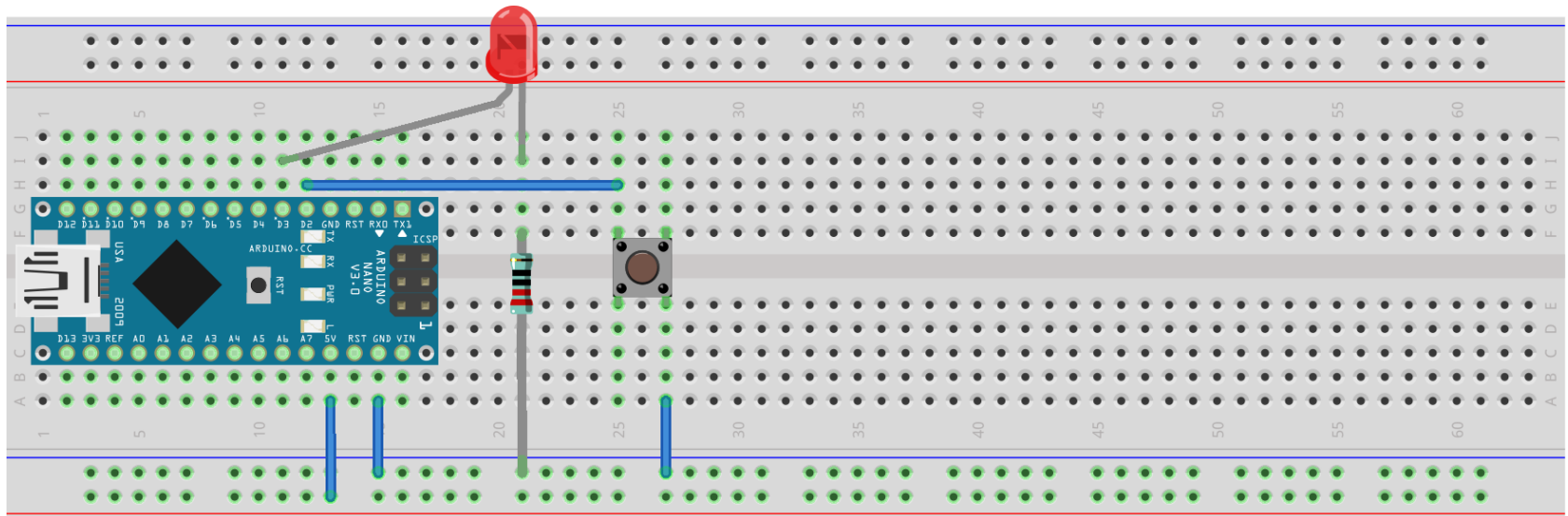
void loop() {
    if(!digitalRead(BOTAO)) {
        digitalWrite(LED, HIGH);
    } else {
        digitalWrite(LED, LOW);
    }
}
```

Circuito pull-up



Exercício 3

- Configurar uma saída e uma entrada **pull-up**
- Acender o LED quando o botão está pressionado



fritzing



Solução exercício 3

```
#define BOTAO 2
#define LED 3

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

void loop() {
    if(!digitalRead(BOTAO)) {
        digitalWrite(LED, HIGH);
    } else {
        digitalWrite(LED, LOW);
    }
}
```



Função analogRead()

`analogRead(pin) ;`

Argumentos:

- pin – numero do pin onde queremos ler o valor analógico (0-7)

Retorna:

- Inteiro entre 0 e 1023



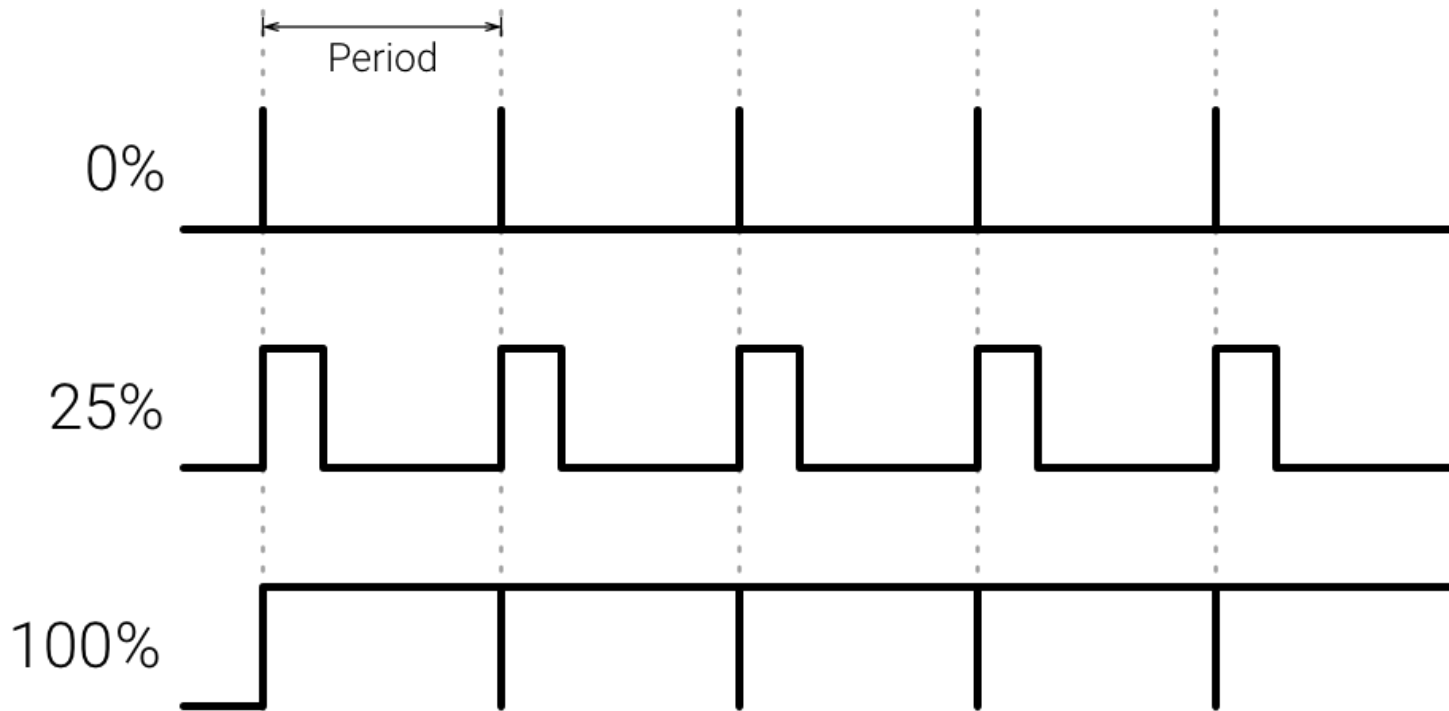
Função analogWrite()

```
analogWrite(pin, value);
```

Argumentos:

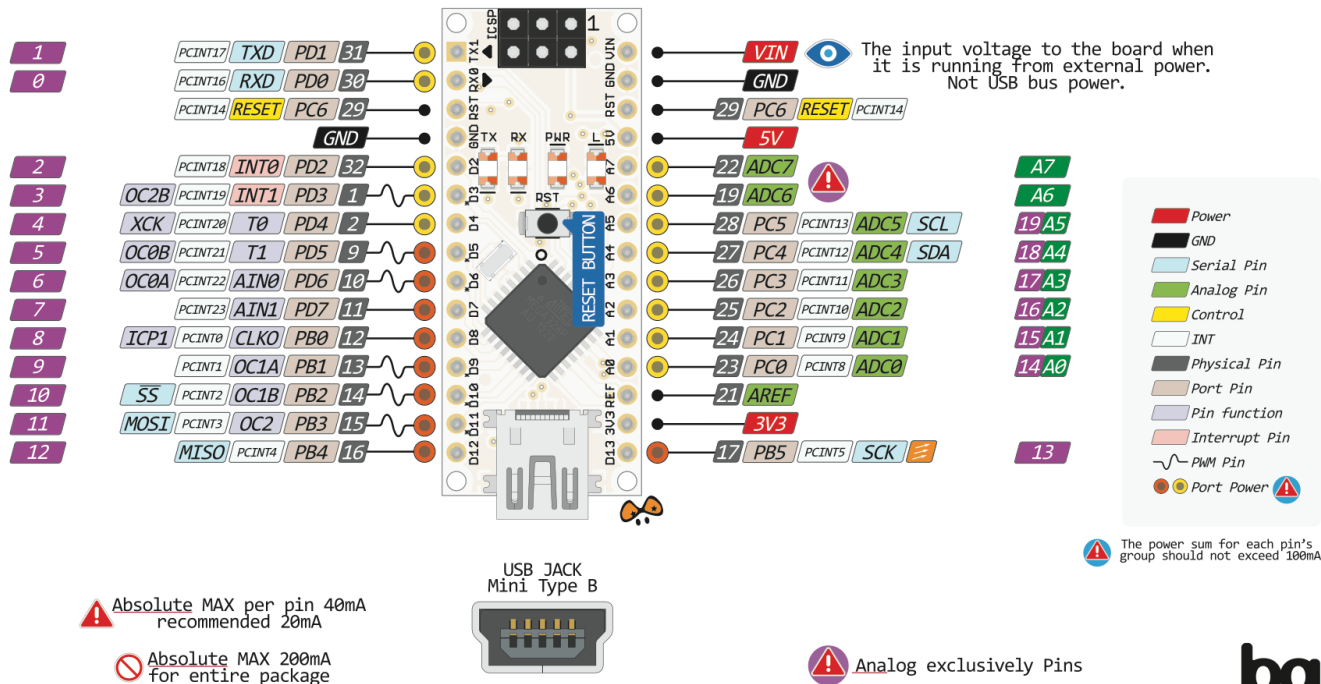
- pin – numero do pin onde queremos escrever um valor
- value – numero inteiro entre 0(sempré desligado) e 255(sempré ligado) que corresponde ao **duty cycle** do sinal de saída.

PWM



Saídas PWM

NANO PINOUT



Função map()

`map(value, fromLow, fromHigh, toLow, toHigh);`

Argumentos:

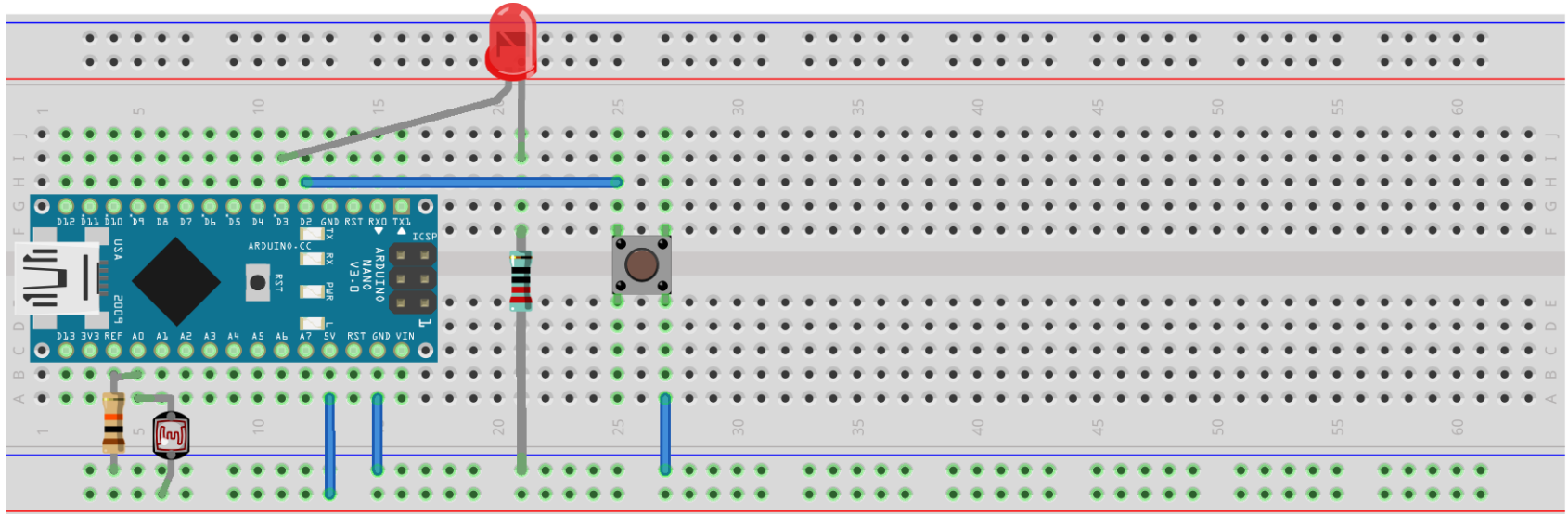
- Value: o numero(variável) a mapear
- fromLow: limite mínimo do valor presente em value
- fromHigh: limite máximo dos valor presente em value
- toLow: limite mínimo do valor retornado
- toHigh: limite máximo do valor retornado

Retorna:

- Valor mapeado

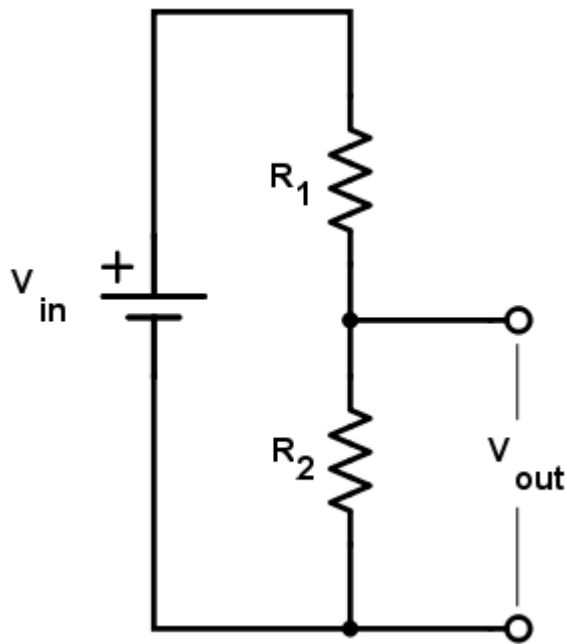
Exercício 4

- Acender o LED apenas quando o botão esta a ser pressionado
- Alterar a intensidade do led consoante a leitura do sensor



fritzing

Divisor de tensão



$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$



Solução exercício 4

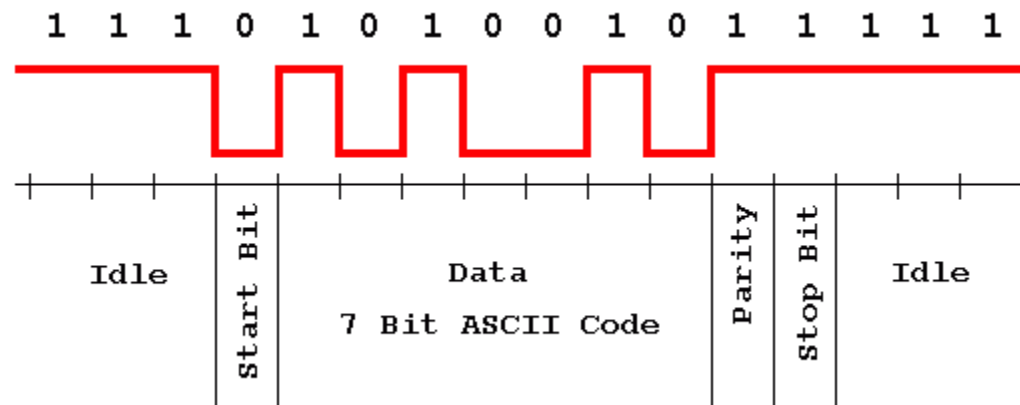
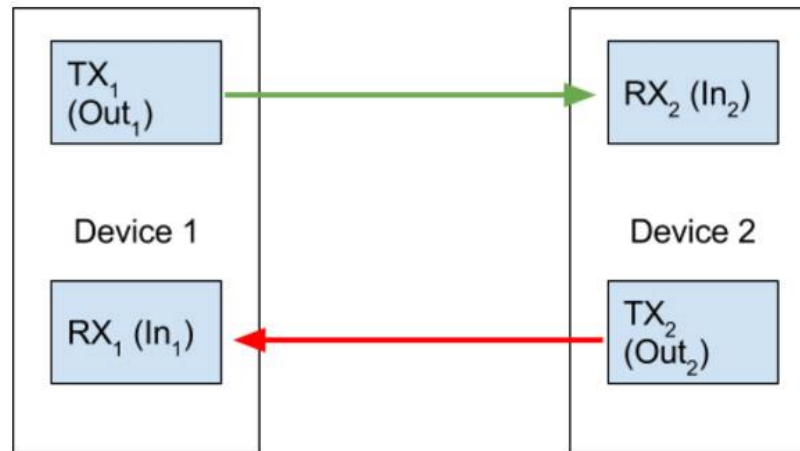
```
#define BOTAO 2
#define LED 3

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

void loop() {
    int sensor, led;
    sensor = analogRead(0);
    led = map(sensor, 0, 1023, 0, 255);

    if(!digitalRead(BOTAO)) {
        analogWrite(LED, led);
    } else {
        analogWrite(LED, LOW);
    }
}
```

Comunicação Serial





Função Serial.begin()

`Serial.begin(speed) ;`

Argumentos:

- Speed: Velocidade da *baud rate* em bits por segundo
- Velocidades: 300, 600, 1200, 2400, 4800, **9600**, 14400, 19200, 28800, 38400, 57600, or 115200



Função Serial.parseInt()

```
Serial.parseInt();
```

Procura pelo próximo valor convertível para *int* na porta serial

A conversão termina quando não foram lidos nenhuns caracteres durante um tempo especificado(`Serial.setTimeout()`) ou quando um valor que não é dígito é lido



Função Serial.available()

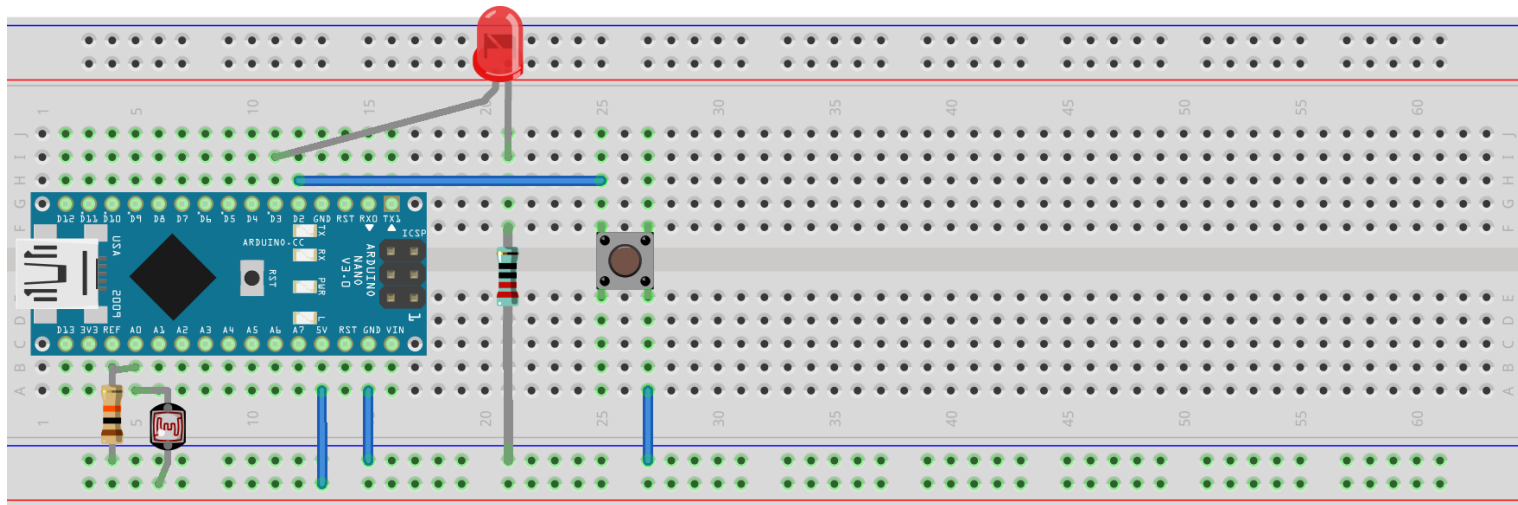
```
Serial.available() ;
```

Obter o número de bytes (caracteres) disponíveis para leitura da porta serial

```
if (Serial.available() != 0) {  
    analogWrite(LED, Serial.parseInt());  
}
```


Exercício 5

- Enviar por porta serial:
 - 0 quando o botão não está pressionado
 - valor entre 0 e 255 quando o botão está a ser pressionado, esse valor deve ser proporcional ao valor medido pelo LDR
- Alterar a intensidade do led consoante a leitura da porta Serial, são valido valores entre 0 e 255





Solução exercício 5

```
#define BOTAO 2
#define LED 3
#define LDR 0

void setup() {
    pinMode(BOTAO, INPUT_PULLUP);
    pinMode(LED, OUTPUT);
    Serial.begin(9600);
}

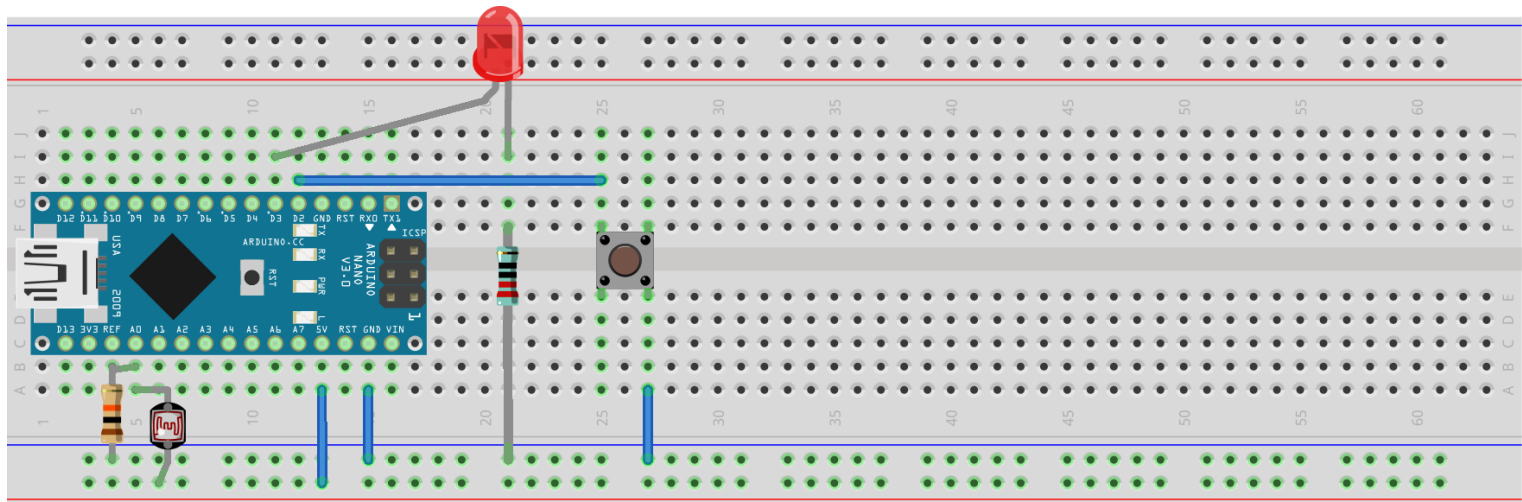
void loop() {
    int sensor, led;
    sensor = analogRead(LDR);
    led = map(sensor, 0, 1023, 0, 255);

    if(!digitalRead(BOTAO)) {
        Serial.println(led);
    } else {
        Serial.println(0);
    }

    if(Serial.available() != 0) {
        analogWrite(LED, Serial.parseInt());
    }
}
```

Exercício 6

- Tornar o botão persistente, um toque no botão deve ligar o envio do valor medido pelo LDR se o envio estava desligado e vice-versa



fritzing



Solução exercício 6

```
#define BOTAO 2
#define LED 3
#define LDR 0

bool state = false, curr = false, prev = false;

void setup() {
  pinMode(BOTAO, INPUT_PULLUP);
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int sensor, led;
  sensor = analogRead(LDR);
  led = map(sensor, 0, 1023, 0, 255);

  curr = digitalRead(BOTAO);

  if(curr == LOW && prev == HIGH){
    state = !state;
  }

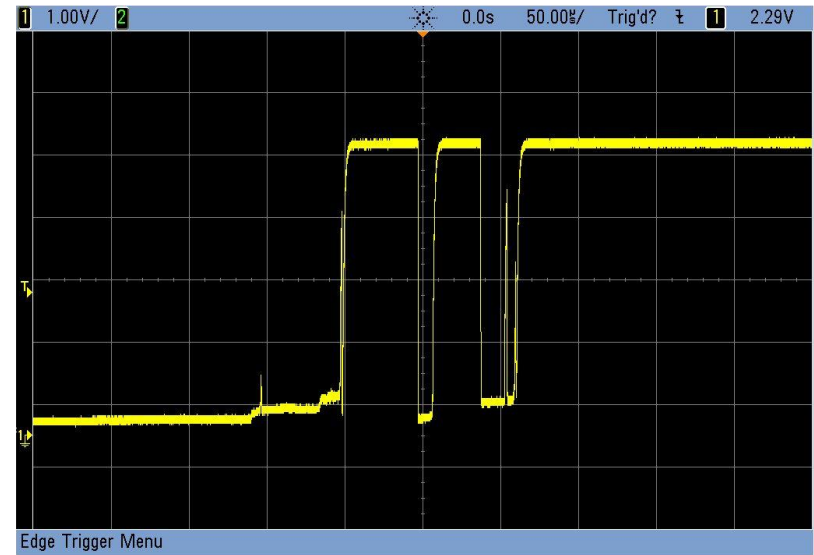
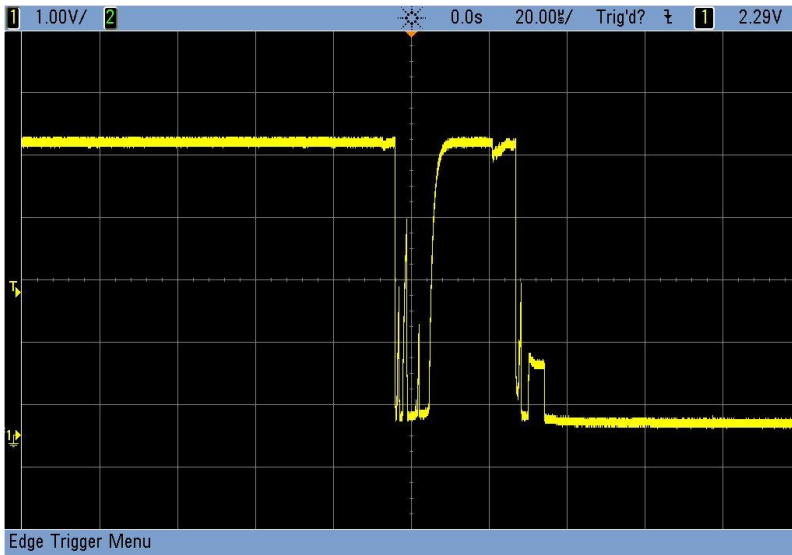
  if(!state){
    Serial.println(led);
  }else{
    Serial.println(0);
  }

  if(Serial.available() != 0){
    analogWrite(LED, Serial.parseInt());
  }

  prev = curr;
}
```



Debounce





Função delay()

```
delay(time) ;
```

Argumentos:

- time – tempo em milissegundos que o programa fica parado



Exercício 7

```
#define BOTAO 2
#define LED 3
#define LDR 0

bool state = false, curr = false, prev = false;

void setup() {
    pinMode(BOTAO, INPUT_PULLUP);
    pinMode(LED, OUTPUT);
    Serial.begin(9600);
}

void loop() {
    int sensor, led;
    sensor = analogRead(LDR);
    led = map(sensor, 0, 1023, 0, 255);

    curr = digitalRead(BOTAO);

    if(curr == LOW && prev == HIGH){
        state = !state;
    }
```

```
    if(!state){
        Serial.println(led);
    }else{
        Serial.println(0);
    }

    if(Serial.available() != 0){
        analogWrite(LED, Serial.parseInt());
    }

    prev = curr;

    delay(100);
}
```