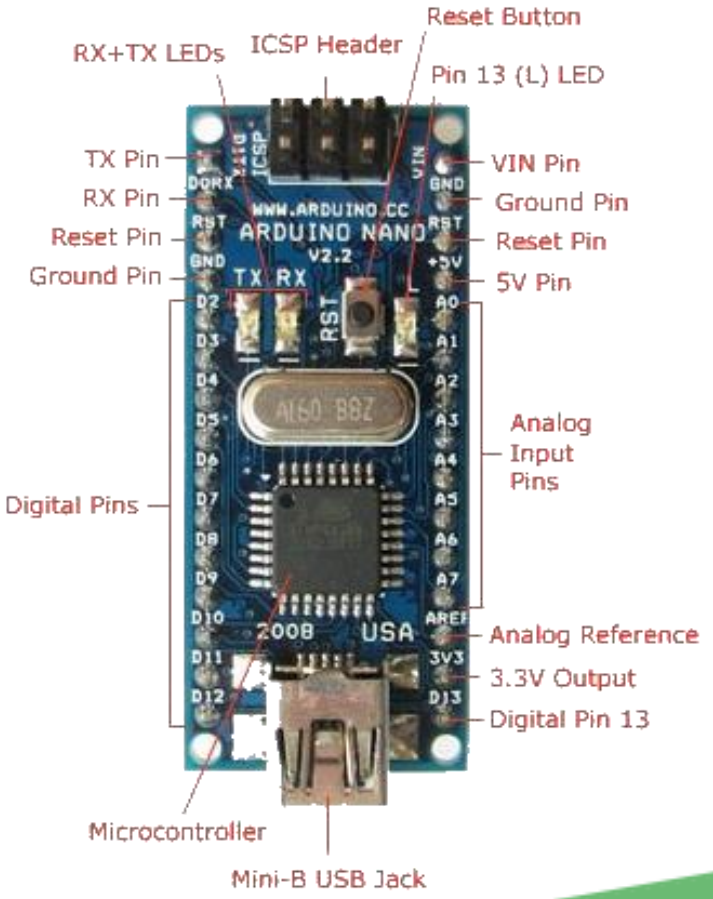
Workshop Arduino

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



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 digitalRead()

```
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

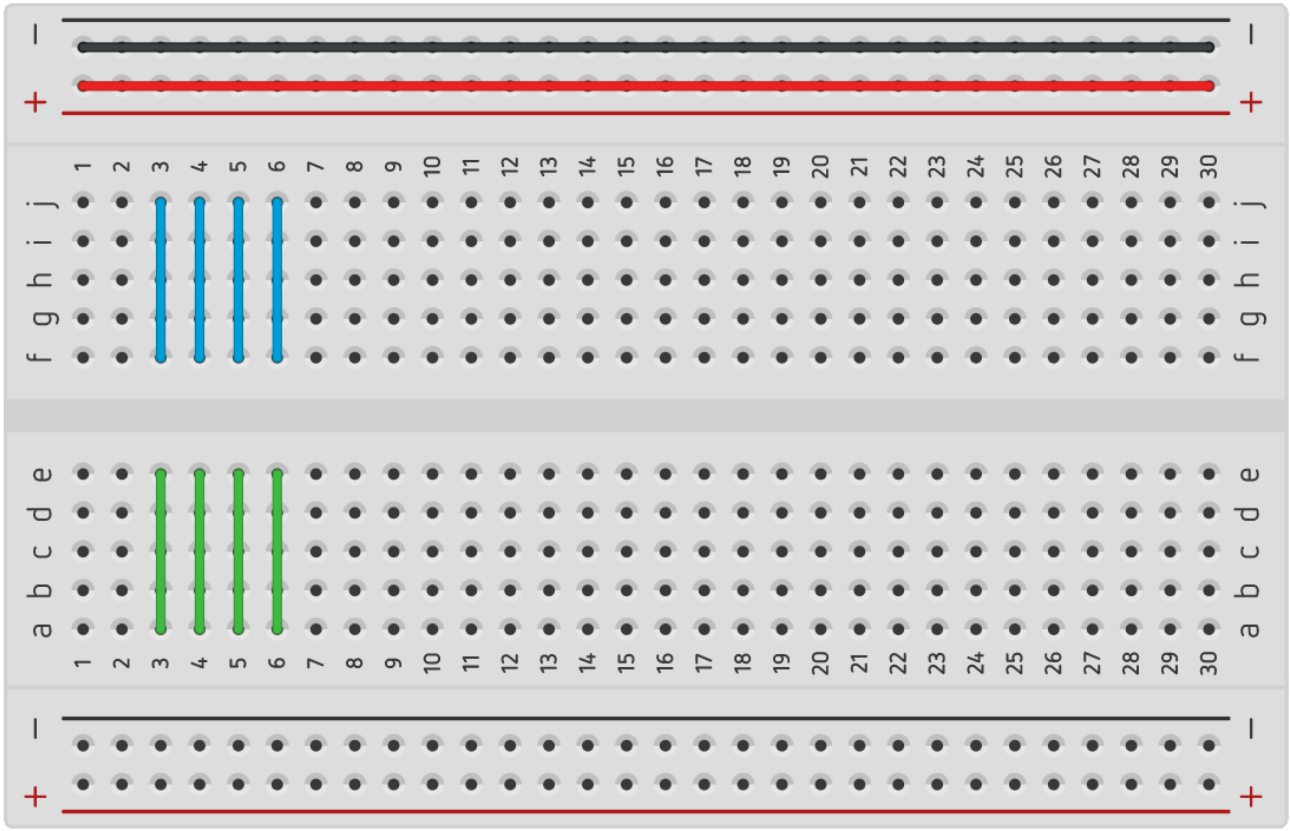
Função digitalWrite()

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

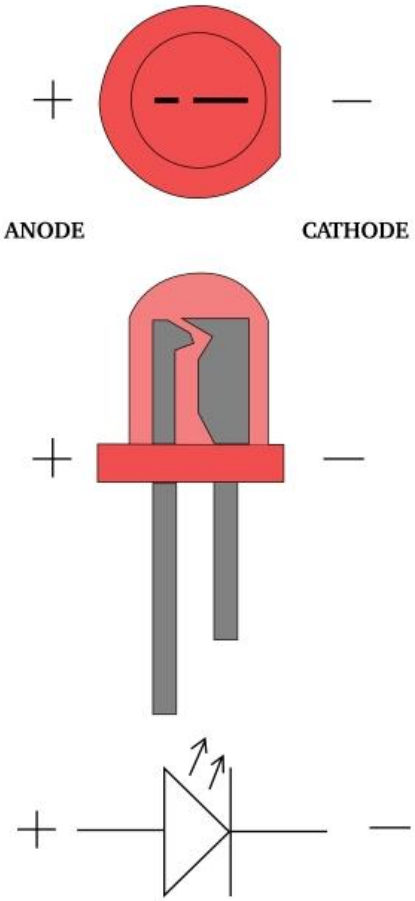
Argumentos:

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

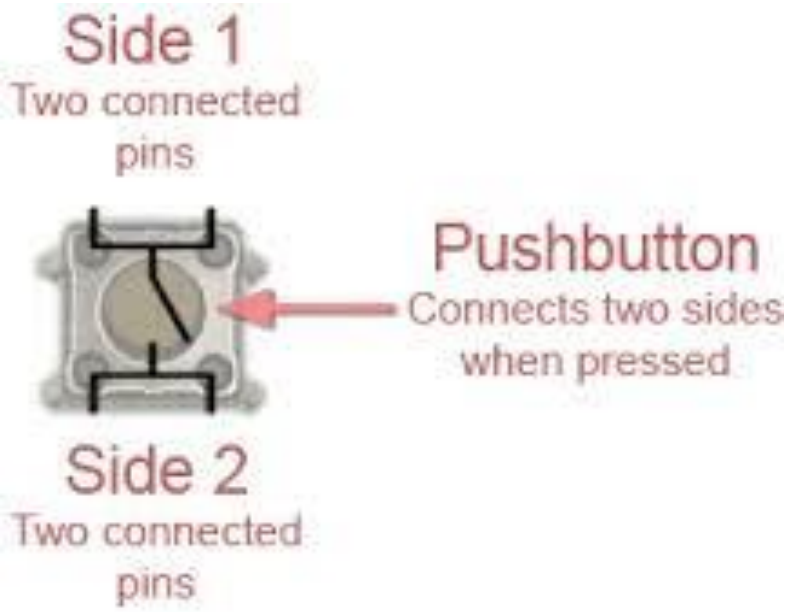
Como funciona a Breadboard?



Como funciona o LED?



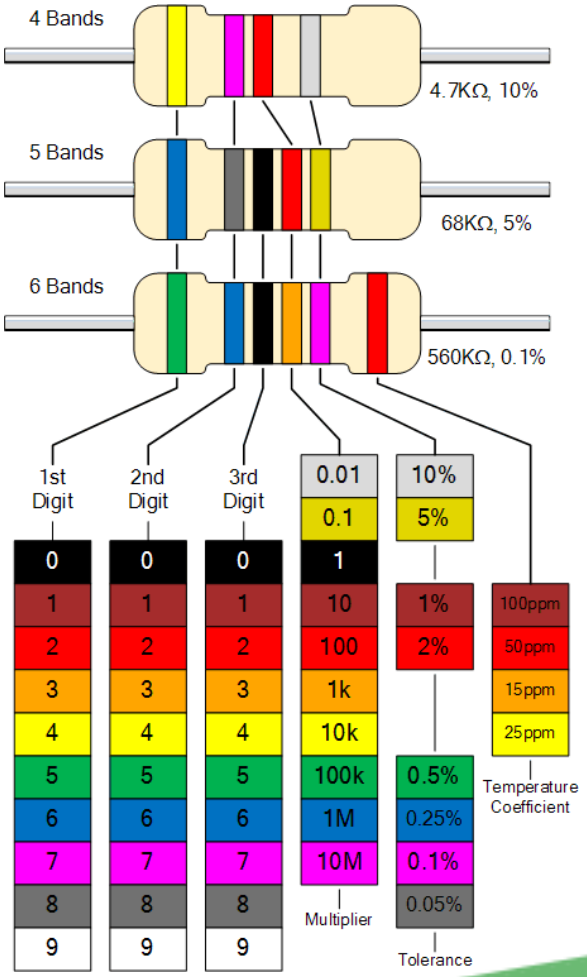
Como funciona o botão?



Resistência

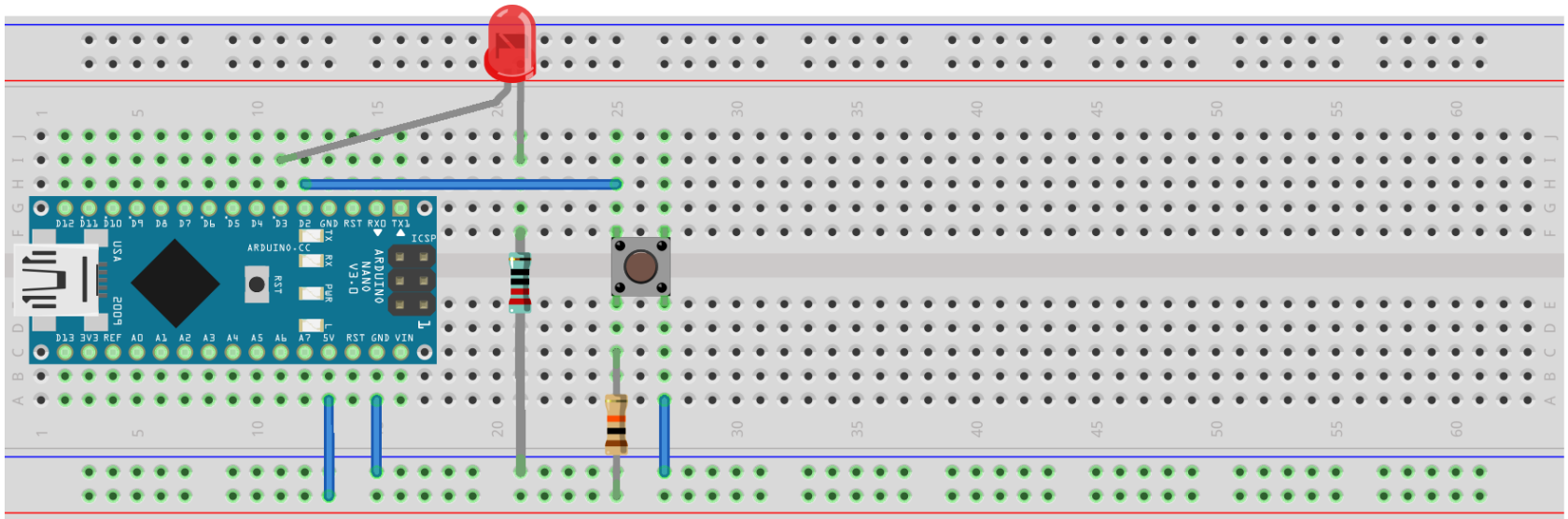
$$V = I * R$$

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

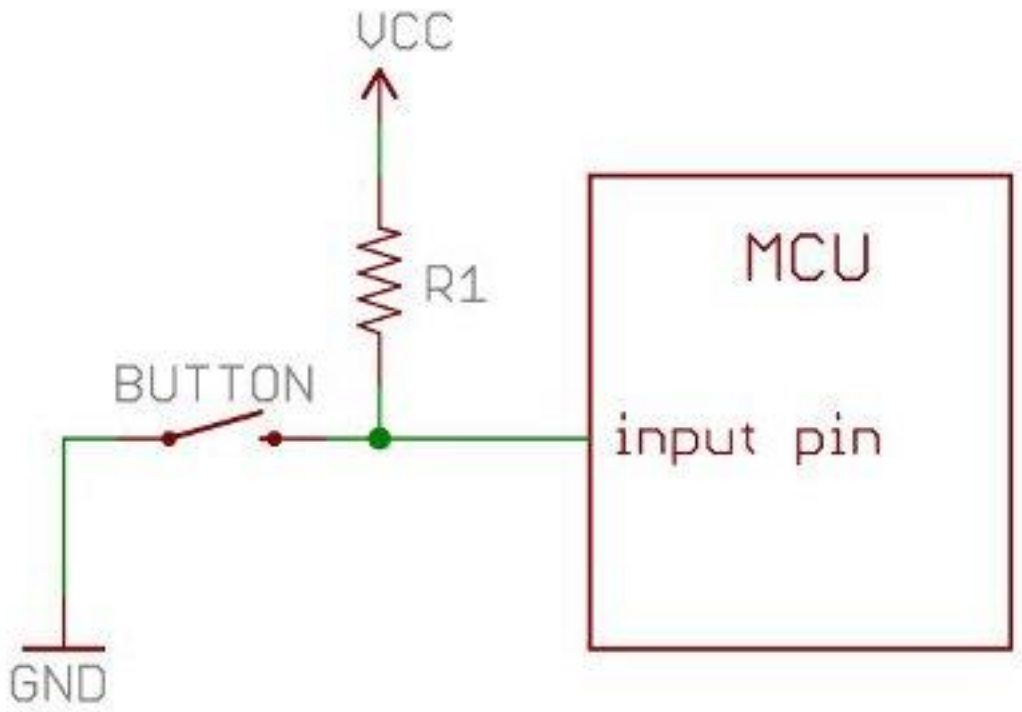


Exercício 1A

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



Circuito pull-up

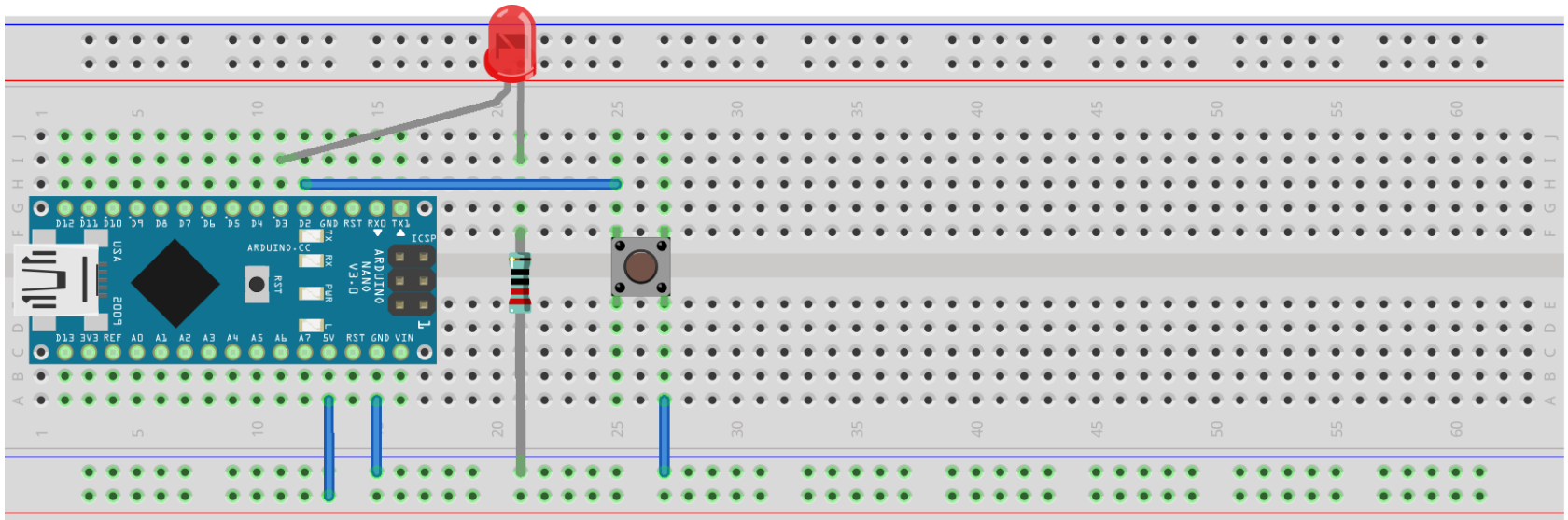


Solução exercício 1A

```
#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);
  }

}
```

Exercício 1B

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



Solução exercício 1B

```
#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

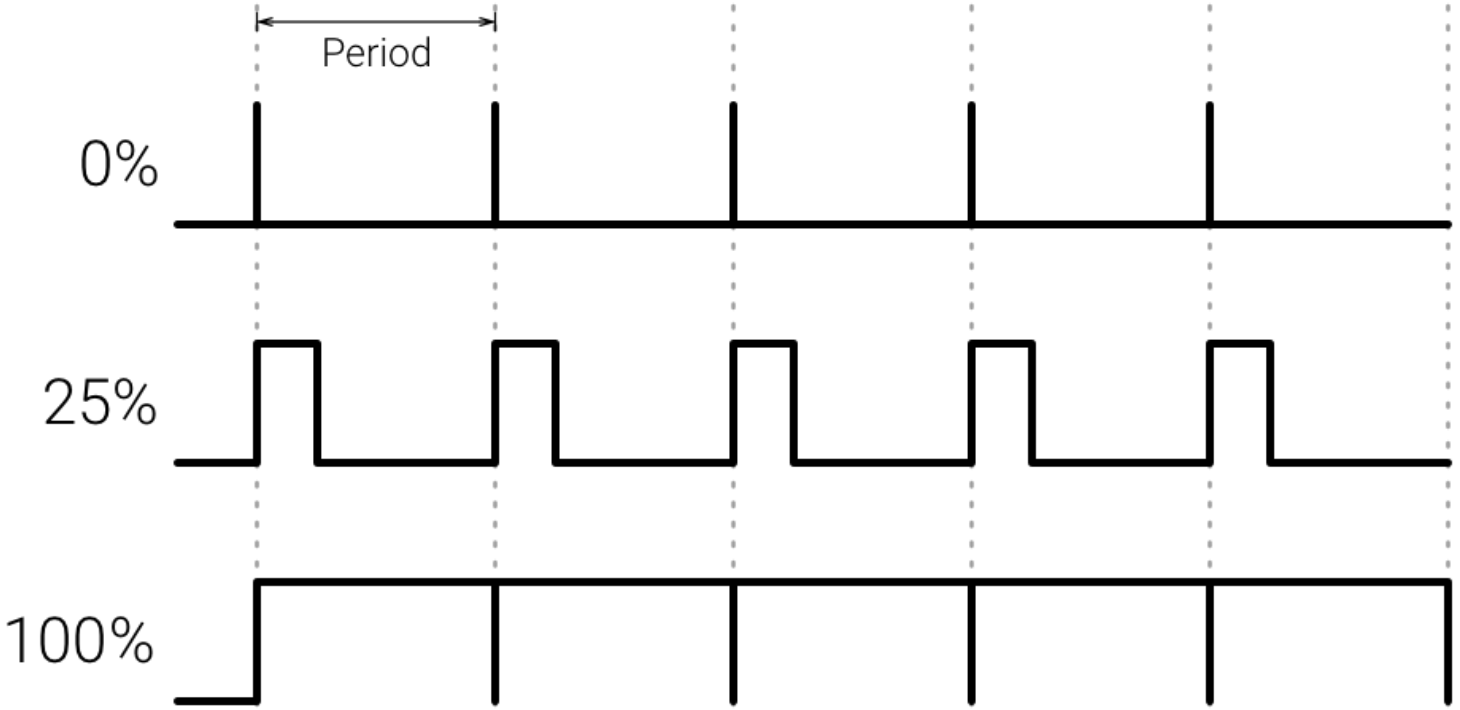
Função analogWrite()

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

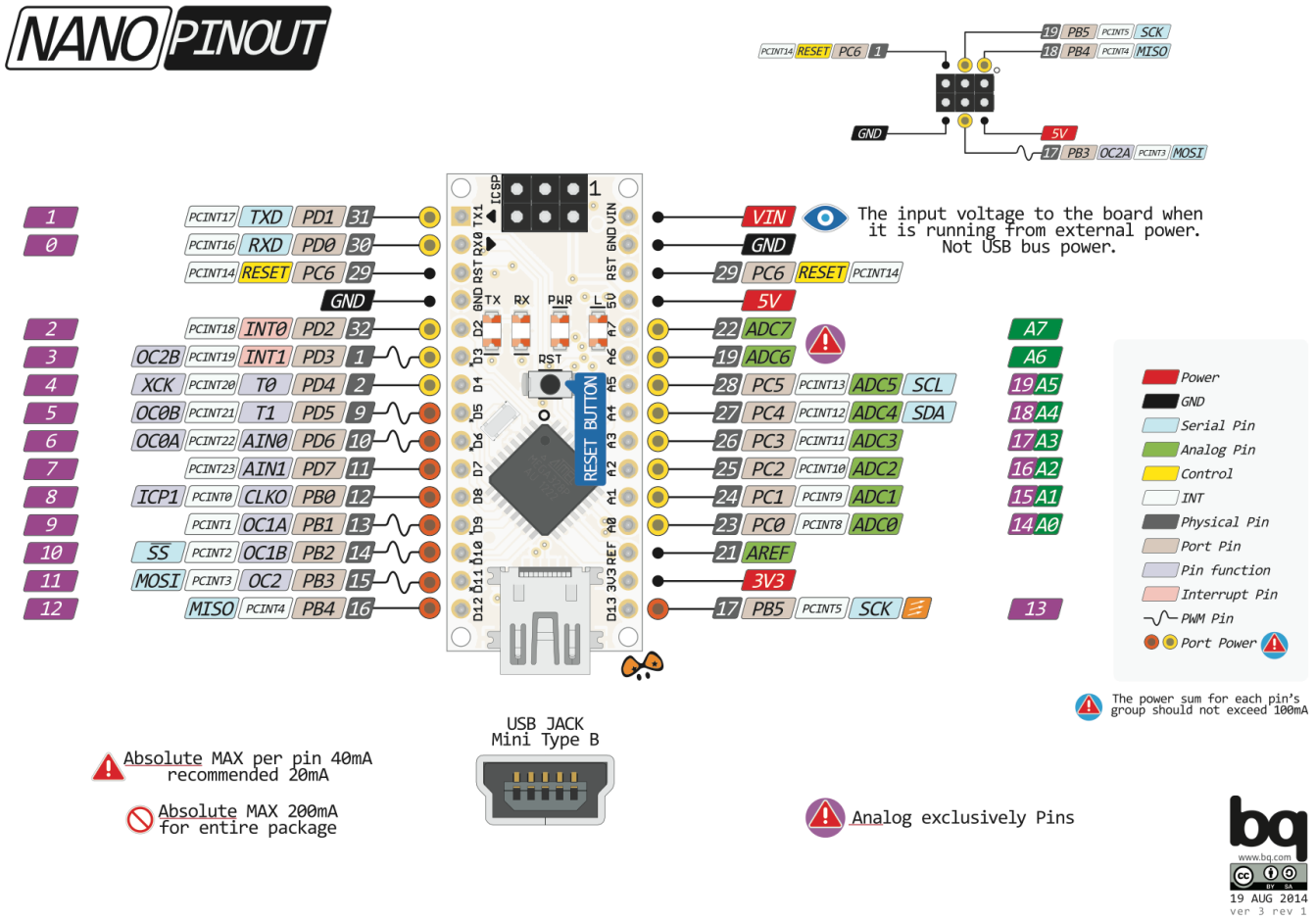
Argumentos:

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

PWM



Saídas PWM


NANO PINOUT
The input voltage to the board when it is running from external power. Not USB bus power.
RESET BUTTON
USB JACK Mini Type B
Absolute MAX per pin 40mA recommended 20mA
Absolute MAX 200mA for entire package
Analog exclusively Pins
The power sum for each pin's group should not exceed 100mA
Power
GND
Serial Pin
Analog Pin
Control
INT
Physical Pin
Port Pin
Pin function
Interrupt Pin
PWM Pin
Port Power
www.bq.com
19 AUG 2014
ver 3 rev 1

Função map()

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

Argumentos:

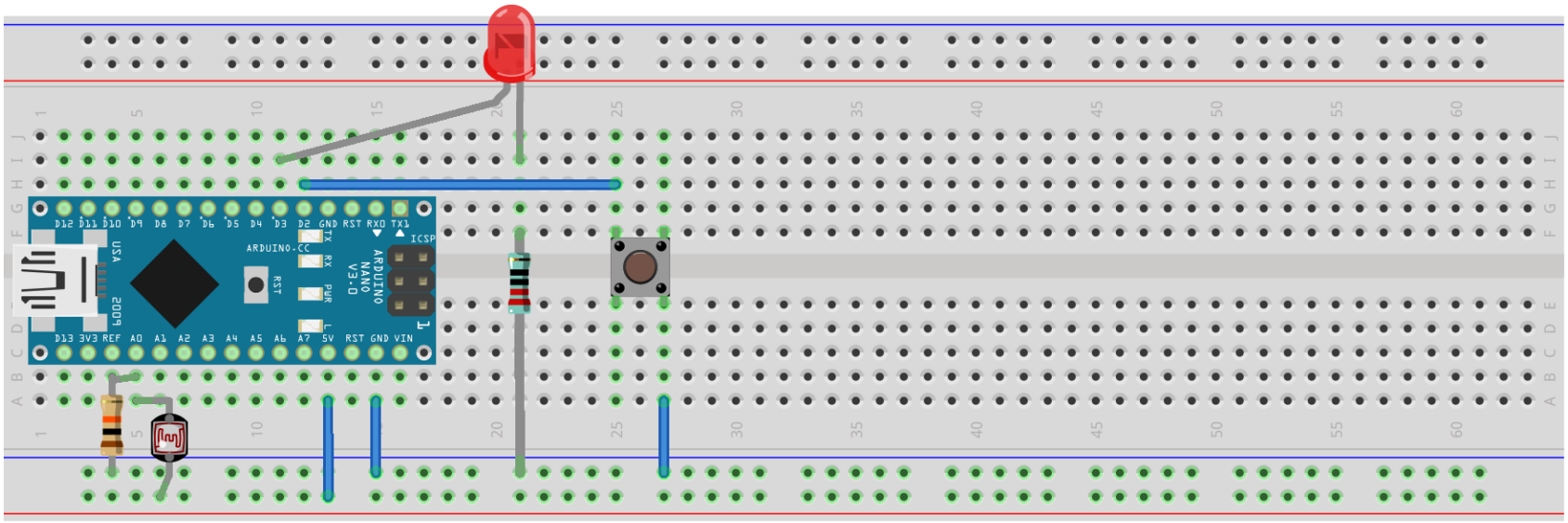
- Value: o numero(variável) a mapear
- fromLow: limite mínimo do valor presente em value
- fromHigh: limite máximo dos valor presente em value
- toLow: limite mínimo do valor retornado
- toHigh: limite máximo do valor retornado

Retorna:

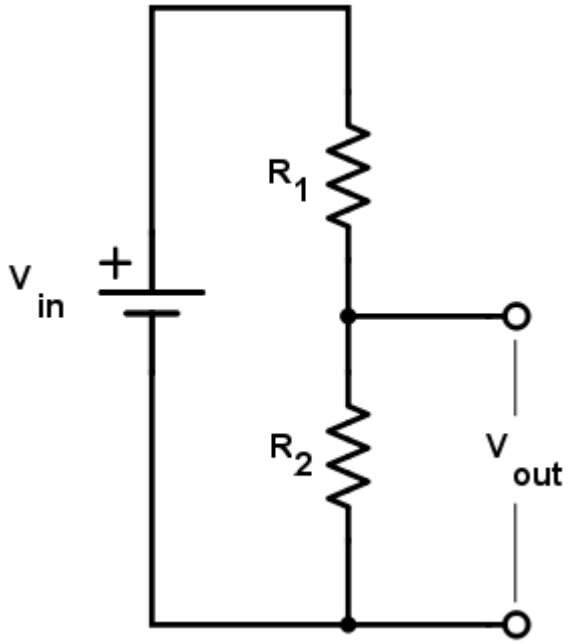
- Valor mapeado

Exercício 2

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



Divisor de tensão

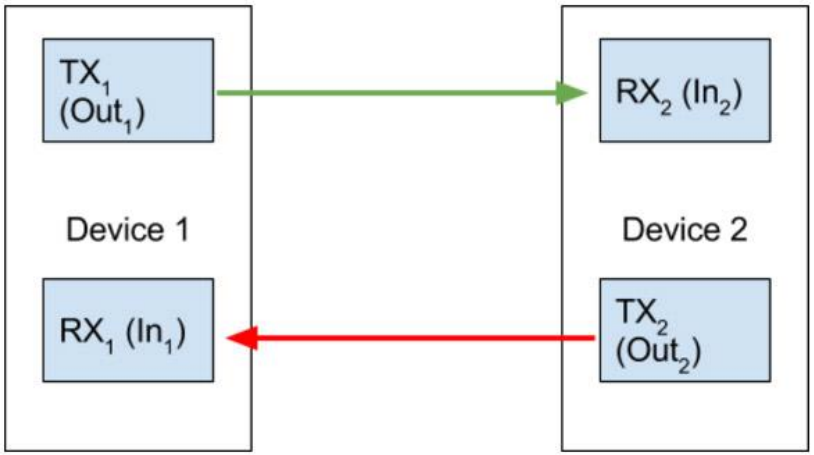


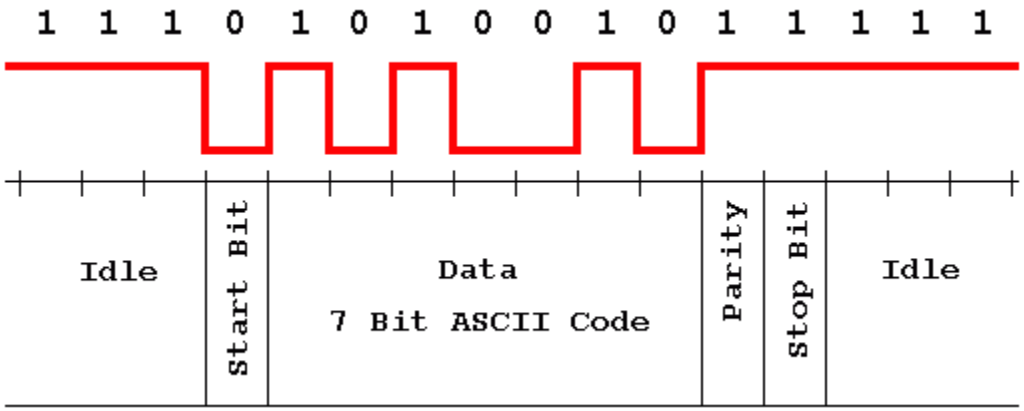
$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$

Solução exercício 2

```
#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()

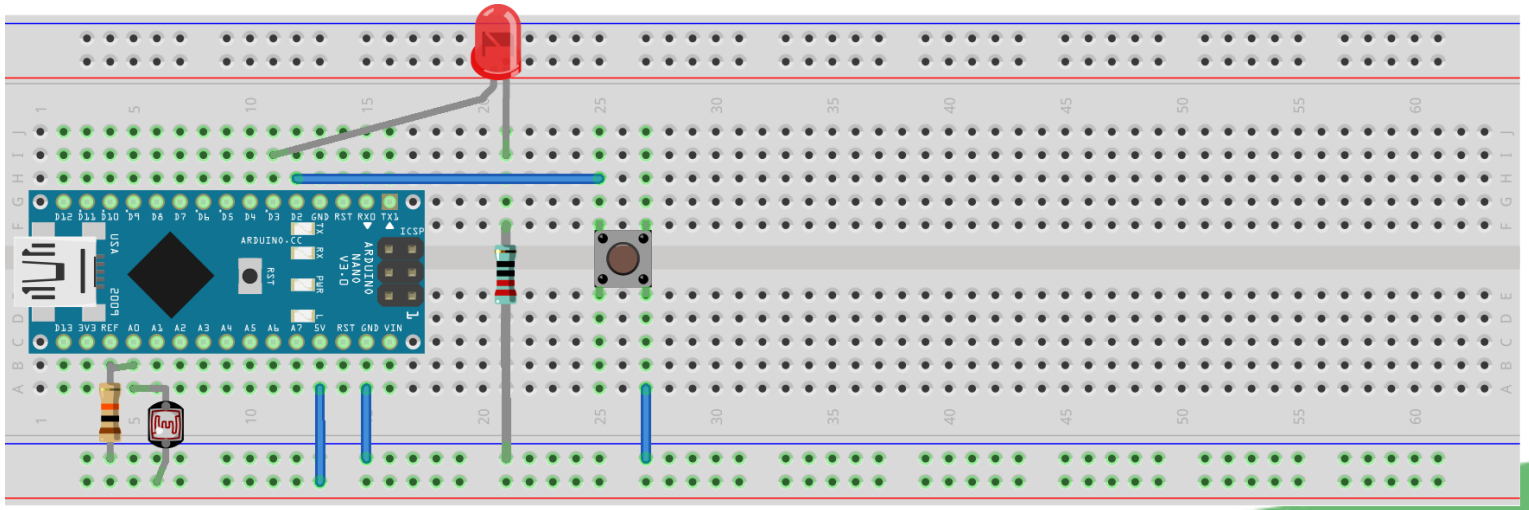
```
Serial.available();
```

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

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

Exercício 3

- 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 3

```
#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());
  }

}
```

Função tone()

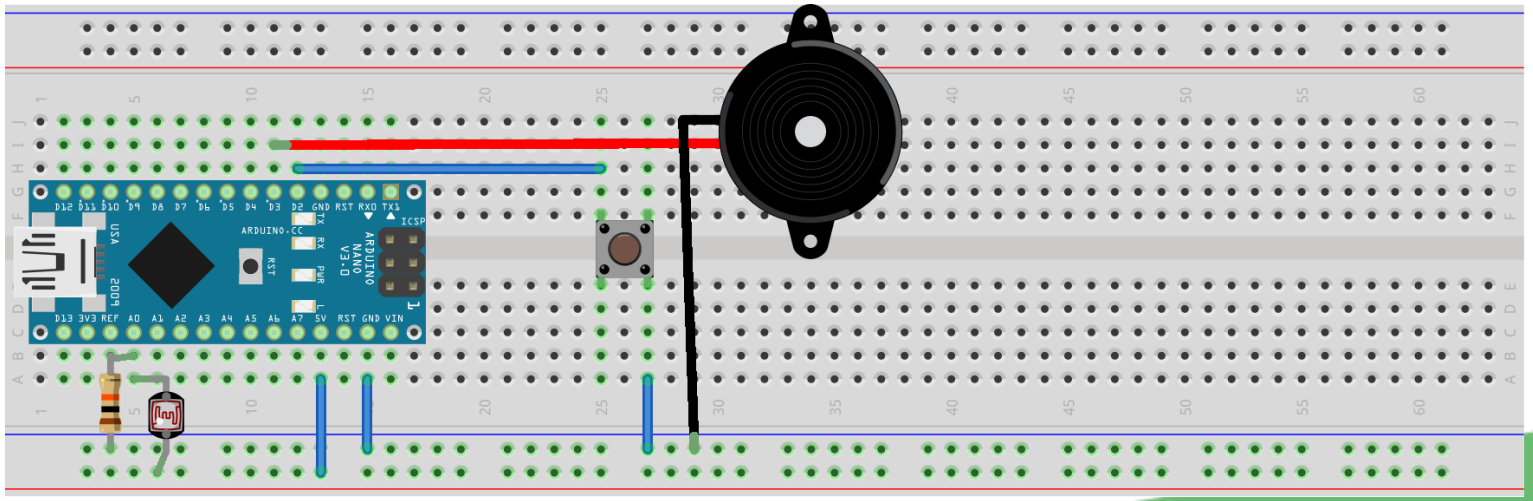
```
tone(pin, frequency);
tone(pin, frequency, duration);
```

Argumentos:

- Pin: Pin a gerar a onda quadrada com duty cycle a 50% e com as propriedades definidas nos outros argumentos
- Frequency: Frequência do sinal em hertz
- Duration: Duração do sinal em milisegundos

Exercício 4

- Enviar por porta serial:
  - 0 quando o botão não está pressionado
  - valor entre 500 e 5000 quando o botão está a ser pressionado, esse valor deve ser proporcional ao valor medido pelo LDR
- Alterar a frequência do som emitido pelo buzzer consoante a leitura da porta Serial, são validos valores entre 500 e 5000

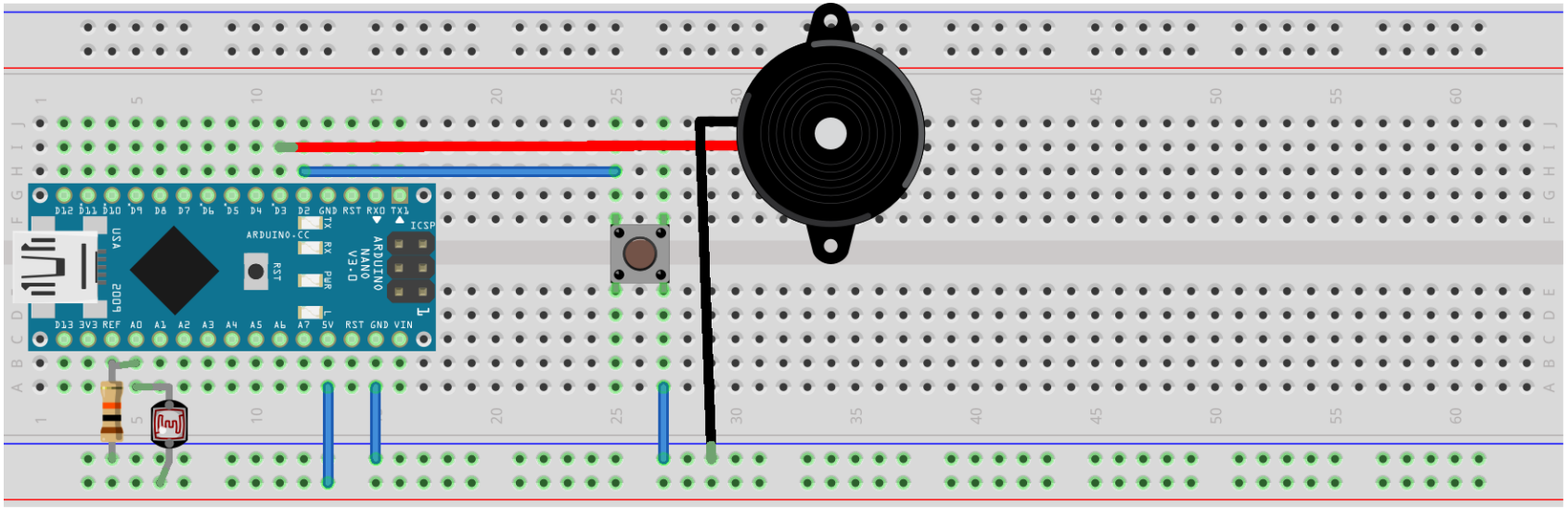


Solução exercício 4

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

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

void loop() {
  int sensor, buzzer, serial_in;
  sensor = analogRead(LDR);
  buzzer = map(sensor,0,1023,500,5000);

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

  if(Serial.available()!=0){
    serial_in = Serial.parseInt();
    if(serial_in==0){
      noTone(BUZZER);
    }else{
      tone(BUZZER,serial_in);
    }

  }

}
```

Exercício 5A

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



Solução exercício 5A

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

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

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

void loop() {

  int sensor, buzzer, serial_in;
  sensor = analogRead(LDR);
  buzzer = map(sensor,0,1023,500,5000);

  curr = digitalRead(BOTAO);

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

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

  if(Serial.available()!=0){
    serial_in = Serial.parseInt();
    if(serial_in==0){
      noTone(BUZZER);
    }else{
      tone(BUZZER,serial_in);
    }
  }

  prev = curr;
}
```

Exercício 5B

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

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

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

void loop() {

  int sensor, buzzer, serial_in;
  sensor = analogRead(LDR);
  buzzer = map(sensor,0,1023,500,5000);

  curr = digitalRead(BOTAO);

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

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

  if(Serial.available()!=0){
    serial_in = Serial.parseInt();
    if(serial_in==0){
      noTone(BUZZER);
    }else{
      tone(BUZZER,serial_in);
    }

  }

  prev = curr;

  delay(1000);

}
```

Função attachInterrupt()

```
attachInterrupt(interrupt,ISR,mode);
```

Argumentos:

- interrupt – numero da interrupção que queremos configurar (≠ numero do pin)
- ISR – Função a executar como interrupção
- mode – Existem quato configurações possíveis:
  - LOW
  - CHANGE
  - RISING
  - FALLING

Associação entre o numero da interrupção e o pino

Placa	INT.0	INT.1	INT.2	INT.3	INT.4	INT.5
Uno, Ethernet, **Nano**	2	3				
Mega2560	2	3	21	20	19	18
32u4 based (e.g. Leonardo, Micro)	3	2	0	1	7	

Função digitalPinToInterrupt()

```
digitalPinToInterrupt(pin);
```

Argumentos:

- pin – numero do pin cujo numero de interrupção queremos saber

Retorno:

- numero de interrupção associado ao pin colocado em argumento

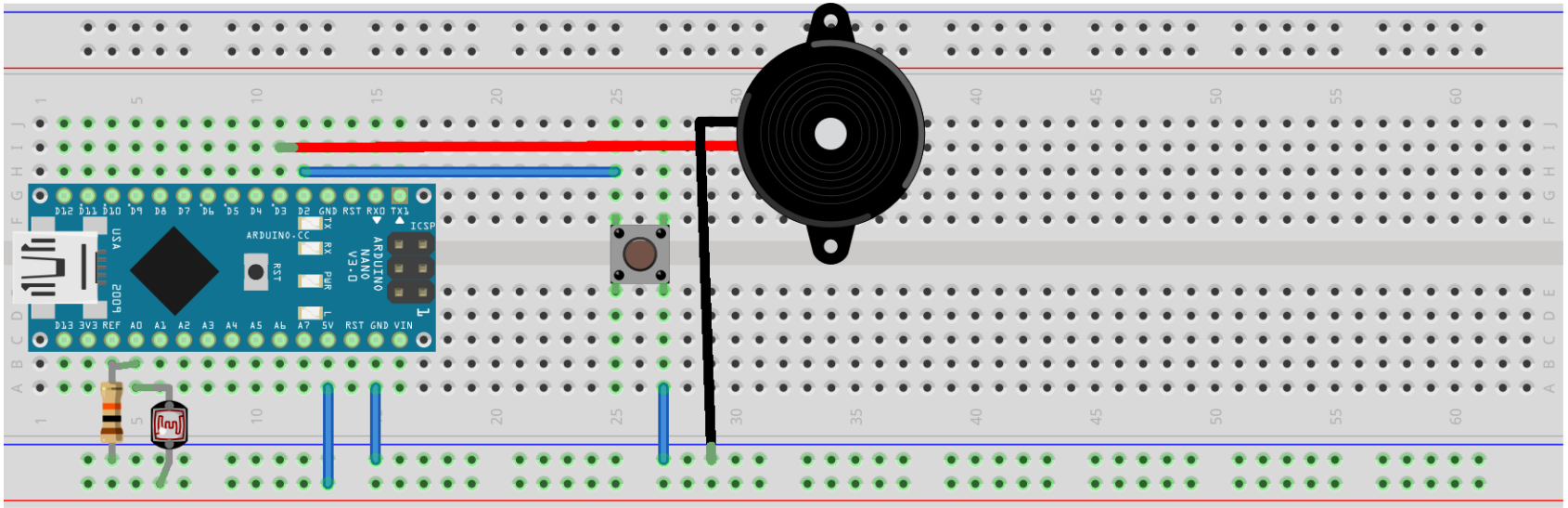
Interrupção

- Função delay() **não** funciona dentro da rotina de interrupção
- As rotinas de interrupção devem ser curtas
- Deve ser usado o qualificador volatile em variáveis usadas nas rotinas de interrupção

```
volatile bool var;

void setup() {
  attachInterrupt(digitalPinToInterrupt(2),exemplo,CHANGE);
}

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

void exemplo(){
  var = !var;
}
```

Exercício 6A

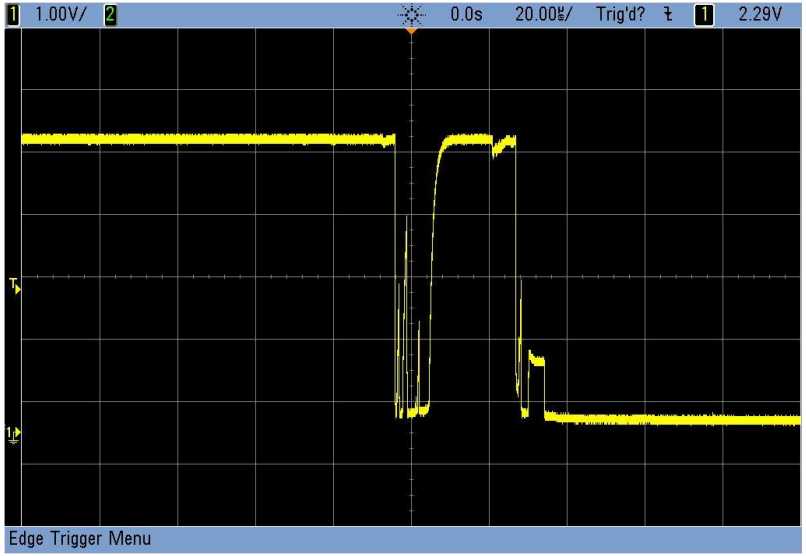
- Associar interrupção externa ao botão
- O botão deve ligar o envio do valor do buzzer se o envio estava desligado e vice-versa

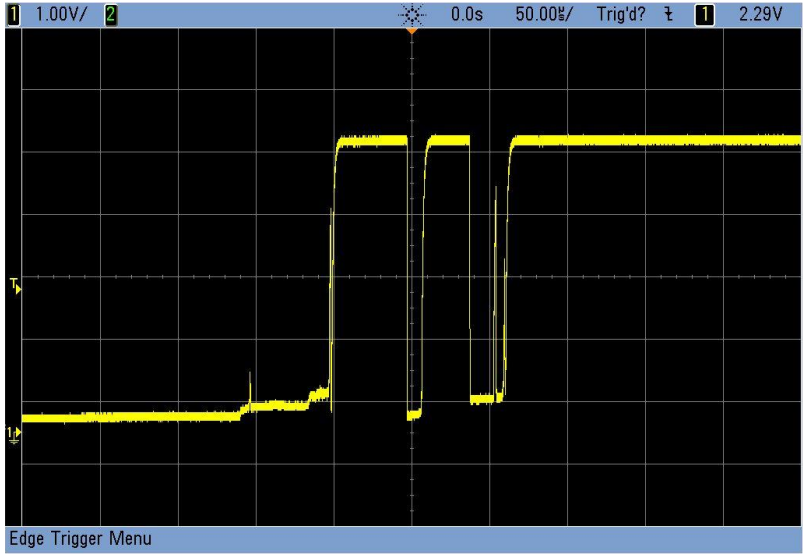


Solução exercício 6A

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

volatile bool play = false;

void setup() {
  pinMode(BOTAO,INPUT_PULLUP);
  pinMode(BUZZER,OUTPUT);
  Serial.begin(9600);
  attachInterrupt(digitalPinToInterrupt(BOTAO), interrupt, FALLING);
}

void loop() {
  int sensor, buzzer, serial_in;
  sensor = analogRead(LDR);
  buzzer = map(sensor,0,1023,500,5000);

  if(!play){
    Serial.println(buzzer);
  }else{
    Serial.println(0);
  }
  if(Serial.available()!=0){
    serial_in = Serial.parseInt();
    if(serial_in==0){
      noTone(BUZZER);
    }else{
      tone(BUZZER,serial_in);
    }
  }
}

void interrupt(){
  play = !play;
}
```

Debounce


1 1.00V/ 2
0.0s 20.00ns/ Trig'd?
1 2.29V
Edge Trigger Menu


1 1.00V/ 2
0.0s 50.00ns/ Trig'd?
1 2.29V
Edge Trigger Menu

Função millis()

```
millis();
```

Retorno:

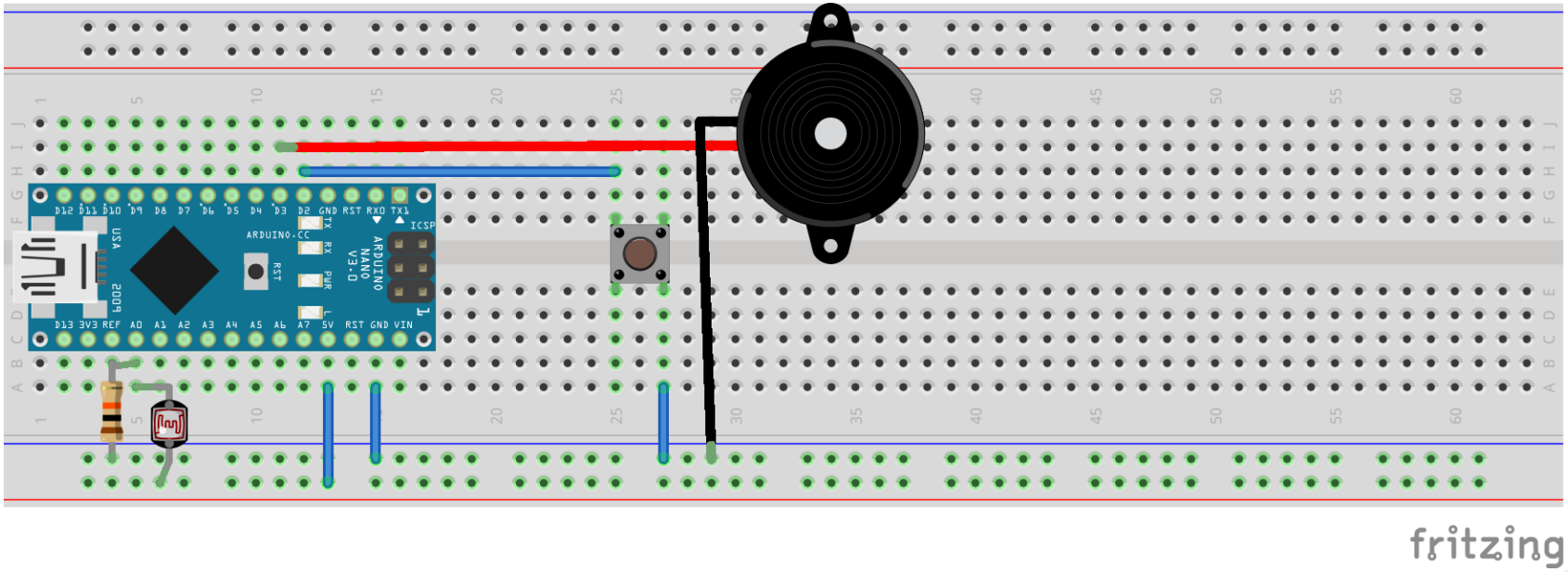
- Numero de milissegundos desde que o Arduino está ligado, volta a zero passado 50 dias!

Deve ser usado um **unsigned long** para guardar o valor retornado por esta função.

Exercício 6B

- Implementar um debouncer

Sugestão: Não deixar alterar a variável *play* se esta foi alterada há menos de 500 ms.



Solução exercício 6B

```
#define BOTAO 2
#define BUZZER 3
#define LDR 0
#define DELAY 500

volatile bool play = false;
volatile unsigned long timer = 0;

void setup() {
  pinMode(BOTAO,INPUT_PULLUP);
  pinMode(BUZZER,OUTPUT);
  Serial.begin(9600);
  attachInterrupt(digitalPinToInterrupt(BOTAO), interrupt, FALLING);
}

void loop() {
  int sensor, buzzer, serial_in;
  sensor = analogRead(LDR);
  buzzer = map(sensor,0,1023,500,5000);

  if(!play){
    Serial.println(buzzer);
  }else{
    Serial.println(0);
  }
  if(Serial.available()!=0){
    serial_in = Serial.parseInt();
    if(serial_in==0){
      noTone(BUZZER);
    }else{
      tone(BUZZER,serial_in);
    }
  }
}

void interrupt(){
  if((millis()-timer > DELAY)){
    play = !play;
    timer = millis();
  }
}
```

Exercício 6C

```
#define BOTAO 2
#define BUZZER 3
#define LDR 0
#define DELAY 500

volatile bool play = false;
volatile unsigned long timer = 0;

void setup() {
  pinMode(BOTAO,INPUT_PULLUP);
  pinMode(BUZZER,OUTPUT);
  Serial.begin(9600);
  attachInterrupt(digitalPinToInterrupt(BOTAO), interrupt, FALLING);
}

void loop() {
  int sensor, buzzer, serial_in;
  sensor = analogRead(LDR);
  buzzer = map(sensor,0,1023,500,5000);

  if(!play){
    Serial.println(buzzer);
  }else{
    Serial.println(0);
  }
  if(Serial.available()!=0){
    serial_in = Serial.parseInt();
    if(serial_in==0){
      noTone(BUZZER);
    }else{
      tone(BUZZER,serial_in);
    }
  }

  delay(1000);
}

void interrupt(){
  if((millis()-timer > DELAY)){
    play = !play;
    timer = millis();
  }
}
```