Arduino

Aula 03

Entrada digital
Comunicação Arduino > Processing

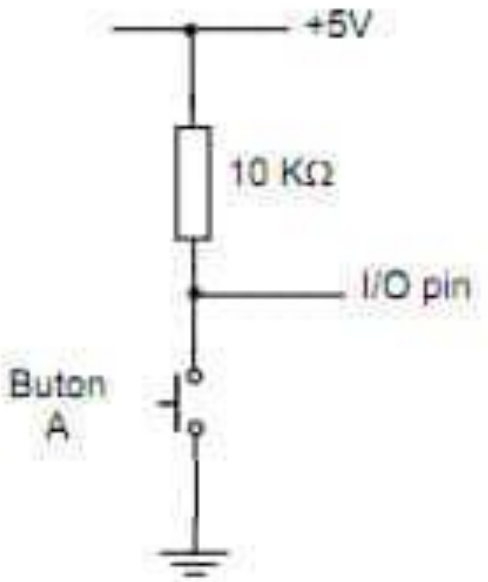
Entrada Digital

Sintaxe:

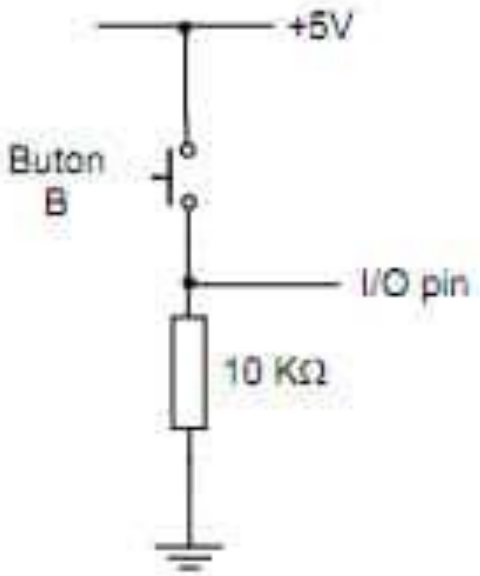
- `pinMode(porta,INPUT);`

- Resistores *pull-up*
  - Para evitar um estado indefinido, as entradas digitais possuem um resistor chamado *pull-up*
  - Para ativar o resistor *pull-up*, escreva HIGH na porta de entrada:
    - **`digitalWrite(porta,HIGH)`**

Entrada digital
Resistores ”pull-up” e ”pull-down”



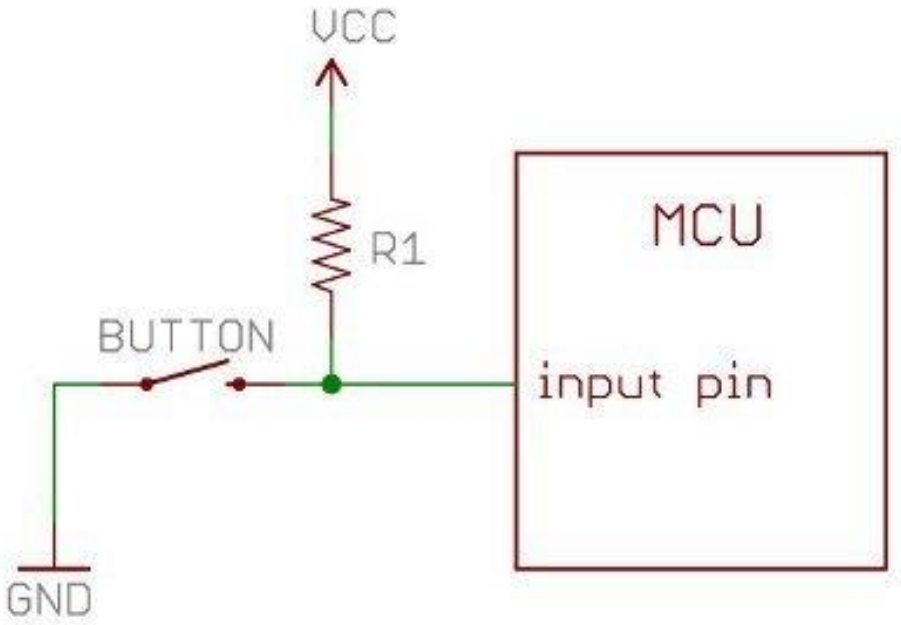
Pull-Up



Pull-Down

Os resistores “pull up” e “pull down” são utilizados para evitar flutuação em pinos digitais configurados como pinos de entrada (INPUT). Em geral, é necessário implementar externamente, mas muitas vezes há resistores “pull up” implementados internamente em alguns pinos do microcontrolador.

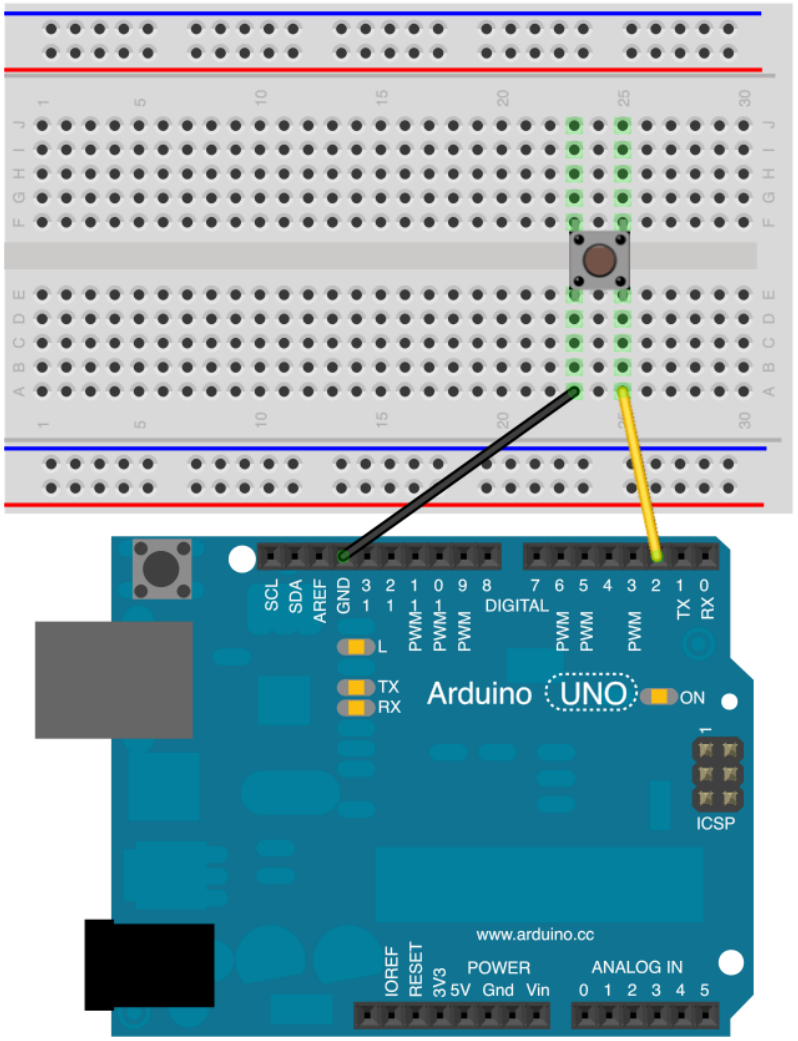
Entrada Digital – Resistor ”pull up”



No caso dos microcontroladores Atmega do Arduino, já existem resistores “pull up” internos em todos os pinos digitais e analógicos.

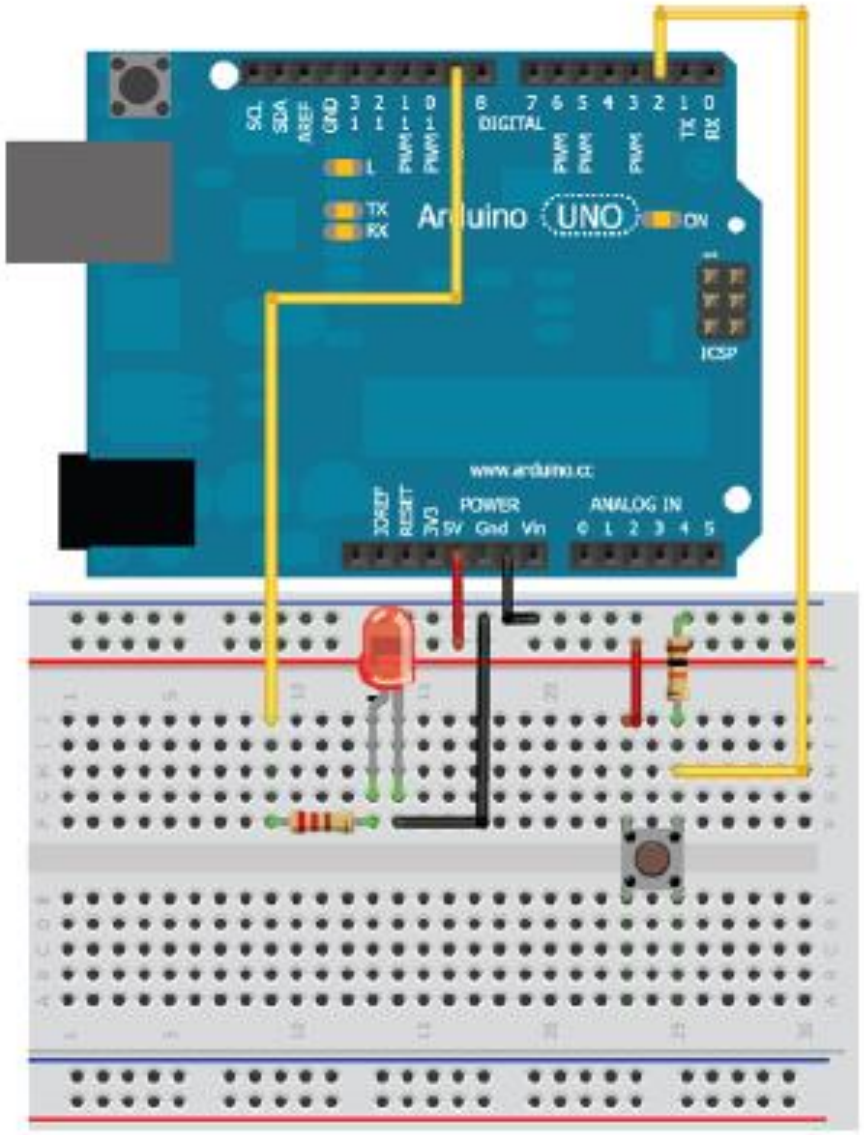
OBS: Só use resistores “pull up” nos pinos analógicos caso utilize estes como pinos digitais.

Leitura de porta digital com ”pull-up” interno




```
// EntradaPullup

void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP);
  pinMode(13, OUTPUT);
}

void loop() {
  int sensorVal = digitalRead(2);
  Serial.println(sensorVal);
  if (sensorVal == HIGH) {
    digitalWrite(13, LOW);
  } else {
    digitalWrite(13, HIGH);
  }
}
```

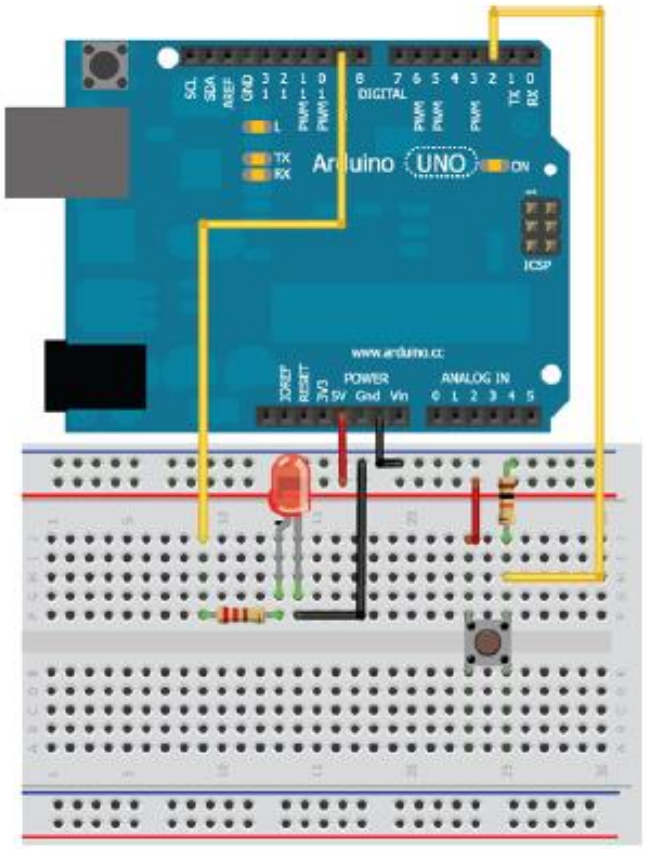
Acende LED com resistor ”pull-up”




```
// BotaoLED

const int LED = 13;
const int BOTAO = 2;

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

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

Botão como interruptor

```
const int LED = 13;
const int BOTAO = 2;
boolean lastBotao = LOW;
boolean currentBotao = LOW;
boolean ledOn = false;

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

boolean rebote(boolean last) {
  boolean situacao = digitalRead(BOTAO);
  if (last != situacao)
    delay(5);
  situacao = digitalRead(BOTAO);
  return situacao;
}

void loop() {
  currentBotao = rebote(lastBotao);
  if (lastBotao == LOW && currentBotao == HIGH)
  {
    ledOn = !ledOn;
  }
  lastBotao = currentBotao;
  digitalWrite(LED, ledOn);
}
```

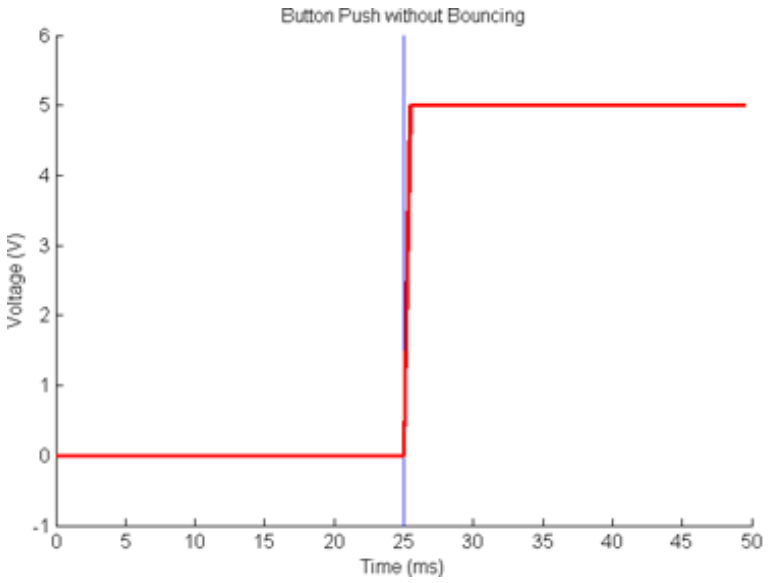


Variável temporizadora `millis()`

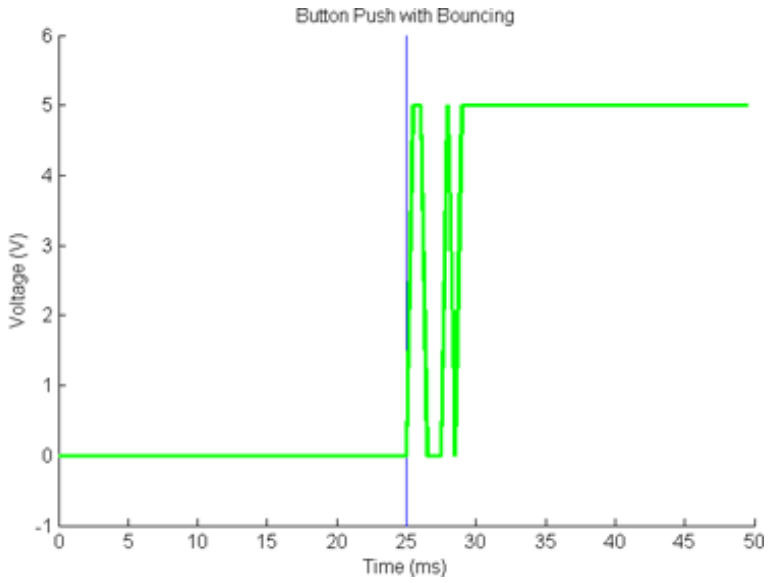
```
// TimerMillis

unsigned long time;

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

void loop(){
  Serial.print("Time: ");
  time = millis();
  Serial.println(time);
  delay(1000);
}
```

Entrada Digital: rebote do resistor ”pull up”

A Figura 1 abaixo ilustra o comportamento esperado do acionamento da chave momentânea



Na prática, o comportamento do acionamento da chave momentânea é representado pela Figura 2



O botão é fisicamente acionado durante 25 ms. Seria esperado que durante este interval de tempo o nível lógico da chave seria alto, porém a chave oscila entre os níveis alto e baixo, conforme mostra a Figura 2 acima.

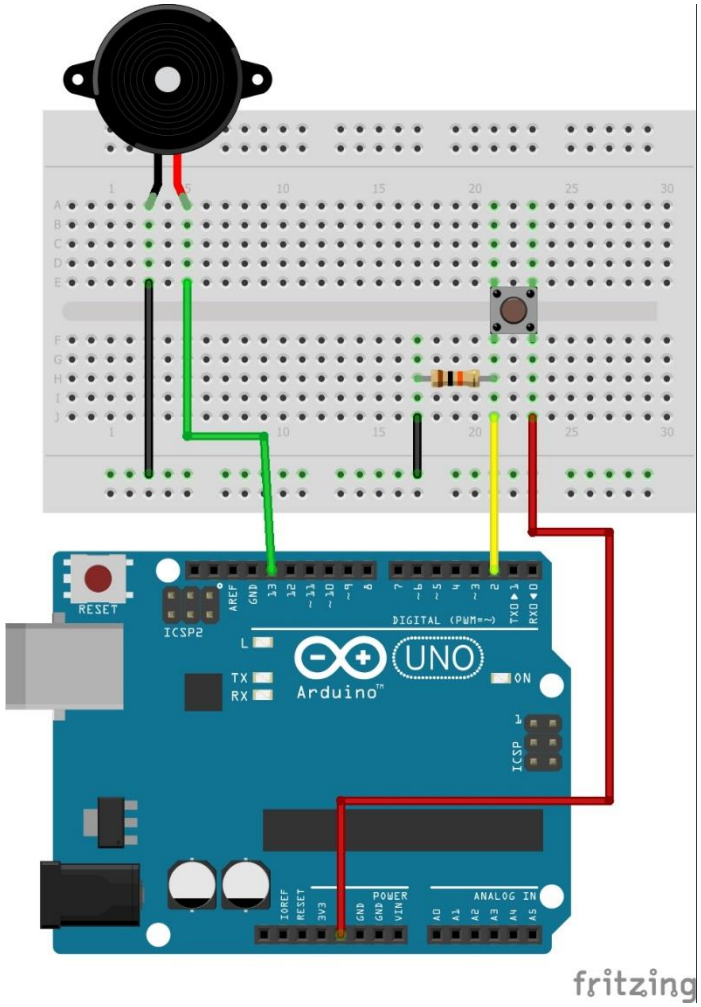
Sketch para eliminar o efeito do rebote (*debounce*)

```
/* Debounce

  Each time the input pin goes from LOW to HIGH
(e.g. because of a push-button
  press), the output pin is toggled from LOW to
HIGH or HIGH to LOW. There's a
  minimum delay between toggles to debounce the
circuit (i.e. to ignore noise).

  The circuit:
- LED attached from pin 13 to ground
- pushbutton attached from pin 2 to +5V
- 10 kilohm resistor attached from pin 2 to ground
*/
```

Debounce

```
const int buttonPin = 2;
const int ledPin = 13;
int ledState = HIGH;
int buttonState;
int lastButtonState = LOW;
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, ledState);
}

void loop() {
  int reading = digitalRead(buttonPin);
  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }
  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != buttonState) {
      buttonState = reading;
      if (buttonState == HIGH) {
        ledState = !ledState;
      }
    }
  }
  digitalWrite(ledPin, ledState);
  lastButtonState = reading;
}
```

Campainha com botão




```
// Campainha

const int SIRENE = 13;
const int BOTAO = 2;

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

void loop()
{
  if (digitalRead(BOTAO) == LOW) {
    digitalWrite(SIRENE, LOW);
  } else {
    digitalWrite(SIRENE, HIGH);
  }
  int acionaBotao = digitalRead(BOTAO);
  Serial.println(acionaBotao);
}
```

Controle pisca LED com botão

```
// Controle de LED com botao

const int ledPin = 13; //led no pino 13
const int Botao = 2; //botao no pino 2

int EstadoBotao = 0; //Variavel para ler o status do pushbutton

void setup(){
  pinMode(ledPin, OUTPUT); //Pino do led será saída
  pinMode(Botao, INPUT); //Pino com botão será entrada
}

void loop(){
  EstadoBotao = digitalRead(Botao);
  if (EstadoBotao == HIGH){ //Se botão estiver pressionado (HIGH)
    digitalWrite(ledPin, HIGH); // acende o led do pino 13.
  }
  else{ //se não estiver pressionado
    digitalWrite(ledPin, LOW); //deixa o led do pino 13 apagado
  }
}
```

Controle da intensidade do LED
Usando laço de repetição `for`

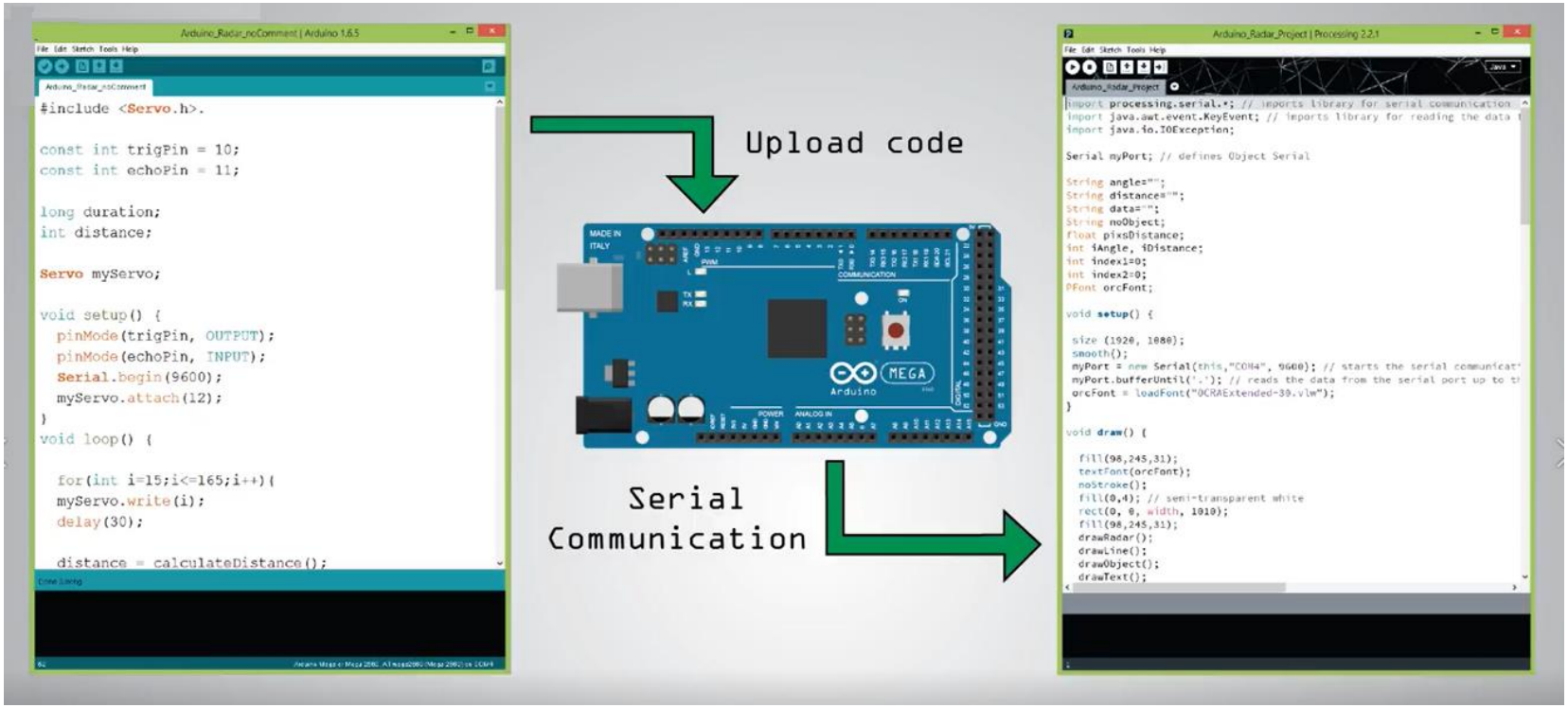
```
/* Esse programa escrito em C do Arduino aumenta e diminui gradativamente o brilho de um LED
conectado no pino PWM 10 do Arduino. */

int i=0;                                  // declaração da variável global inteira i iniciada com 0
void ledOn( );                            // declaração da função criada ledOn do tipo void
void setup( )  {
       pinMode(10,OUTPUT); // aqui 2 parâmetros são passados à função pinMode( )
}
void loop( )    {
       for (i=0; i <= 255; i++) ledOn( );          // aumenta o brilho do led
       for (i=255; i >= 0; i--) ledOn( );          // diminui o brilho do led
}
void ledOn( ) {                                    // função que acende o led
       analogWrite (10, i);      // o nº do pino e o valor de i são passados à função analogWrite( )
       delay (10);
}
```

Comunicação Arduino > Processing

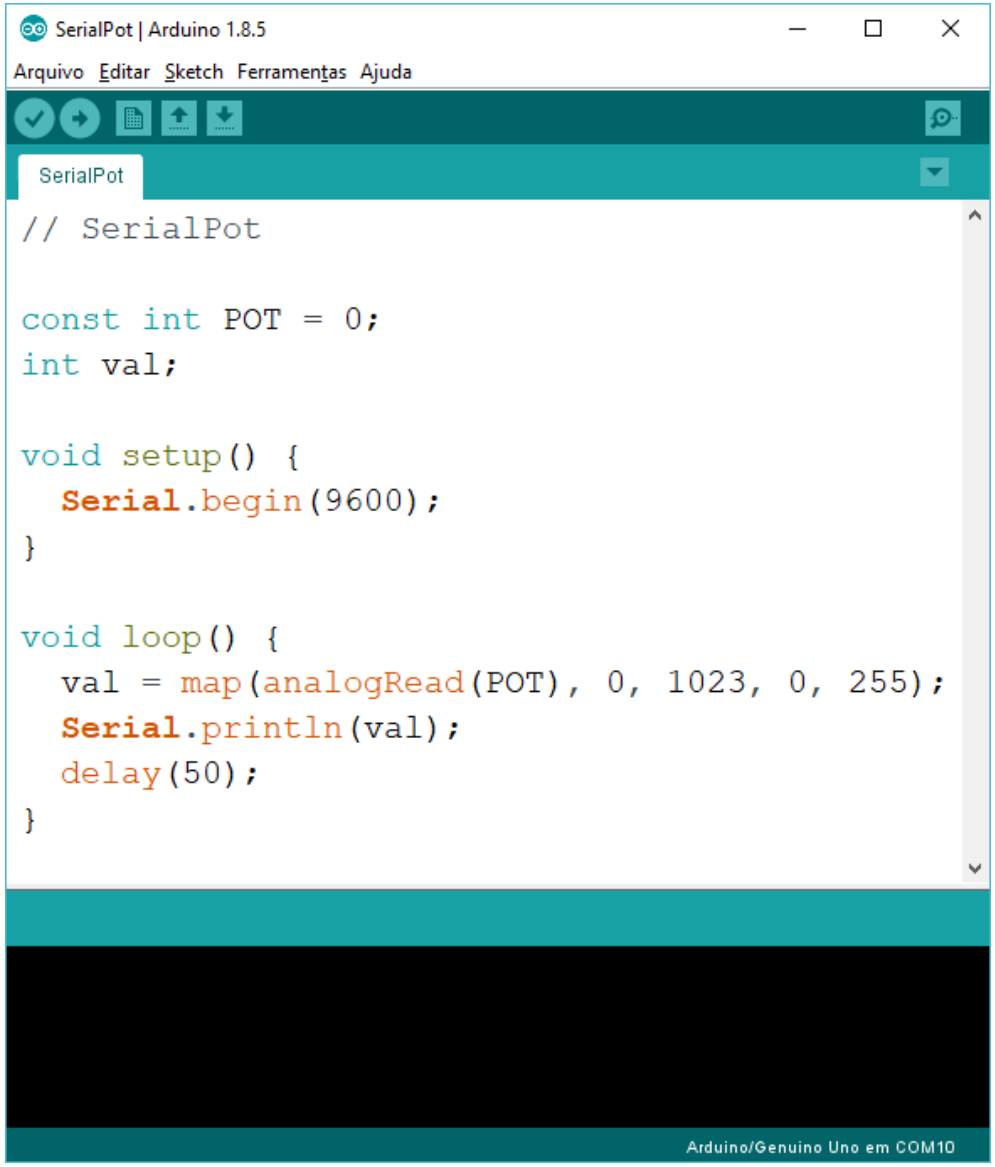
Comunicação Arduino > Processing

Sketch Arduino

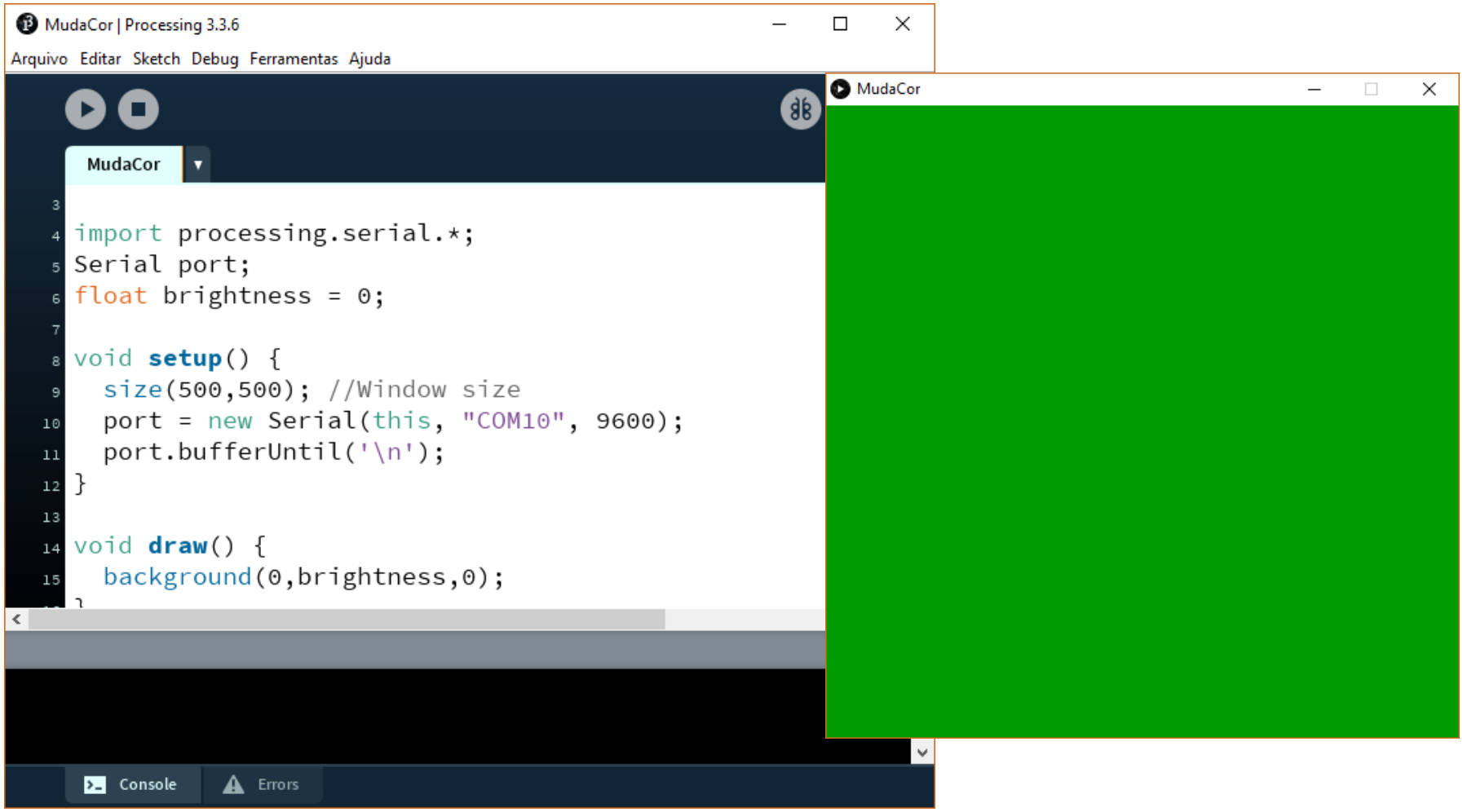
Sketch Processing



Arduino

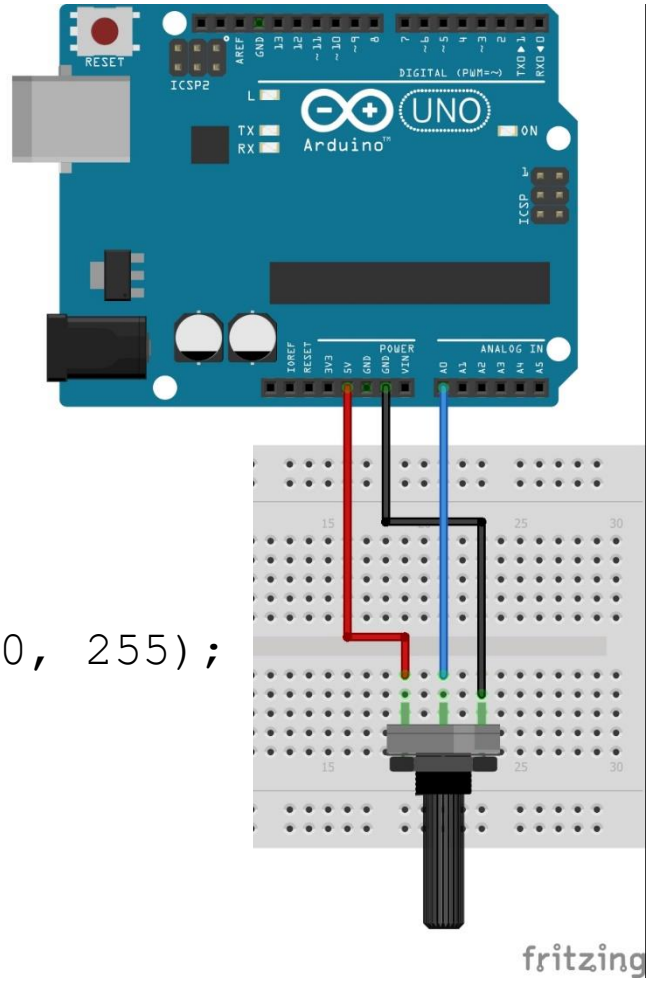

SerialPot | Arduino 1.8.5
Arquivo Editar Sketch Ferramentas Ajuda
SerialPot
// SerialPot
const int POT = 0;
int val;
void setup() {
Serial.begin(9600);
}
void loop() {
val = map(analogRead(POT), 0, 1023, 0, 255);
Serial.println(val);
delay(50);
}
Arduino/Genuino Uno em COM10

Processing


MudaCor | Processing 3.3.6
Arquivo Editar Sketch Debug Ferramentas Ajuda
MudaCor
import processing.serial.*;
Serial port;
float brightness = 0;
void setup() {
size(500,500); //Window size
port = new Serial(this, "COM10", 9600);
port.bufferUntil('\n');
}
void draw() {
background(0,brightness,0);
Console
Errors
MudaCor

Arduino: controle potenciométrico

```
// SerialPot

const int POT = 0;
int val;

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

void loop() {
  val = map(analogRead(POT), 0, 1023, 0, 255);
  Serial.println(val);
  delay(50);
}
```



Processing: controle de cor do canvas

```
// MudaCor Processing Sketch
// Importa e inicializa porta serial

import processing.serial.*;
Serial port;
float brightness = 0;

void setup() {
  size(500,500);
  port = new Serial(this, "COM10", 9600);
  port.bufferUntil('\n');
}

void draw() {
  background(0,brightness,0);
}

void serialEvent(Serial port) {
  brightness = float(port.readStringUntil('\n'));
}
```