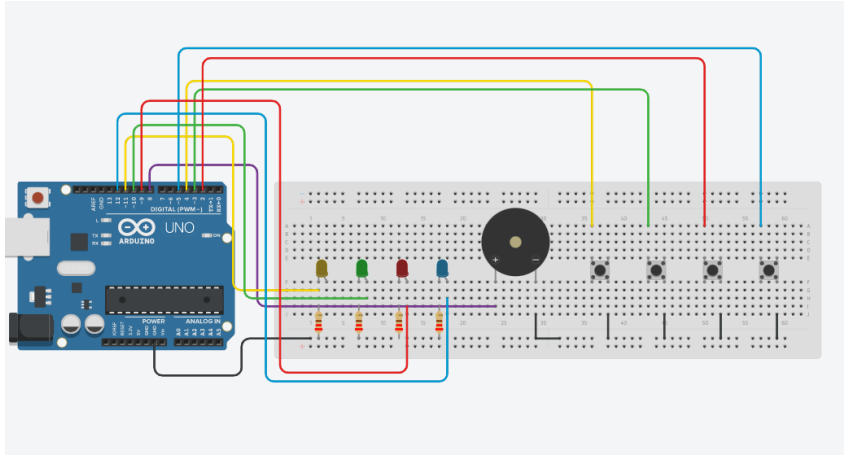
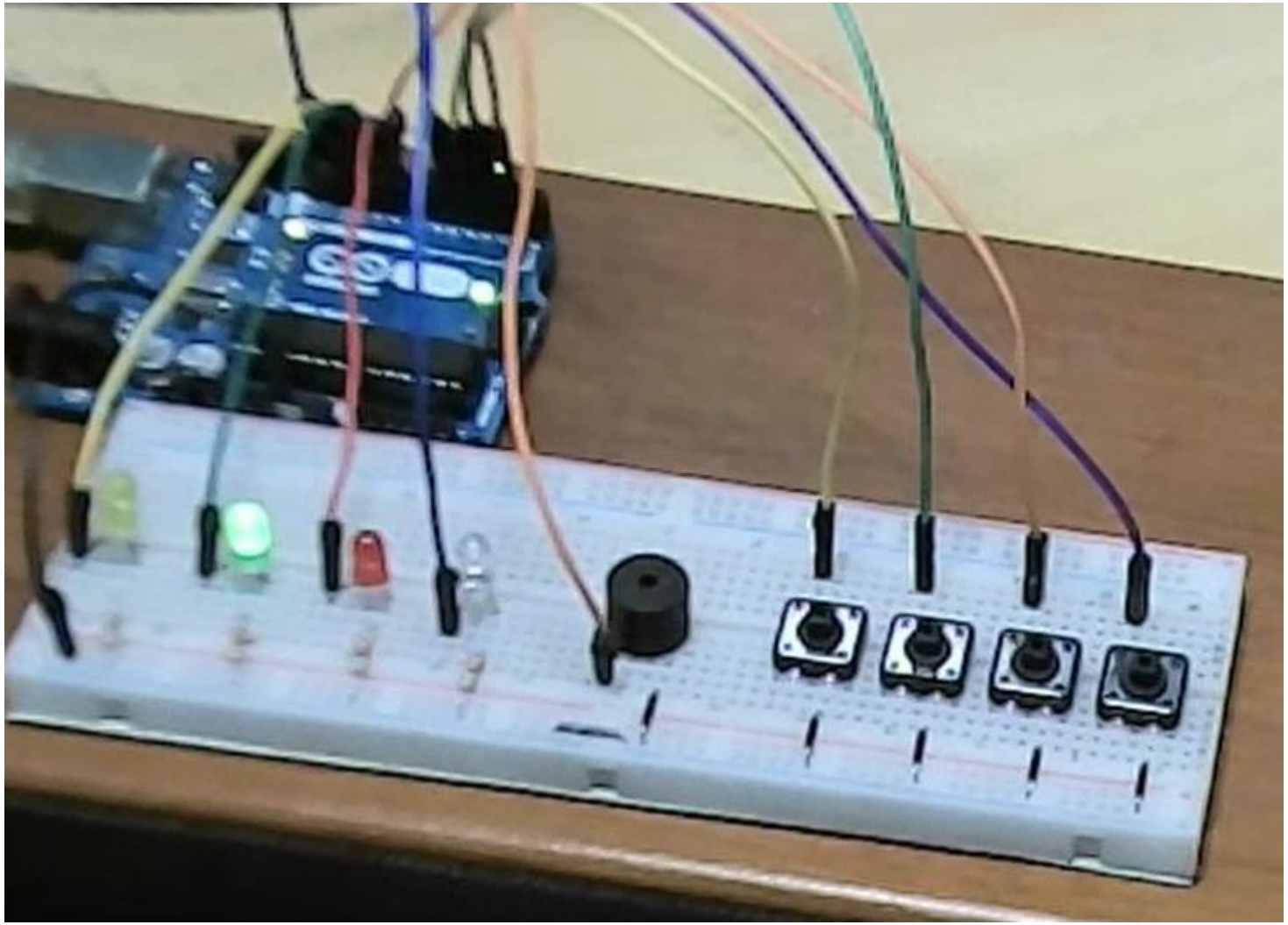
Professor Rafael Oliveira (Tio Rafa)

MAMUTE Eletrônica

GENIUS FEITO COM ARDUINO

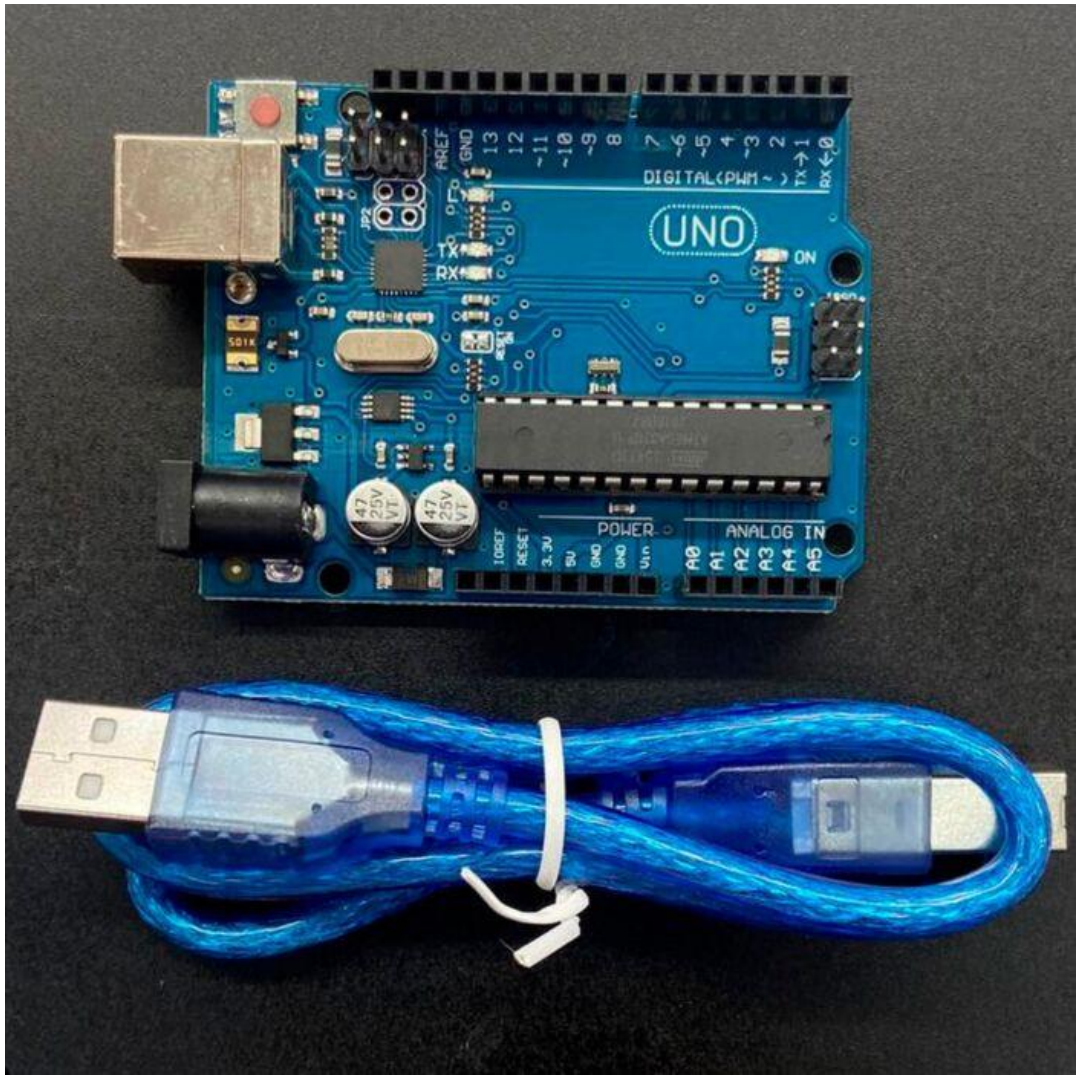




Componentes Eletrônicos você encontra aqui na Mamute Eletrônica | Kit de Componentes do Projeto Jogo da Memória - GENIUS - Prof. Rafael Oliveira (mamuteeletronica.com.br)

Materiais necessários:

- 4 LEDs (vermelho, verde, azul e amarelo)
- 4 resistores de 220 ohms
- 4 botões (push buttons)
- 1buzzer
- Fios jumper macho/macho
- Protoboard

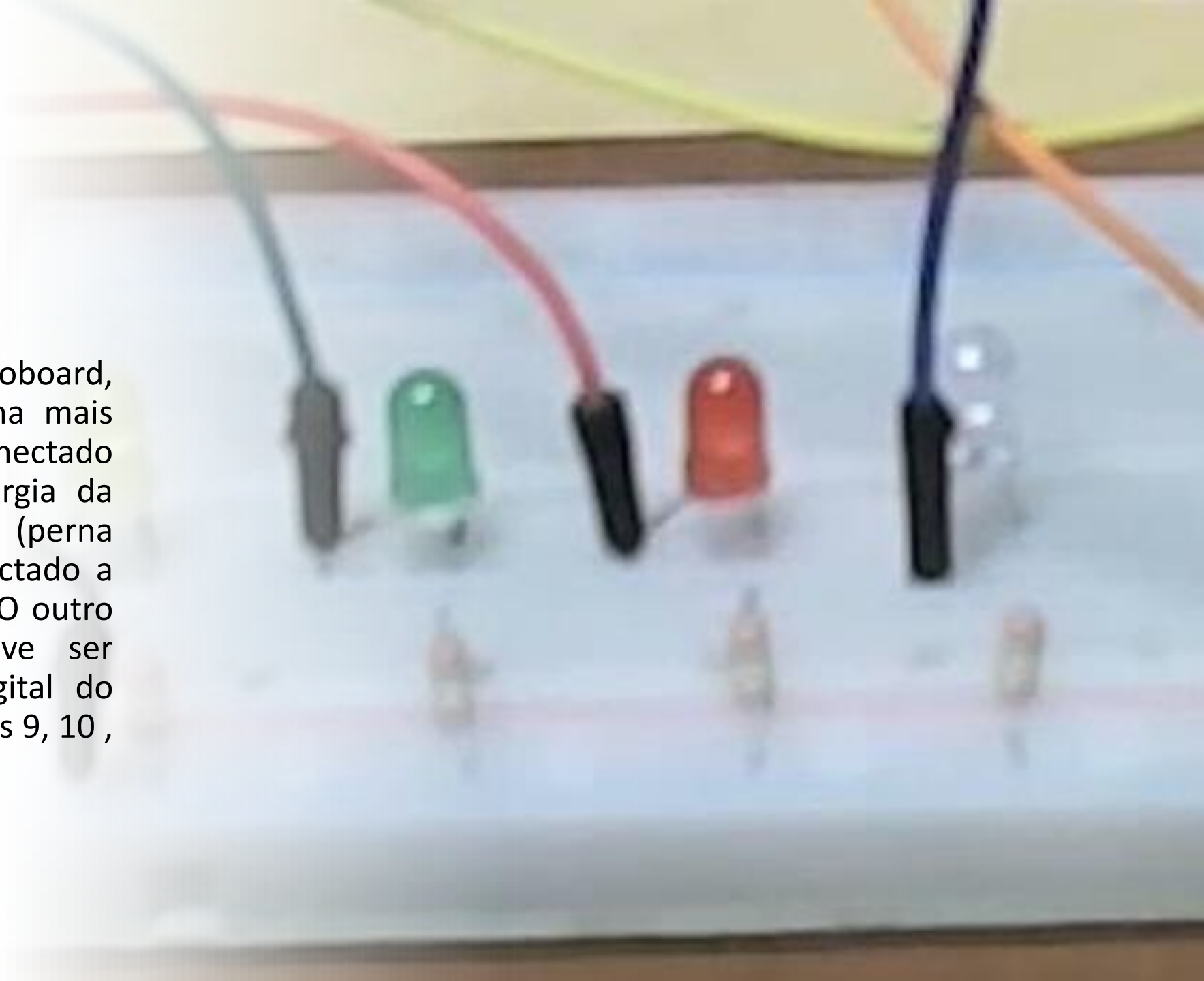


1 placa Arduino Uno

[Componentes Eletrônicos você encontra aqui na Mamute Eletrônica | Placa Arduino Uno R3 com Cabo USB (mamuteeletronica.com.br)]

MONTAGEM

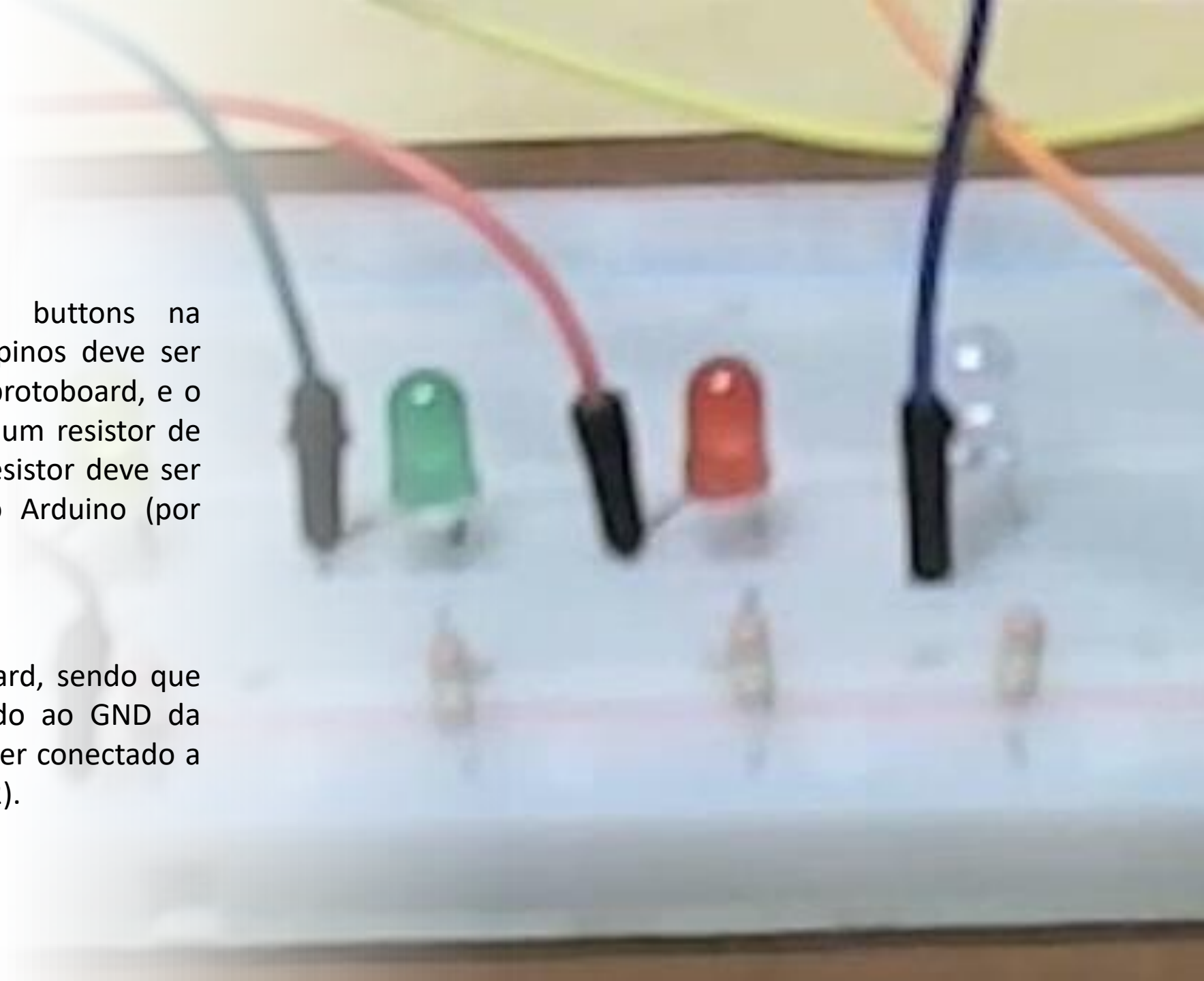
1: Conectar os LEDs na protoboard, sendo que o anodo (perna mais longa) do LED deve ser conectado a uma das faixas de energia da protoboard, e o catodo (perna mais curta) deve ser conectado a um resistor de 220 ohms. O outro extremo do resistor deve ser conectado a um pino digital do Arduino (por exemplo, pinos 9, 10 , 11 e 12).

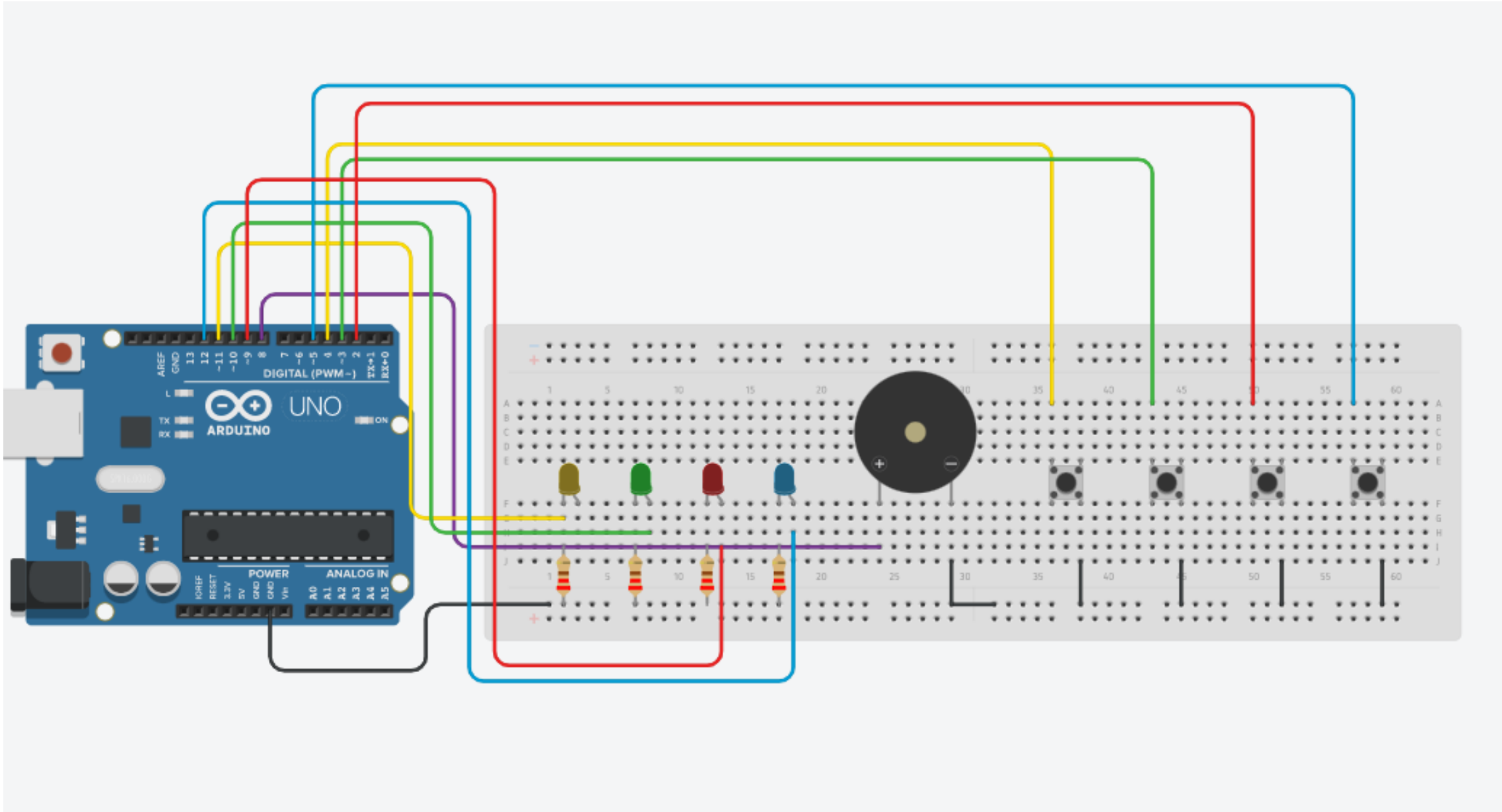


MONTAGEM

2: Conectar os botões push buttons na protoboard, sendo que um dos pinos deve ser conectado à faixa de energia da protoboard, e o outro pino deve ser conectado a um resistor de 220 ohms. O outro extremo do resistor deve ser conectado a um pino digital do Arduino (por exemplo, pinos 2, 3, 4 e 5).

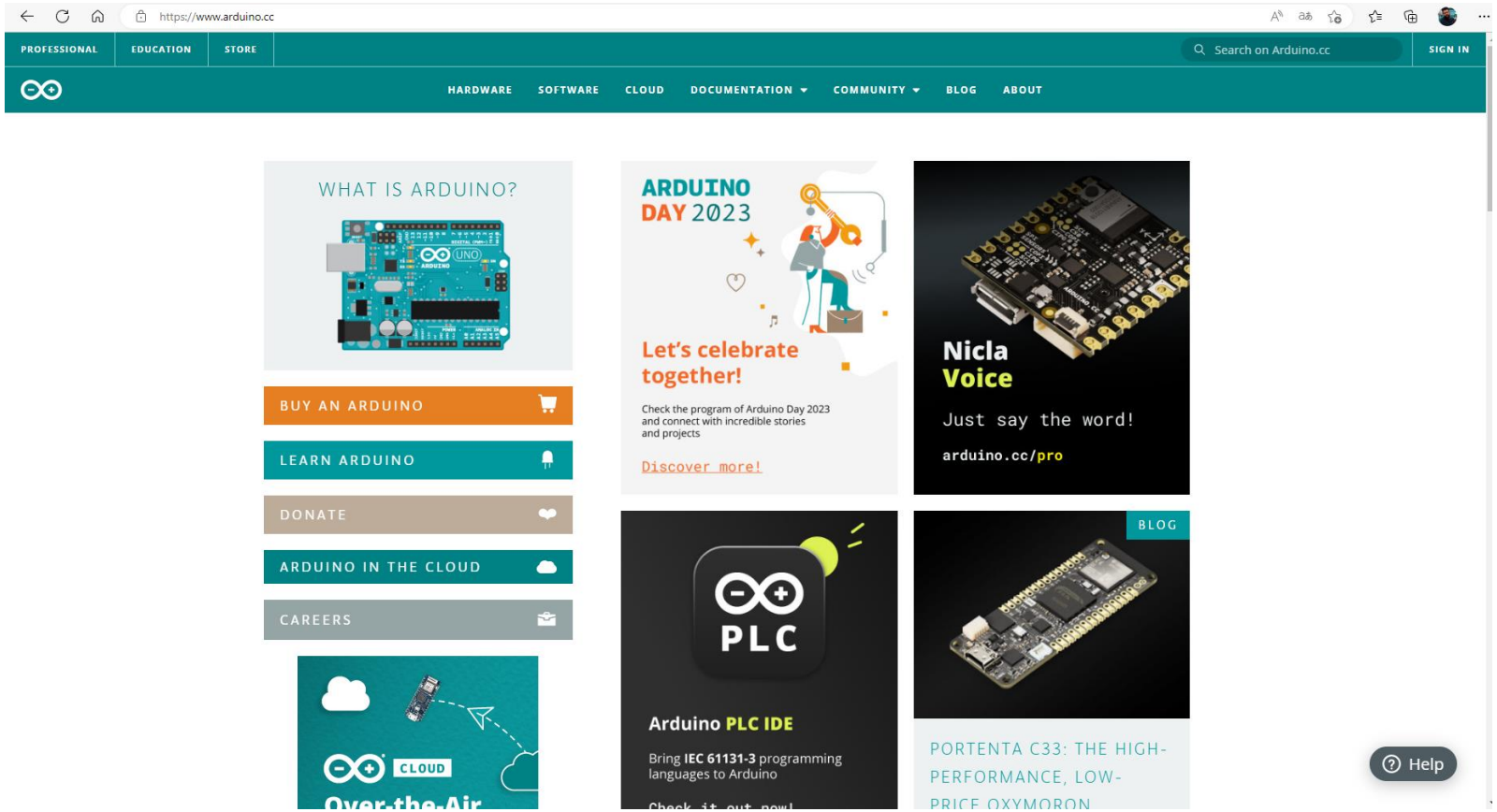
3: Conectar o buzzer na protoboard, sendo que um dos pinos deve ser conectado ao GND da protoboard, e o outro pino deve ser conectado a um pino digital do Arduino (pino 2).




DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN

Instalação do ambiente de desenvolvimento integrado (IDE) do Arduino

Acesse: www.arduino.cc



https://www.arduino.cc/en/software

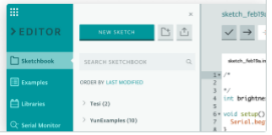
PROFESSIONAL | EDUCATION | STORE | Search on Arduino.cc | SIGN IN

HARDWARE | SOFTWARE | CLOUD | DOCUMENTATION | COMMUNITY | BLOG | ABOUT

Arduino Web Editor

Start coding online and save your sketches in the cloud. The most up-to-date version of the IDE includes all libraries and also supports new Arduino boards.

CODE ONLINE GETTING STARTED



Downloads

Arduino IDE 2.0.4

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the **Arduino IDE 2.0 documentation**.

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on **GitHub**.

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

macOS Intel, 10.14: "Mojave" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

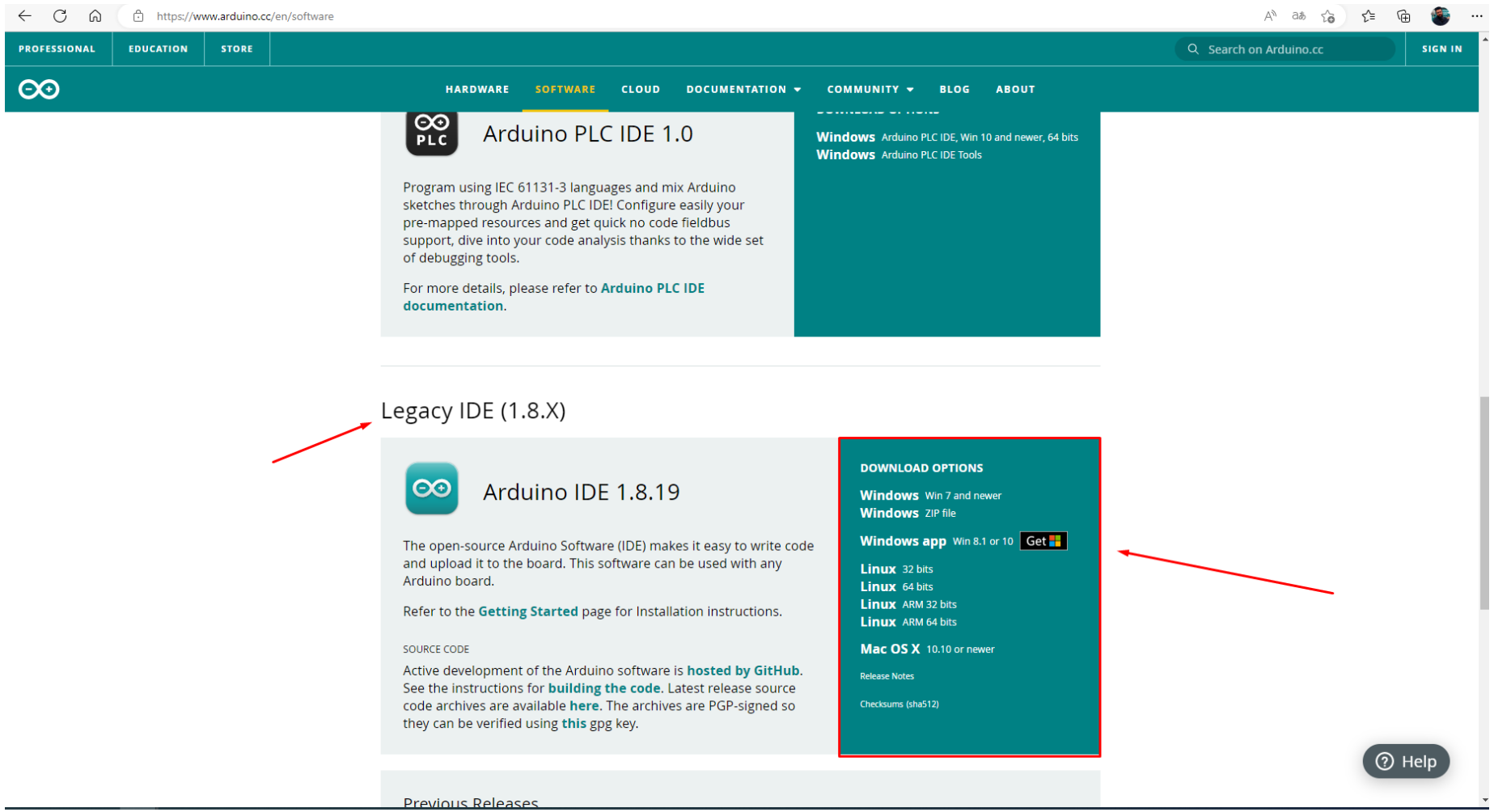
Release Notes

Nightly Builds

Download a **preview of the incoming release** with the most updated features and bugfixes

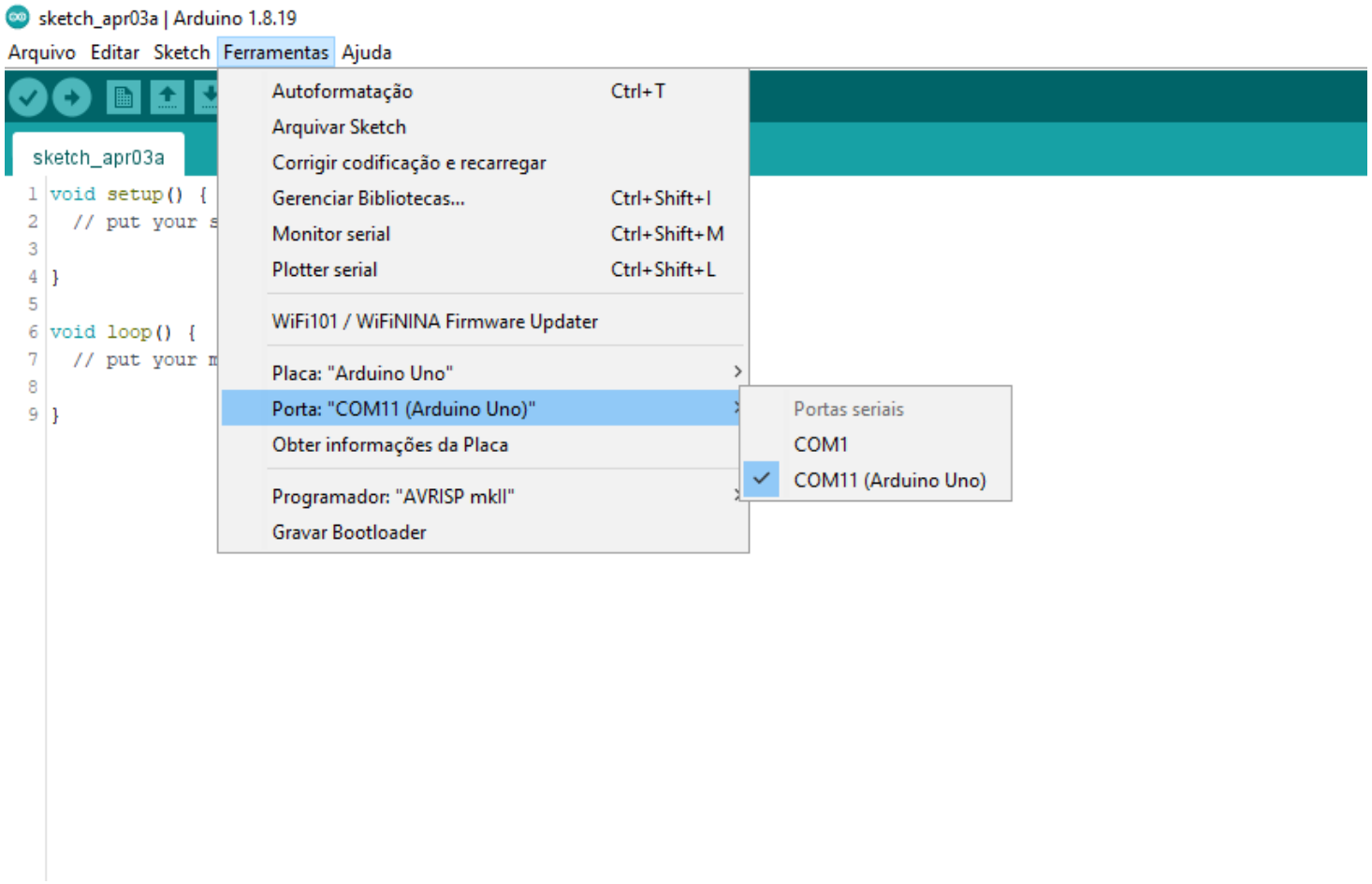
Help

https://www.arduino.cc/en/software


https://www.arduino.cc/en/software
PROFESSIONAL
EDUCATION
STORE
Search on Arduino.cc
SIGN IN
HARDWARE
SOFTWARE
CLOUD
DOCUMENTATION
COMMUNITY
BLOG
ABOUT
Arduino PLC IDE 1.0
Windows Arduino PLC IDE, Win 10 and newer, 64 bits
Windows Arduino PLC IDE Tools
Program using IEC 61131-3 languages and mix Arduino sketches through Arduino PLC IDE! Configure easily your pre-mapped resources and get quick no code fieldbus support, dive into your code analysis thanks to the wide set of debugging tools.
For more details, please refer to Arduino PLC IDE documentation.
Legacy IDE (1.8.X)
Arduino IDE 1.8.19
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.
SOURCE CODE
Active development of the Arduino software is hosted by GitHub. See the instructions for building the code. Latest release source code archives are available here. The archives are PGP-signed so they can be verified using this gpg key.
DOWNLOAD OPTIONS
Windows Win 7 and newer
Windows ZIP file
Windows app Win 8.1 or 10 Get
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits
Mac OS X 10.10 or newer
Release Notes
Checksums (sha512)
Help
Previous Releases

Verificando a comunicação do Arduino com seu Computador

Para enviar o código, o Arduino precisa estar conectado e confirmado a porta em que ele está configurado, como na imagem abaixo



Carregue o código no Arduino.

CÓDIGO

Genius_TioRafa | Arduino 1.8.19

Arquivo Editar Sketch Ferramentas Ajuda

Genius_TioRafa § | pitches.h

```
// Inclui a biblioteca "pitches.h" para utilizar as notas musicais
#include "pitches.h"

/* Constantes - define os números dos pinos dos LEDs,
botões, do speaker e também as notas do jogo: */

const byte ledPins[] = {9, 10, 11, 12};
const byte buttonPins[] = {2, 3, 4, 5};
#define SPEAKER_PIN 8

#define MAX_GAME_LENGTH 100

const int gameTones[] = { NOTE_G3, NOTE_C4, NOTE_E4, NOTE_G5};

/* Variáveis globais - armazenam o estado do jogo */
byte gameSequence[MAX_GAME_LENGTH] = {0};
byte gameIndex = 0;

/**
   Configura a placa Arduino e inicializa a comunicação serial
*/
void setup() {
  Serial.begin(9600);
  for (byte i = 0; i < 4; i++) {
    pinMode(ledPins[i], OUTPUT);
    pinMode(buttonPins[i], INPUT_PULLUP);
  }
  pinMode(SPEAKER_PIN, OUTPUT);
  // A linha seguinte prepara o gerador de números aleatórios.
 // Ela assume que o pino A0 está flutuante (desconectado):
  randomSeed(analogRead(A0));
}

/**
 Acende o LED indicado e toca uma nota correspondente
*/
void lightLedAndPlayTone(byte ledIndex) {
  digitalWrite(ledPins[ledIndex], HIGH);
  tone(SPEAKER_PIN, gameTones[ledIndex]);
  delay(300);
  digitalWrite(ledPins[ledIndex], LOW);
  noTone(SPEAKER_PIN);
}
```

MAMUTE Eletrônica

CÓDIGO

```
/**
   Toca a sequência atual de notas que o usuário deve repetir
*/
void playSequence() {
  for (int i = 0; i < gameIndex; i++) {
    byte currentLed = gameSequence[i];
    lightLedAndPlayTone(currentLed);
    delay(50);
  }
}

/**
   Espera até que o usuário pressione um dos botões,
e retorna o índice desse botãon
*/
byte readButtons() {
  while (true) {
    for (byte i = 0; i < 4; i++) {
      byte buttonPin = buttonPins[i];
      if (digitalRead(buttonPin) == LOW) {
        return i;
      }
    }
    delay(1);
  }
}

/**
  Toca a sequência de game over e exibe a pontuação do jogador
*/
void gameOver() {
  Serial.print("Game over! your score: ");
  Serial.println(gameIndex - 1);
  gameIndex = 0;
  delay(200);
/**
  Toca uma sequência de notas para indicar o fim do jogo
```

MAMUTE Eletrônica

CÓDIGO

```
  tone(SPEAKER_PIN, NOTE_DS5);
  delay(300);
  tone(SPEAKER_PIN, NOTE_D5);
  delay(300);
  tone(SPEAKER_PIN, NOTE_CS5);
  delay(300);
  for (byte i = 0; i < 10; i++) {
    for (int pitch = -10; pitch <= 10; pitch++) {
      tone(SPEAKER_PIN, NOTE_C5 + pitch);
      delay(5);
    }
  }
  noTone(SPEAKER_PIN);
  delay(500);
}

/**
   Obtém a entrada do usuário e compara com a sequência esperada
*/
bool checkUserSequence() {
  for (int i = 0; i < gameIndex; i++) {
    byte expectedButton = gameSequence[i];
    byte actualButton = readButtons();
    lightLedAndPlayTone(actualButton);
    if (expectedButton != actualButton) {
      return false;
    }
  }

  return true;
}

/**
 Reproduz um som de vitória sempre que o usuário termina um nível
```

MAMUTE Eletrônica

CÓDIGO

```
118 void playLevelUpSound() {
119   tone(SPEAKER_PIN, NOTE_E4);
120   delay(150);
121   tone(SPEAKER_PIN, NOTE_G4);
122   delay(150);
123   tone(SPEAKER_PIN, NOTE_E5);
124   delay(150);
125   tone(SPEAKER_PIN, NOTE_C5);
126   delay(150);
127   tone(SPEAKER_PIN, NOTE_D5);
128   delay(150);
129   tone(SPEAKER_PIN, NOTE_G5);
130   delay(150);
131   noTone(SPEAKER_PIN);
132 }
133
134 /**
135    Loop principal do jogo
136 */
137 void loop() {
138   // Add a random color to the end of the sequence
139   gameSequence[gameIndex] = random(0, 4);
140   gameIndex++;
141   if (gameIndex >= MAX_GAME_LENGTH) {
142     gameIndex = MAX_GAME_LENGTH - 1;
143   }
144
145   playSequence();
146   if (!checkUserSequence()) {
147     gameOver();
148   }
149
150   delay(300);
151
152   if (gameIndex > 0) {
153     playLevelUpSound();
154     delay(300);
155   }
156 }
```