



Oficina de Entrada e Saída com Arduino



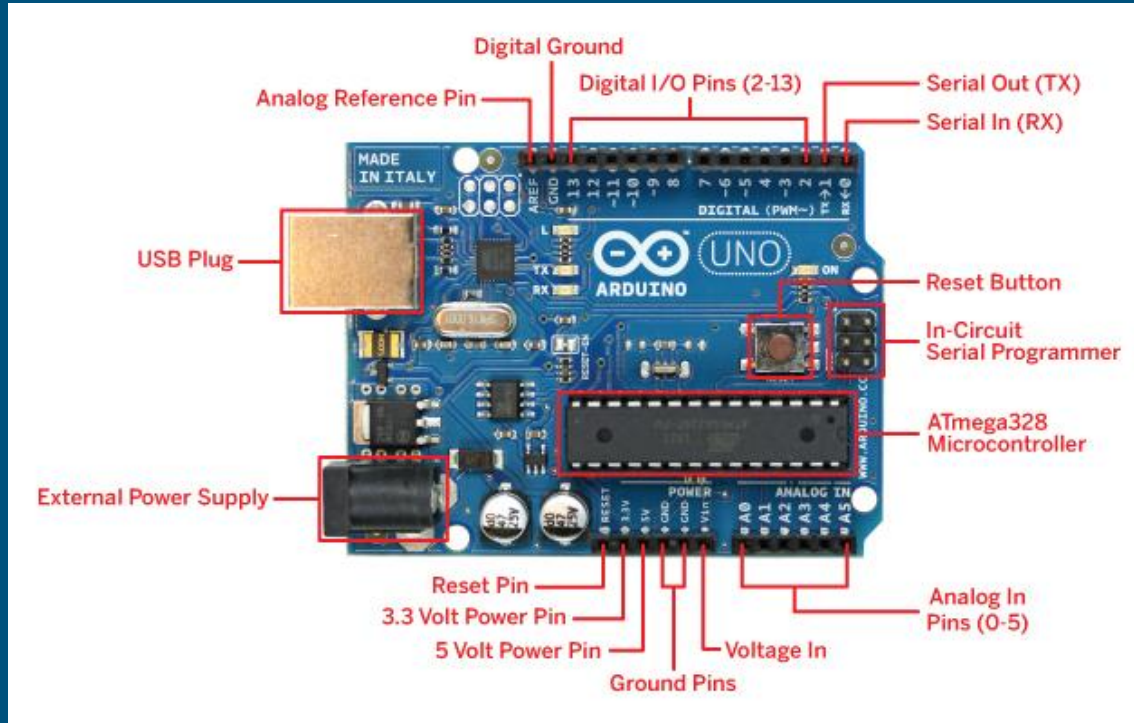
Grupo HardwareLivreUSP



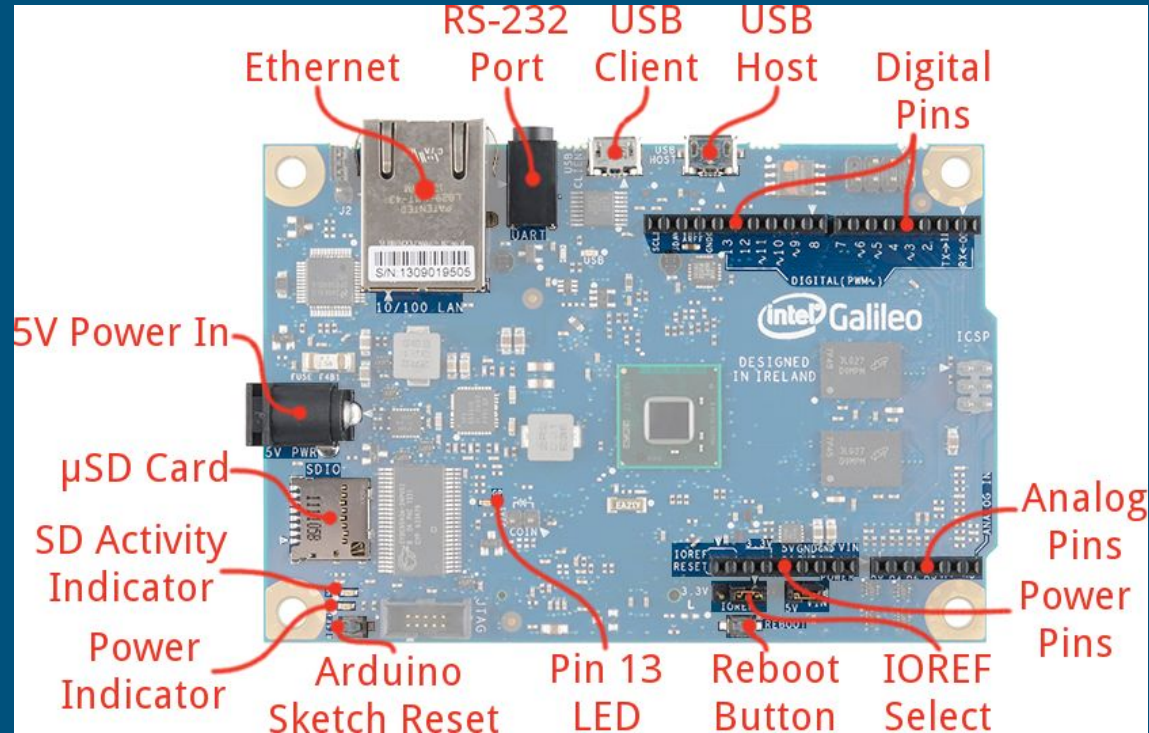
Conteúdo da oficina

- Estrutura de uma placa Arduino
- Conceitos de sinais elétricos
 - Sinal digital, analógico, PWM
- Ambiente de desenvolvimento
 - Os blocos básicos de um programa em Arduino
 - Receber e enviar sinais digitais
- Projeto Genius (Jogo da memória)

○ Arduino UNO

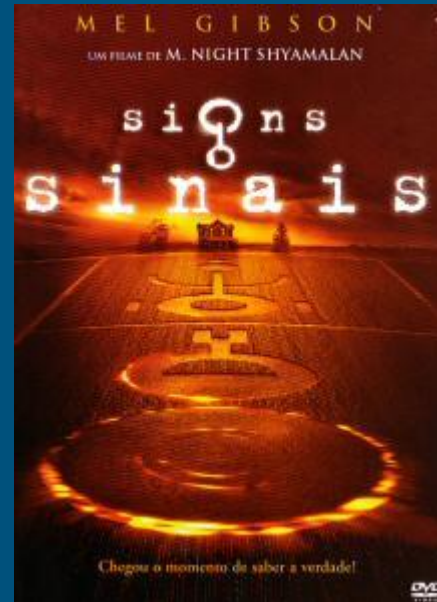


A Galileo



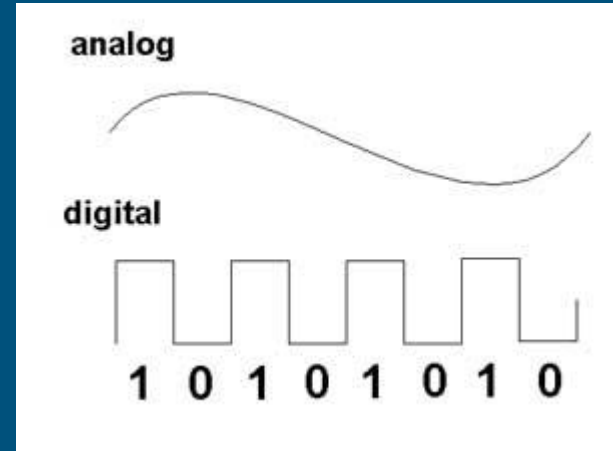
Entrada e Saída (I/O)

- Entrada e saída de dados / sinais em relação ao microcontrolador
- Entrada
 - Um sinal externo é enviado ao microcontrolador
- Saída
 - O microcontrolador gera um sinal e o envia ao meio externo
- Que sinais são esses?



Analógico Vs Digital

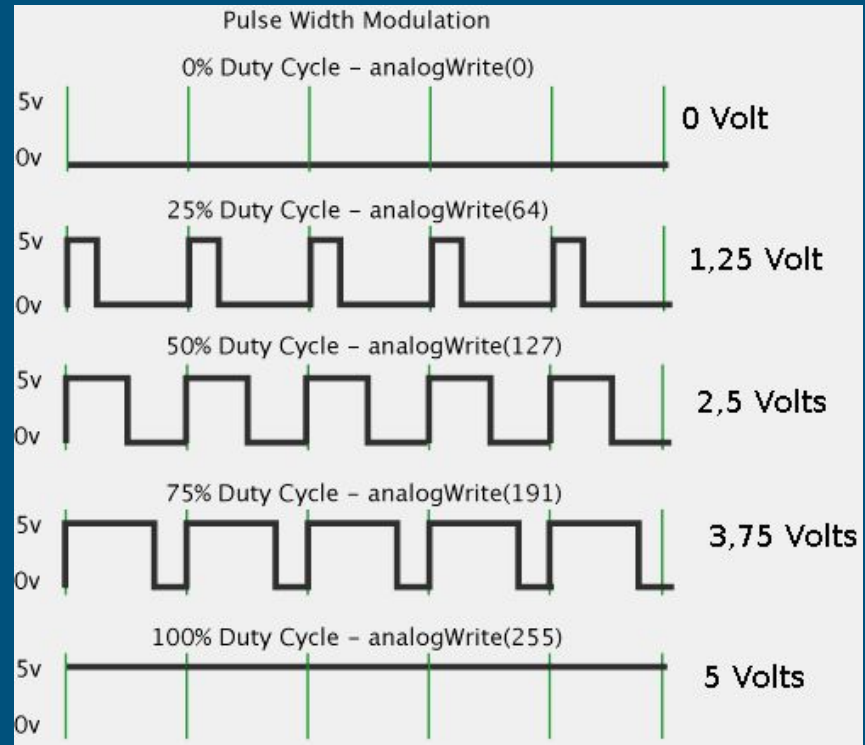
- Sinal analógico
 - Pode assumir qualquer valor entre o limite mínimo e máximo
- Sinal digital
 - Discretizado (geralmente apenas 0 ou 1)



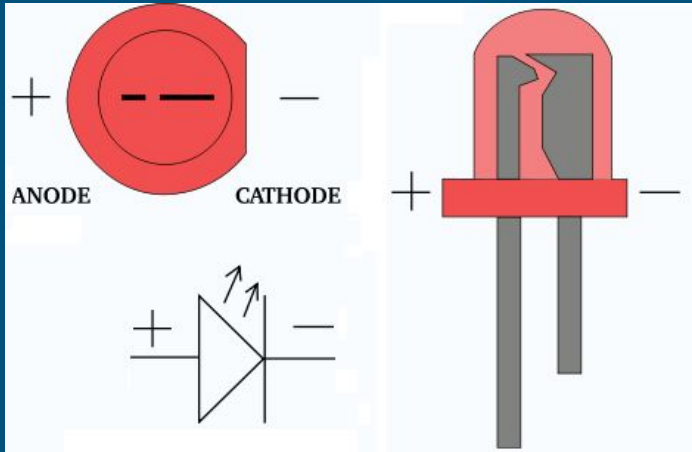
Pulse Width Modulation (PWM)

- Pulsos (ligado e desligado / 1 e 0) durante determinada proporção de tempo em um ciclo constante
- Simula um sinal analógico com um sinal digital

<https://www.arduino.cc/en/Tutorial/PWM>



O LED (Light Emitting Diode)



- Usualmente operam em um nível de tensão de 1,6 a 3,3 volts e sob uma corrente elétrica próxima de 20 mA
- Possuem polaridade
 - Perna maior é positiva
 - Perna menor é negativa

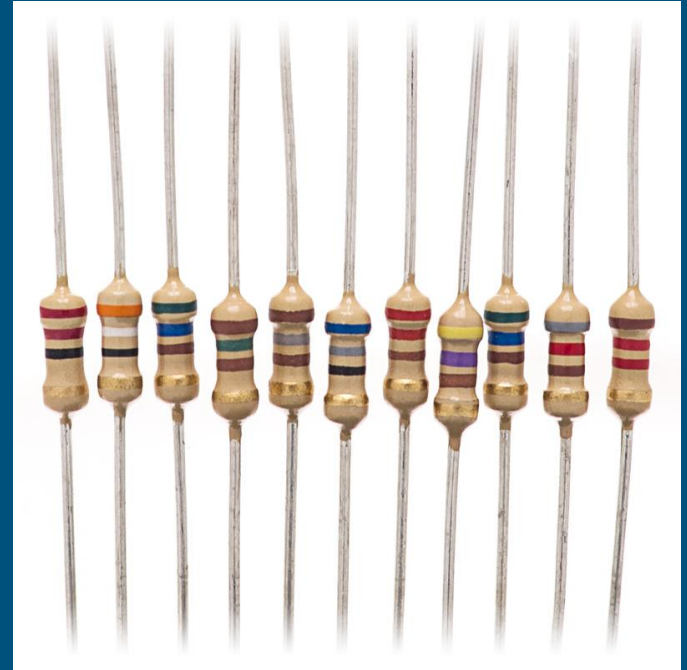


https://en.wikipedia.org/wiki/Light-emitting_diode

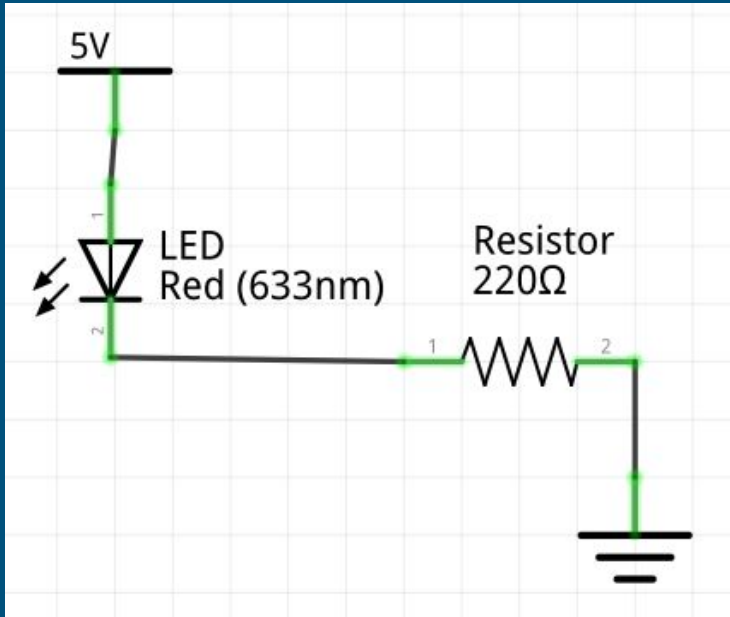
Resistores



- Dificultam a passagem de corrente elétrica
- Provocam queda do potencial elétrico de uma ponta para a outra



LED e Resistor



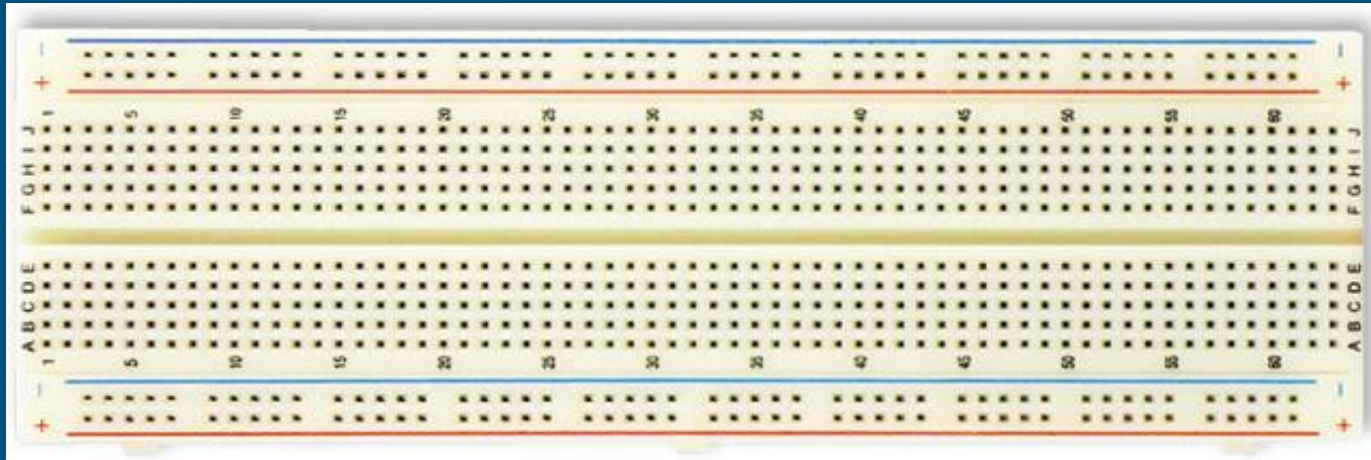
Normalmente junto ao LED é usado um resistor de 220 a 330 ohms para evitar que o LED queime.

Fundamento teórico:

- Primeira lei de Ohm: $i = V / R$
- Lei de Kirchhoff das tensões (LKT)

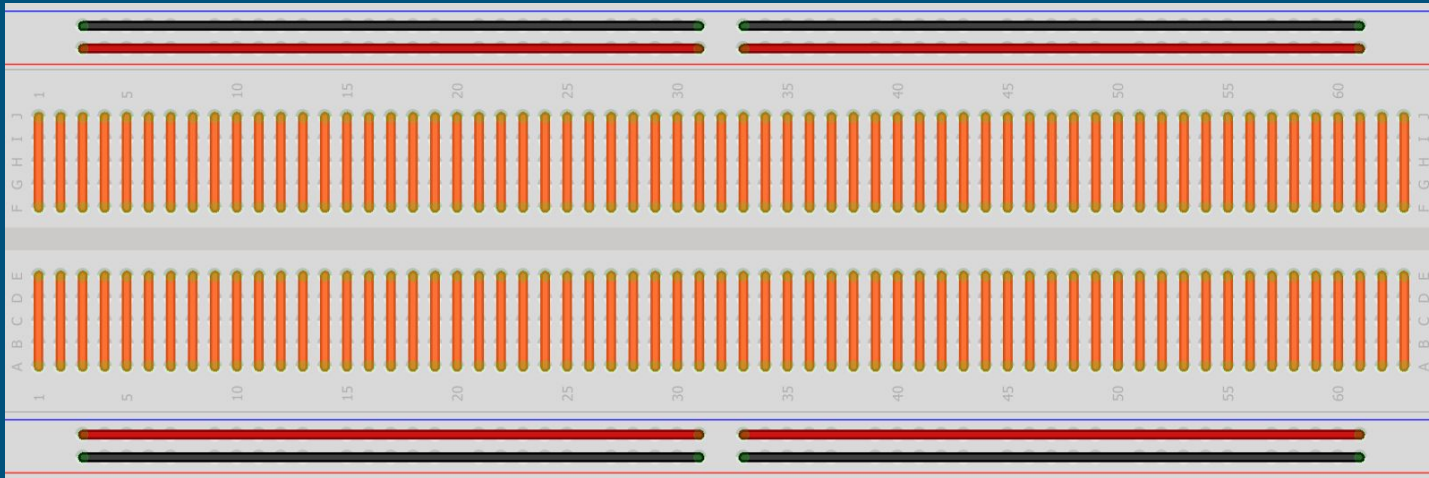
Protoboard / breadboard

Muito usada para fazer testes e protótipos iniciais.



Protoboard / breadboard

- As ilhas no centro da placa estão conectadas horizontalmente
- As ilhas nas laterais da placa estão conectadas verticalmente



Programando um Arduino

- Instale a IDE do Arduino obtida em:
 - <https://www.arduino.cc/en/Main/Software>
- Para a IDE da Galileo visite:
 - <http://www.intel.com/content/www/us/en/support/boards-and-kits/intel-galileo-boards/000005614.html>



The screenshot shows the Arduino IDE interface with a sketch named 'memorygame'. The code is as follows:

```
File Edit Sketch Tools Help
memorygame
level = 1;
start = 1;
}

void loop()
{
  if(start) {
    for (int i = 0; i < length; i++) {
      playNote(notes[i], beats[i]*tempo);
      delay(tempo/2);
    }
    lcd.clear();
    lcd.print("Press Again to");
    lcd.setCursor(0, 1);
    lcd.print("Start");
    lcd.setCursor(5, 1);
    lcd.write((uint8_t)0);
    delay(100);
    lcd.setCursor(5, 1);
    lcd.write((uint8_t)1);
    delay(100);
  }

  if (getButtonPress()) {
    start = 0;
    lcd.clear();
    lcd.setCursor(0, 0);
    if (user == game && game != 0 && score < 10) {
      lcd.print("You Are Correct");
      score++;
      lcd.setCursor(0, 1);
      lcd.print("Score:");
      lcd.setCursor(6, 1);
    }
  }
}
```

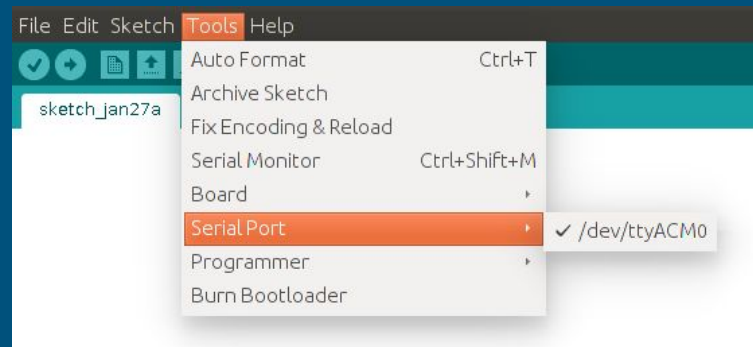
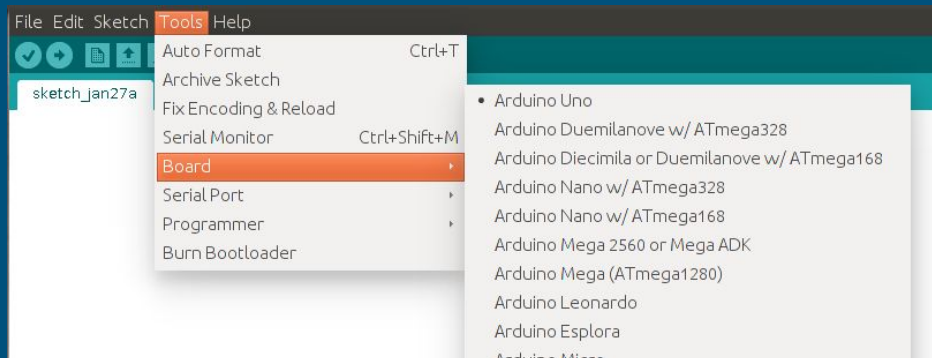
Done compiling.

Binary sketch size: 7.912 bytes (of a 32.256 byte maximum)

14

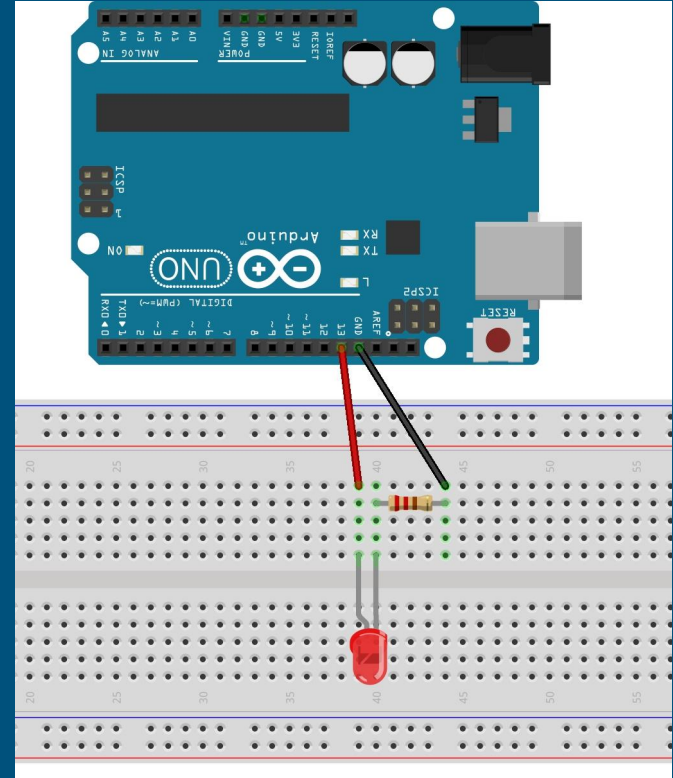
Programando um Arduino

- Conecte o Arduino em uma porta USB do computador
- Selecione a placa Arduino em Tools -> Board
- Selecione a porta USB em Tools -> Serial Port



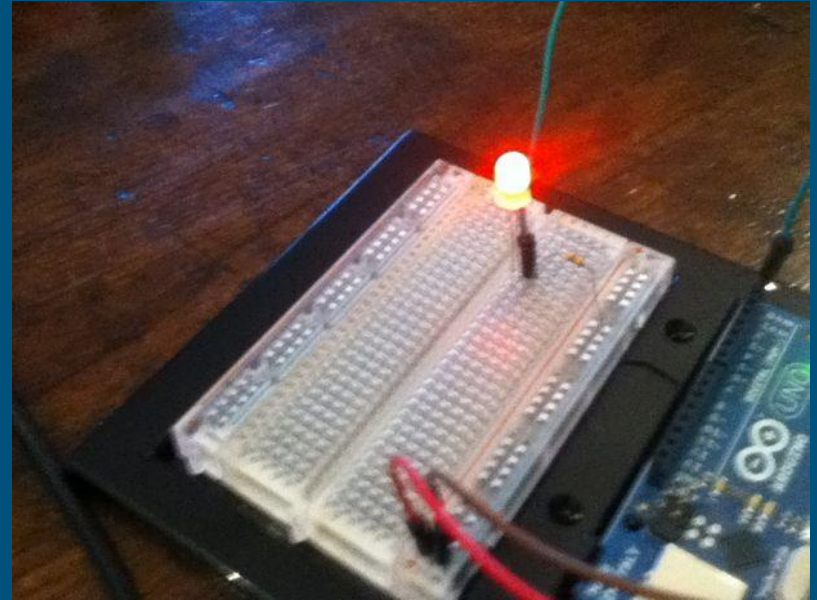
Programando um Arduino

- Monte o projeto na protoboard
- Hardware necessário
 - Placa Arduino
 - Protoboard
 - 1 LED
 - 1 Resistor de 220 Ohms
 - Fios para conectar os componentes



Programando um Arduino

- Abra o código do exemplo em
File -> Examples -> Basics -> Blink
- Compile o código
- Envie o programa para o
Arduino

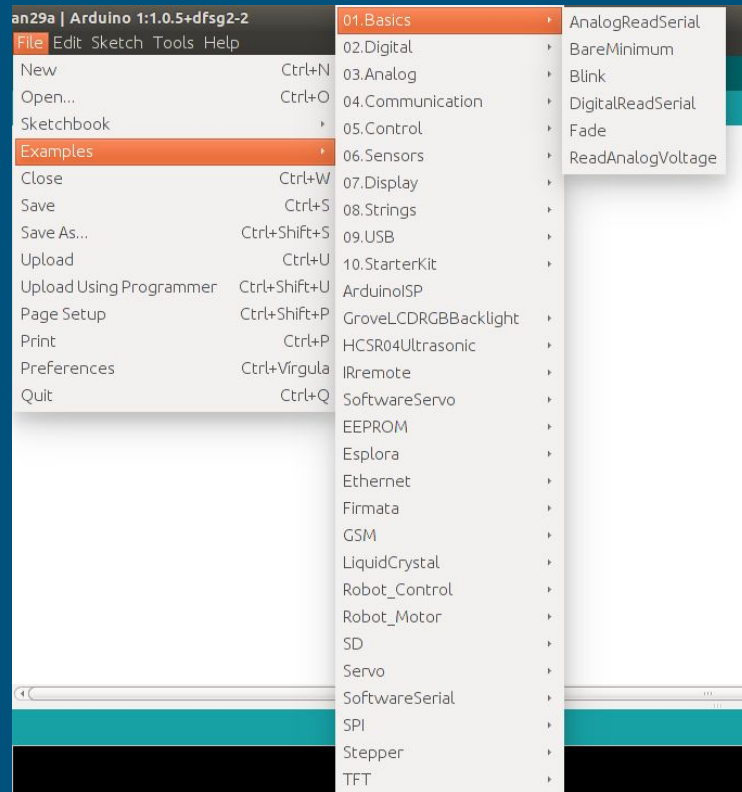


Atividades

- Modifique o projeto
 - Sugestões:
 - altere a frequência com que o LED pisca
 - faça dois LEDs piscarem juntos
 - faça dois LEDs piscarem alternadamente
- Utilize `analogWrite(led, <valor>)` no lugar de `digitalWrite(led, HIGH)`
 - <valor> é um número entre 0 e 255
- Faça o exemplo em File -> Examples -> Basics -> Fade
 - Projeto completo disponível em <https://www.arduino.cc/en/Tutorial/Fade>

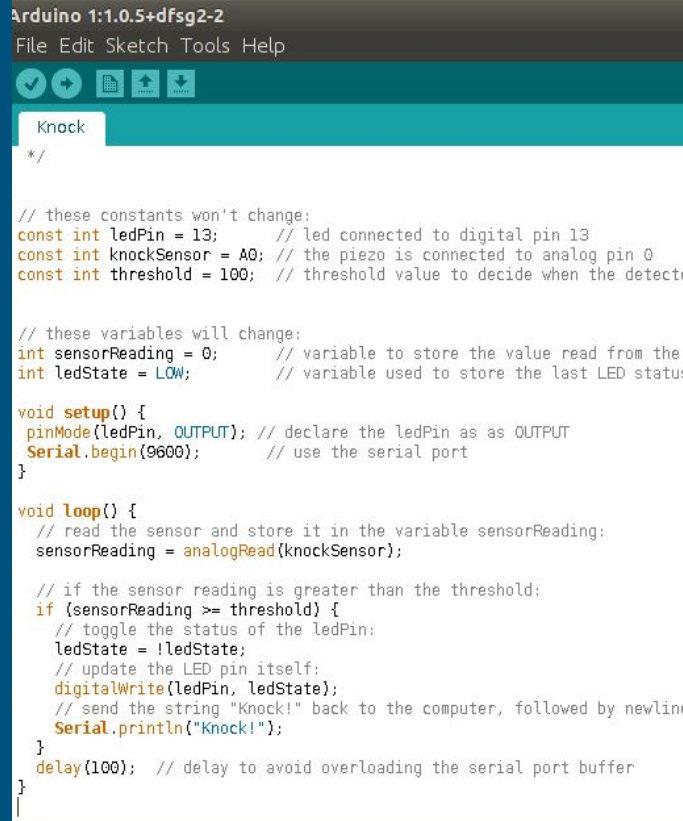
Milhares de exemplos

A IDE Arduino vem com vários exemplos, dos mais simples (piscar um LED) aos mais elaborados envolvendo sensores, motores, etc..



Linguagem de programação

- Linguagem baseada em C/C++
- Estrutura simples
 - `setup()`
 - Executa apenas uma vez logo no início
 - `loop()`
 - Executa ciclicamente após o `setup()`



The screenshot shows the Arduino IDE interface with a sketch named "Knock". The code is written in C++ and is designed to control an LED based on a piezo sensor input. The code includes comments explaining the constants and variables used. The `setup()` function initializes the LED pin as an output and starts the serial port. The `loop()` function reads the sensor value, checks if it exceeds a threshold, and toggles the LED state accordingly, while also printing the state to the serial monitor.

```
Arduino 1:1.0.5+dfsg2-2
File Edit Sketch Tools Help

Knock
*/

// these constants won't change:
const int ledPin = 13;    // led connected to digital pin 13
const int knockSensor = A0; // the piezo is connected to analog pin 0
const int threshold = 100; // threshold value to decide when the detect

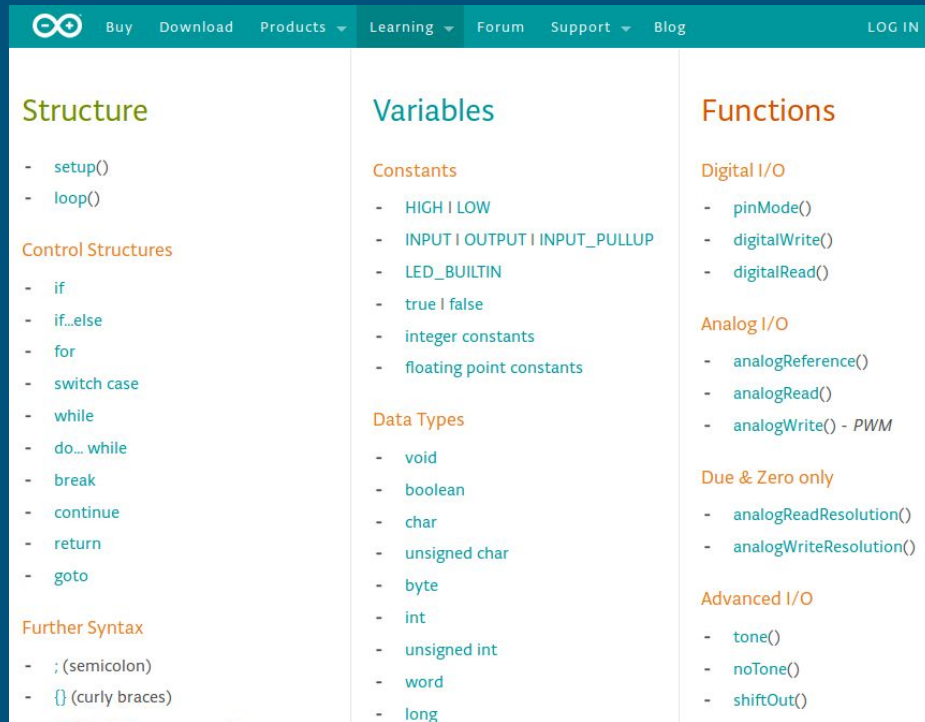
// these variables will change:
int sensorReading = 0;    // variable to store the value read from the
int ledState = LOW;       // variable used to store the last LED status

void setup() {
  pinMode(ledPin, OUTPUT); // declare the ledPin as as OUTPUT
  Serial.begin(9600);      // use the serial port
}

void loop() {
  // read the sensor and store it in the variable sensorReading:
  sensorReading = analogRead(knockSensor);

  // if the sensor reading is greater than the threshold:
  if (sensorReading >= threshold) {
    // toggle the status of the ledPin:
    ledState = !ledState;
    // update the LED pin itself:
    digitalWrite(ledPin, ledState);
    // send the string "Knock!" back to the computer, followed by newline
    Serial.println("Knock!");
  }
  delay(100); // delay to avoid overloading the serial port buffer
}
```

Referência da linguagem



The screenshot displays the Arduino Reference website's 'Structure' section. The navigation bar at the top includes links for 'Buy', 'Download', 'Products', 'Learning', 'Forum', 'Support', 'Blog', and a 'LOG IN' button. The 'Structure' section is organized into three main columns: 'Structure', 'Variables', and 'Functions'. The 'Structure' column lists basic functions like 'setup()' and 'loop()', followed by 'Control Structures' (if, if...else, for, switch case, while, do... while, break, continue, return, goto) and 'Further Syntax' (semicolon, curly braces). The 'Variables' column covers 'Constants' (HIGH | LOW, INPUT | OUTPUT | INPUT_PULLUP, LED_BUILTIN, true | false, integer constants, floating point constants) and 'Data Types' (void, boolean, char, unsigned char, byte, int, unsigned int, word, long). The 'Functions' column is divided into 'Digital I/O' (pinMode(), digitalWrite(), digitalRead()), 'Analog I/O' (analogReference(), analogRead(), analogWrite() - PWM), 'Due & Zero only' (analogReadResolution(), analogWriteResolution()), and 'Advanced I/O' (tone(), noTone(), shiftOut()).

Structure

- setup()
- loop()

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Further Syntax

- ; (semicolon)
- {} (curly braces)

Variables

Constants

- HIGH | LOW
- INPUT | OUTPUT | INPUT_PULLUP
- LED_BUILTIN
- true | false
- integer constants
- floating point constants

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long

Functions

Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - PWM

Due & Zero only

- analogReadResolution()
- analogWriteResolution()

Advanced I/O

- tone()
- noTone()
- shiftOut()

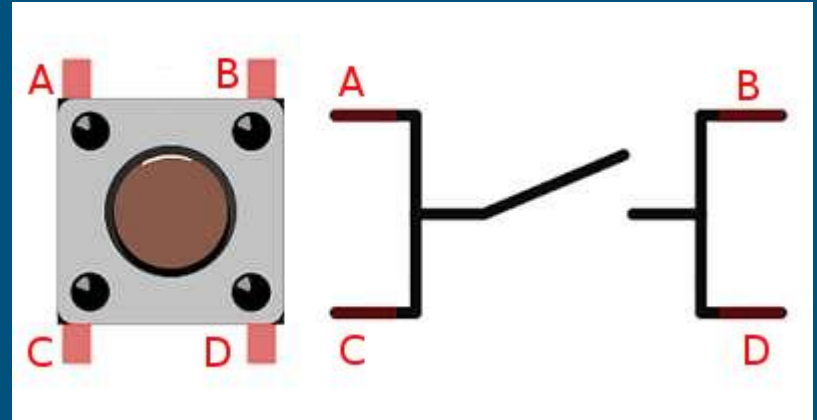
<https://www.arduino.cc/en/Reference/HomePage>

Recebendo e enviando sinais digitais

Objetivo: Ligar e desligar um LED de acordo com o estado de um botão.



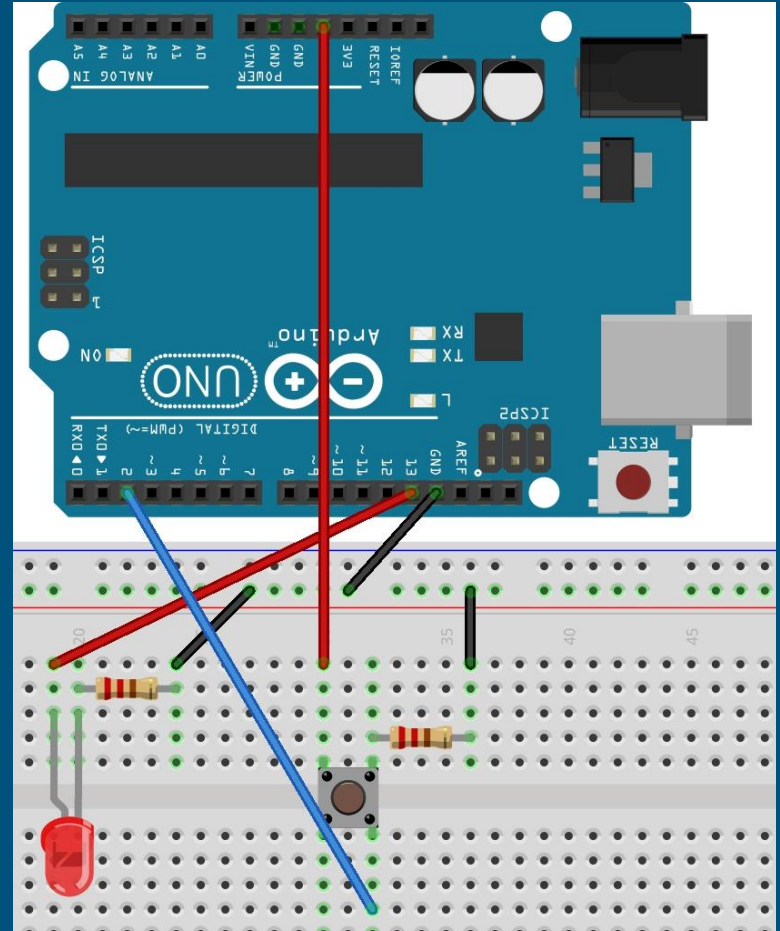
O botão / push button



Quando pressionado o botão conecta os contatos A e C aos contatos em B e D.

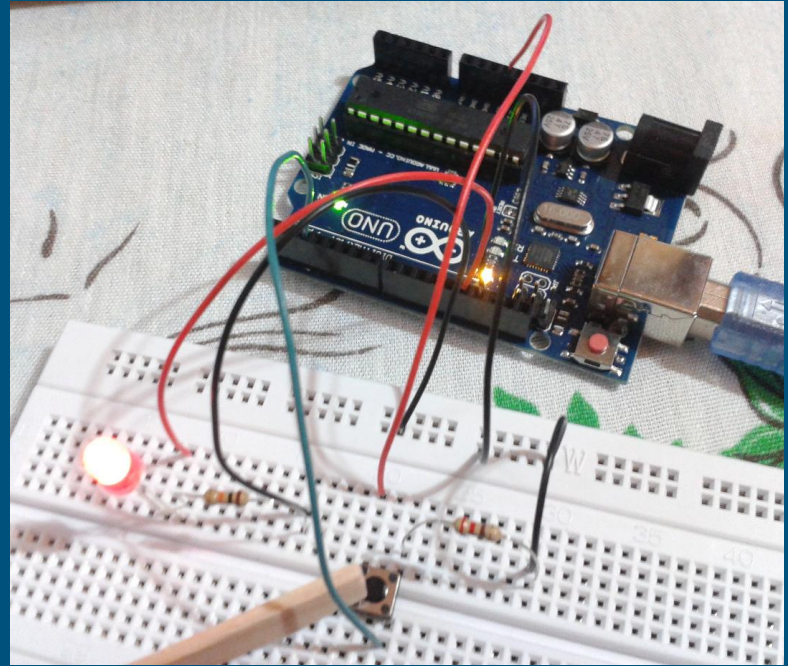
Projeto Botão

- Hardware necessário
 - Placa Arduino
 - Protoboard
 - 1 LED
 - 2 Resistores de 220 ohms
 - Fios para conectar os componentes
 - Um botão



Recebendo e enviando sinais digitais

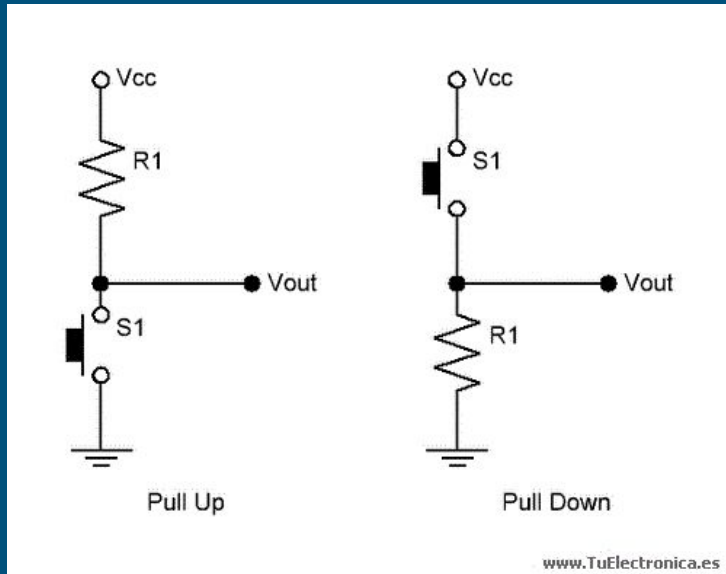
- Código para o projeto em: File -> Examples -> Digital -> Button
- Compile o código (opcional)
- Envie o programa para o Arduino



Atividades

- Mude o código para que o LED permaneça ligado e desligue quando o botão estiver pressionado
- Mude o código para que o LED mude de estado (ligado/desligado) ao pressionar do botão
 - Botão liga e desliga
- Mude o projeto para que o LED ligue e permaneça ligado ao pressionar de um botão e desligue e permaneça desligado ao pressionar de outro botão.
 - Botão liga e (outro) botão desliga

Pullup e pulldown



- Até agora usamos resistores pulldown
- As Arduinos têm resistores internos para uso como resistores pullup
- Acessados com `pinMode(<pino>, INPUT_PULLUP)` isso inverte o comportamento do pino de entrada
 - HIGH quando o sensor está desligado e LOW quando ligado

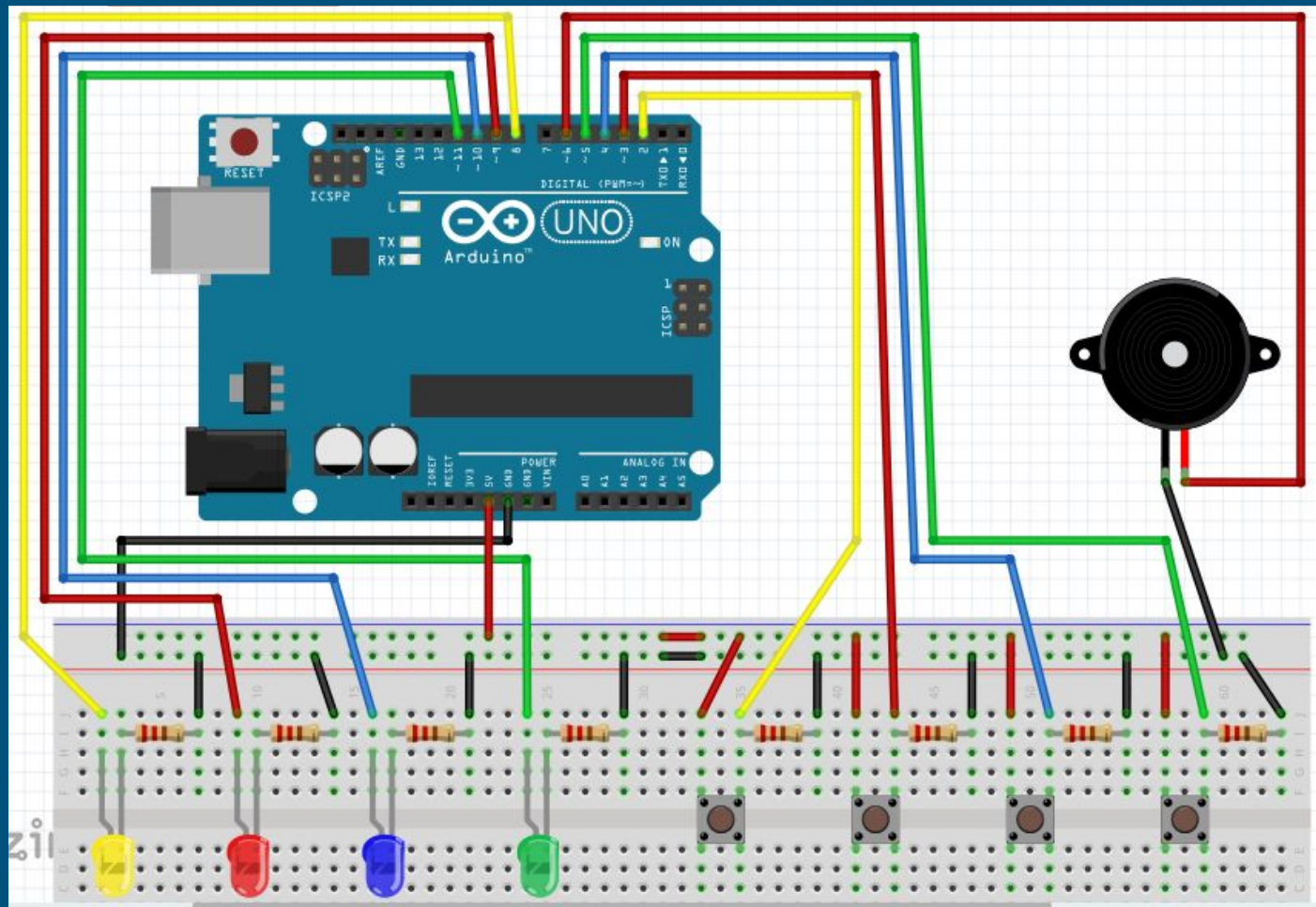
Atividades

- Estude o exemplo em <https://www.arduino.cc/en/Tutorial/InputPullupSerial>
- Faça a atividade anterior (botão liga e botão desliga) sem usar resistores
- Extra: estude o conteúdo em <https://www.arduino.cc/en/Tutorial/DigitalPins>

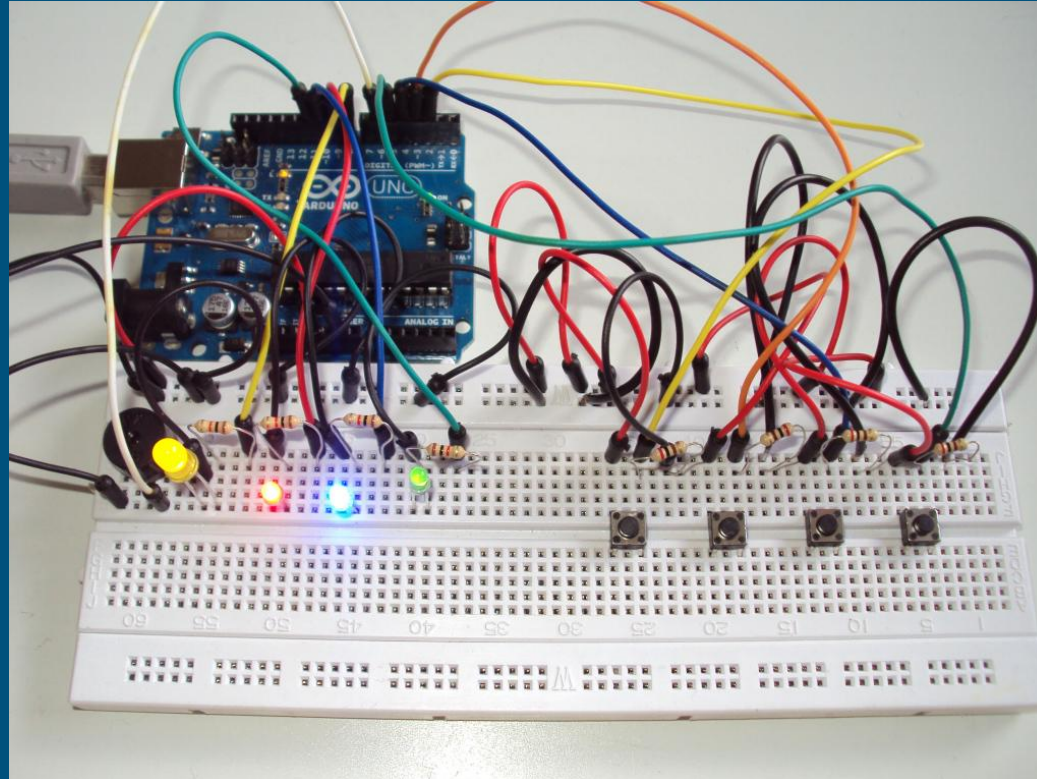
Projeto Genius (Jogo da memória)

- Hardware necessário
 - Placa Arduino
 - Protoboard
 - 4 LEDs
 - 8 Resistores de 220 ohms
 - 4 botões
 - 1 alto falante (speaker)
 - Fios para conectar os componentes

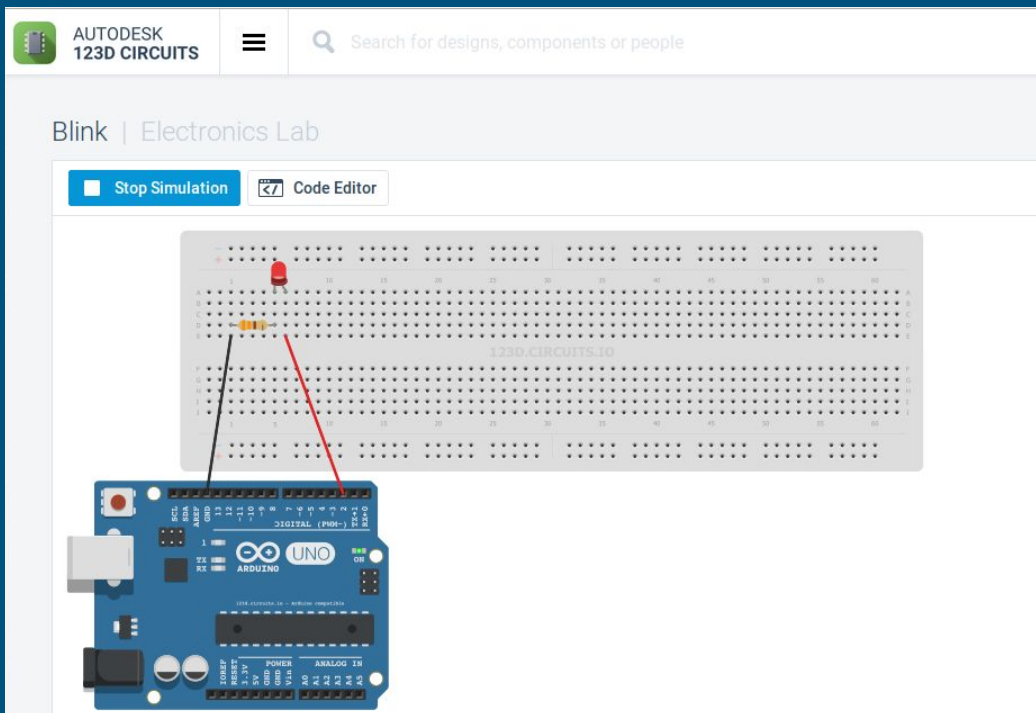




Código e melhor descrição do projeto em
<http://labdegaragem.com/profiles/blogs/arduino-genius-jogo-da-mem-ria>



E se eu não tiver uma placa Arduino?



Se você não tiver uma placa Arduino visite <https://123d.circuits.io/>

Grupo HardwareLivreUSP

Muito obrigado!

Curtam nossa pagina no facebook: www.facebook.com/Hardwarelivreusp

Participe do nosso grupo de Email: harduime@googlegroups.com