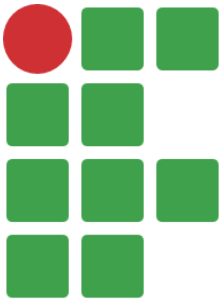


INSTITUTO FEDERAL
Triângulo Mineiro
Campus Uberlândia Centro
Licenciatura em Computação

APOSTILA DE

Robótica - Tinkercad

Prof. Walteno Martins Parreira Júnior

www.waltenomartins.com.br

waltenomartins@iftm.edu.br

2020

SUMÁRIO

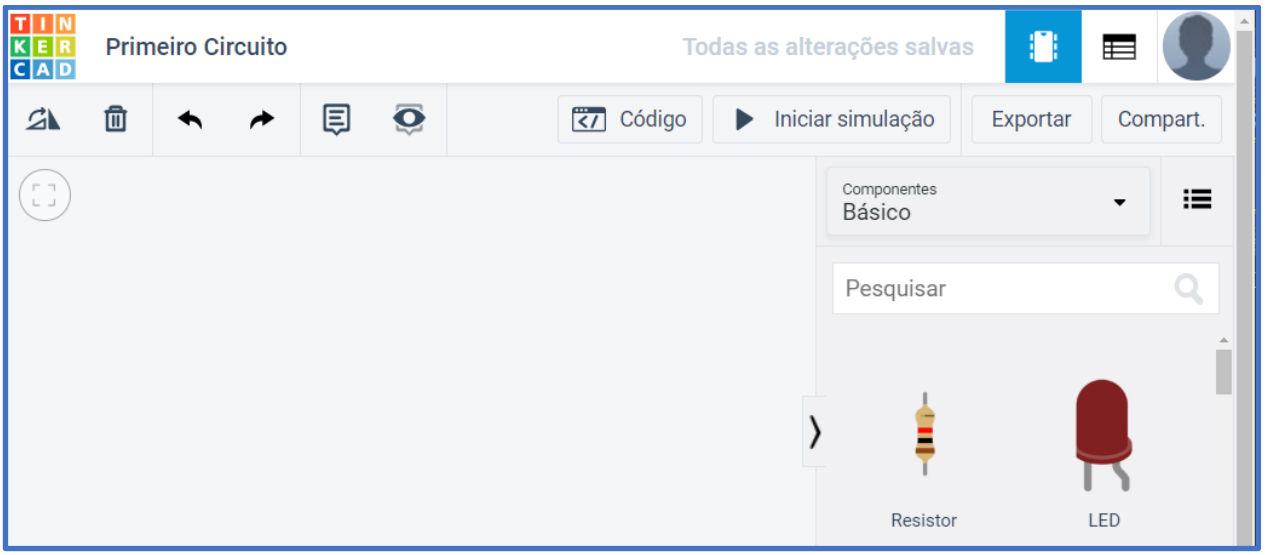
1. O Ambiente do TinkerCad

1.1. Tela Inicial

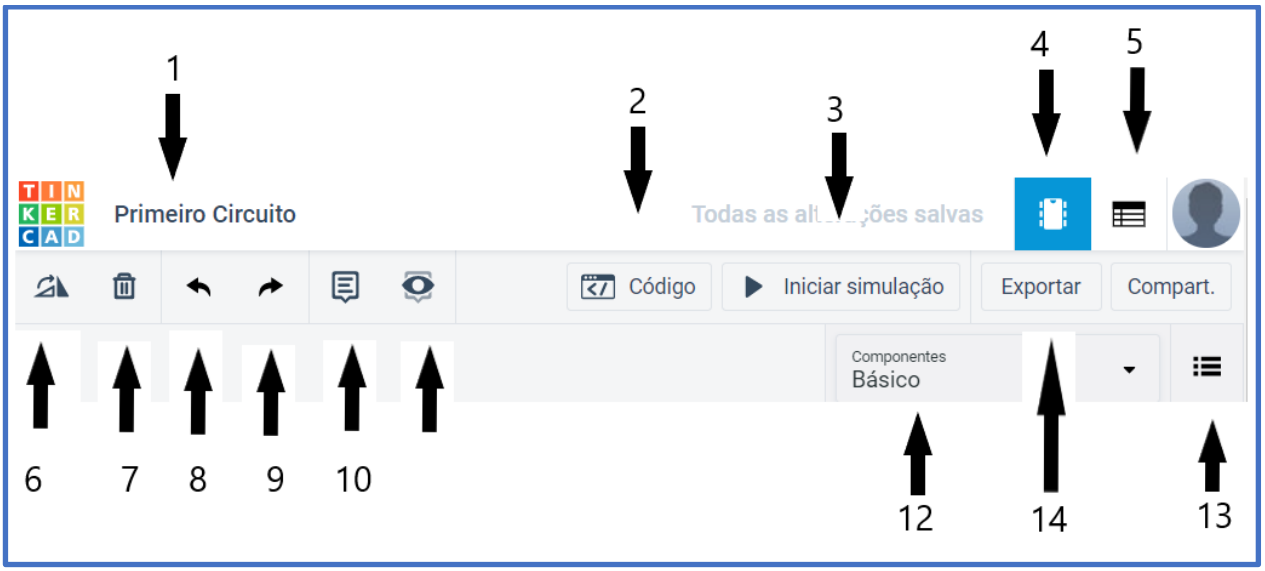
Tinkercad é uma ferramenta online da empresa Autodesk, criadora de software famosos como o Autocad, que permite criar, desenhar circuitos e modificar rapidamente qualquer design que tenha criado. Além de possuir a vantagem de ser uma ferramenta online e gratuita! E é claro não há riscos de queimar componentes eletrônicos.

Site: https://www.tinkercad.com/

O Tinkercad é simples, atraente e possui uma interface gráfica fluída e fácil de ser manipulada. A partir da tela inicial, podemos rotacionar o projeto, excluir elemento que não esteja sendo usando, centralizar a tela e fazer simulações.

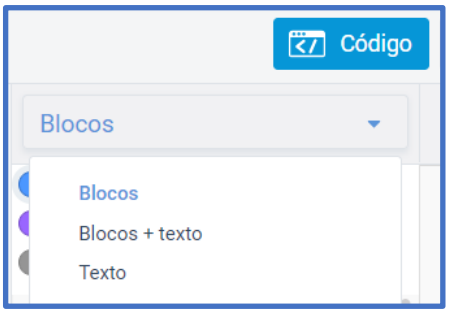


1.2. Ícones do Menu



1 – Titulo do projeto. Inicialmente o Tinkercad propõe um título para o projeto, que pode ser alterado clicando sobre ele e digitando o novo nome.

2 – Editor de Código. Esse botão abre o editor para digitar o código em C que será usado no arduino.



Há três possibilidades de criar o código, somente utilizando Blocos, Blocos e textos e somente textos. A troca de opção elimina o código construído.

Quando escolhido a opção Blocos e textos, a codificação é realizada utilizando os blocos de programação. Explicada no próximo capitulo.

3 – Iniciar Simulação. Após as edições no programa, ou no circuito e para ver o resultado, clique no botão *Iniciar Simulação* e uma emulação do projeto será iniciada. Durante a simulação não pode adicionar componentes e nem editar o código.

4 – Vista do Circuito. É o padrão de exibição da área de trabalho padrão.

Para ampliar ou reduzir a imagem do circuito, usar o mouse, girando o botão circular.

Para mover o circuito, clicar na parte interna, sem selecionar qualquer componente e arrastar. Para mover um componente é só selecionar e arrastar com o mouse.

5 – Lista de Componentes. Apresenta a lista dos componentes incluído no projeto desenvolvido.

6 – Girar. Gira o elemento selecionado em 30 graus para a direita.

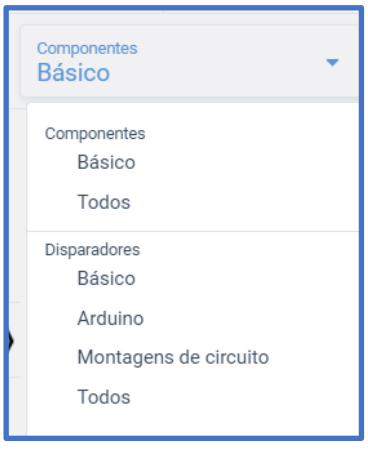
7 – Excluir. Elimina do projeto o item selecionado.

8 – Desfazer. Desfaz a última ação.

9 – Refazer. Refaz se necessário.

10 – Anotação. Permite fazer uma anotação no projeto.

12 – Componentes. Normalmente está selecionado a opção Básico, mas há outras opções que são apresentadas no menu. São os componentes disponíveis para a montagem do circuito.



Básico. Possui os componentes iniciais e mais básicos para um projeto.

Todos. É uma lista mais completa, um nível avançado de projetos.

Disparadores. São pequenos projetos prontos para utilizar. Estão distribuídos em 4 opções: básico, Arduino, montagem de circuito e todos (disponibiliza todos os projetos).

13 – Lista. Faz a apresentação dos componentes em lista, com uma pequena descrição ao lado de cada um.

14 – Exportar Circuito. É possível fazer a exportação do circuito. Mas é necessário instalar um EAGLE.BRD para o layout de placa de circuito.

1.3. Componentes

Estes são os componentes básico utilizados na maioria dos projetos.

a) **Resistor.** Restringe o fluxo de eletricidade em um circuito, reduzindo a voltagem no circuito. Pode-se definir a sua resistência configurando o valor na janela.

b) **LED.** É um condutor de energia elétrica, que quando energizado, emite luz visível a olho nu. Operam com tensões elétricas entre 1,5 V e 3,3 V. Pode-se definir a sua cor na janela.

c) **Botão (Pushbotton).** É uma chave que fecha um circuito quando é pressionada. Pode ser manipulado durante a simulação.

d) **Potenciômetro.** É um tipo de resistor que muda a resistência quando a chave é girada. Na simulação é possível simular o movimento.

e) **Capacitor.** Armazena e libera energia elétrica em um circuito. A função básica do capacitor é a de armazenar cargas elétricas em seu interior. Durante as descargas, os capacitores podem fornecer grandes quantidades de carga elétrica para um circuito.

f) **Interruptor deslizante**. Faz a interrupção ou conexão em um circuito. Tem duas posições: aberto ou fechado.

g) **Bateria 9V**. É utilizada como uma fonte de energia elétrica para o circuito quando há necessidade de maior potência.

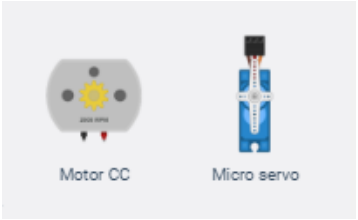
h) **Bateria 3V**. É utilizada como uma fonte de energia elétrica para o circuito quando há necessidade de pouca potência.

i) **Bateria 1,5V**. É a popular pilha, utilizada como uma fonte de energia elétrica para o circuito quando há necessidade de pouca potência.

j) **Placa de Ensaio Pequena (Protoboard)**. É uma placa com orifícios e conexões condutoras utilizada para a montagem de protótipos e projetos em estado inicial. Esta possui 30 linhas e 10 colunas separados por uma divisão. Cada conjunto com 5 orifícios formam um barramento.

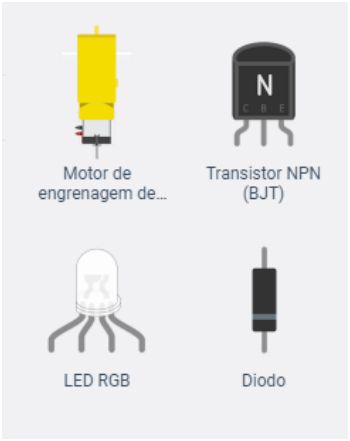
k) **Arduino Uno R3**. É uma plataforma de prototipagem eletrônica de hardware livre e de placa única, projetada com um microcontrolador. Placa programável.

l) **Motor de Vibração.** É um motor que vibra quando acionado.



m) **Motor CC.** São máquinas de corrente contínua (MCC). Funcionam tanto como motores quanto geradores de energia elétrica. São acionados por uma fonte de corrente contínua.

n) **Micro Servo**. É um atuador eletromecânico utilizado para posicionar e manter um objeto em uma determinada posição. É controlado através de um microcontrolador.

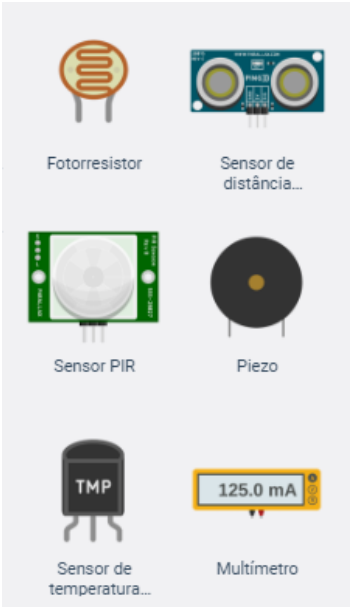


o) **Motor de Engrenagem**. É utilizado com frequência para o acionamento de rodas de robôs.

p) **Transistor NPN(BJT).** É utilizado para amplificar ou trocar o sinal eletrônico. Podem ser ajustados apenas para interromper uma corrente elétrica (fechado) ou para deixá-la passar integralmente (aberto), em uma função idêntica aos interruptores comuns.

q) **LED RGB.** Tipo de LED que combina vermelho, azul e verde para produzir qualquer cor. Tem 4 conectores, o primeiro da esquerda é o vermelho, o 2º é o catodo (-), o 3º é o azul e o 4º é o verde (primeiro da direita).

r) **Diodo.** Permite o fluxo de energia elétrica em uma única direção.



s) **Fotorresistor.** Sensor cuja resistência muda segundo o volume de luz detectado. Na simulação pode ser alterada a luminosidade.

t) **Sensor de Distância Ultrassônico**. Utiliza ondas sonoras para determinar a distancia até um objeto. Deve-se lembrar que o tempo determinado é dobrado pois é lido no seu retorno. Pode ser manipulado durante a simulação.

u) **Sensor PIR.** É usado para detectar movimento à sua frente, utiliza infravermelho. Pode ser manipulado durante a simulação.

v) **Piezo.** Tipo de buzzer que emite ruído em várias frequências.

w) **Sensor de Temperatura**. Emite voltagens diferentes considerando a temperatura detectada. Pode ser manipulado durante a simulação.

x) **Multímetro**. Ferramenta para medir voltagem, corrente e resistência no circuito.

2. Programação

2.1. Texto

Pode-se programar o ambiente de três formas, somente utilizando Blocos, Blocos e textos e somente textos. A troca de opção elimina o código construído. Quando escolhido a opção Blocos e textos, a codificação é realizada utilizando os blocos de programação. Explicada no próximo capitulo.

Os programas para o Arduino são implementados tendo como referência a linguagem C++. Preservando sua sintaxe clássica na declaração de variáveis, nos operadores, nos ponteiros, nos vetores, nas estruturas e em muitas outras características da linguagem.

Elas podem ser divididas em três partes principais: As estruturas, os valores (variáveis e constantes) e as funções.

2.1.1. As estruturas de referências:

Estruturas de controle: if, else, break, ...

Sintaxe básica: #define, #include, ; , ...

Operadores aritméticos e de comparação: +, -, =, ==, !=, ...

Operadores booleanos: &&, ||, !

Acesso a ponteiros: *, &

Operadores compostos: ++, {, +=, ...

Operadores de bits: |, ^ , , ...

2.1.2. Os valores de referências:

Tipos de dados: byte, array, int , char , ...

Conversões: char(), byte(), int(), ...

Variável de escopo e de qualificação: variable scope, static, volatile, ...

Utilitários: sizeof() - diz o tamanho da variável em bytes

2.1.3. Funções e constantes no arduino

O Arduino já provê várias funções e constantes para facilitar a programação:

setup()

loop()

Constantes: HIGH | LOW , INPUT | OUTPUT , ...

Bibliotecas: Serial, Servo, Tone, etc.

2.2. As Funções

Essas funções estão implementadas e disponíveis em bibliotecas que executam funcionalidades básicas do microcontrolador. Algumas não estão implementadas no Tinkercad, somente no Arduino.

2.2.1. Digital I/O

pinMode()

digitalWrite()

digitalRead()

2.2.2. Analógico I/O

analogReference()

analogRead()

analogWrite() - PWM

2.2.3. Avançado I/O

tone()

noTone()

shiftOut()

pulseIn()

2.2.4. Tempo

millis()

micros()

delay()

delayMicroseconds()

2.2.5. Matemática

min()

max()

abs()

constrain()

map()

pow()

sqrt()

2.2.6. Trigonométrica

sin()

cos()

tan()

2.2.7. Números aleatórios

randomSeed()

random()

2.2.8. bits e Bytes

lowByte()

highByte()

bitRead()

bitWrite()

bitSet()

bitClear()

bit()

2.2.9. Interrupções externas

attachInterrupt()

detachInterrupt()

2.2.10. Interrupções

interrupts()

noInterrupts()

2.2.11. Comunicação Serial

Serial.read()

SerialEvent()

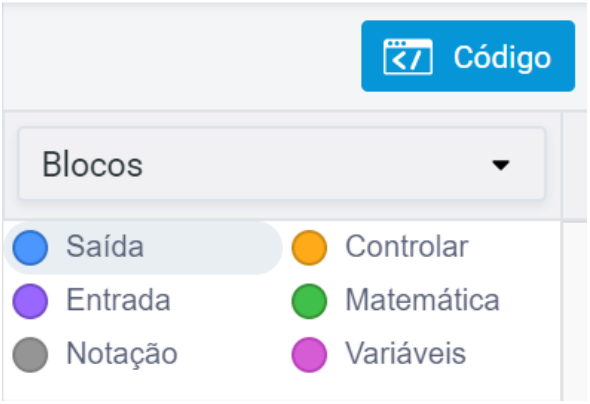
2.3. Blocos

Os blocos são peças em forma de quebra-cabeça que são usados para criar o código para o projeto. Os blocos conectam-se uns aos outros conforme o seu tipo e juntos formam um conjunto de instruções, de modo que cada bloco tenha sua própria forma e um conector de forma apropriada possa ser inserido para evitar erros de sintaxe.

Para construir um programa, deve-se arrastar blocos para o editor de código criando um conjunto de instruções que serão executadas no simulador ou no Arduino quando gravado nele.

Os blocos estão organizados em categorias, algumas básicas das linguagens de programação e outras específicas para a programação no Arduino.

Os blocos organizados no editor de código são automaticamente traduzidos para código fonte na linguagem de programação. O código traduzido aparece na aba “texto” sempre estruturado a partir das duas funções básicas: setup() e loop().



3. Exemplos de Programação

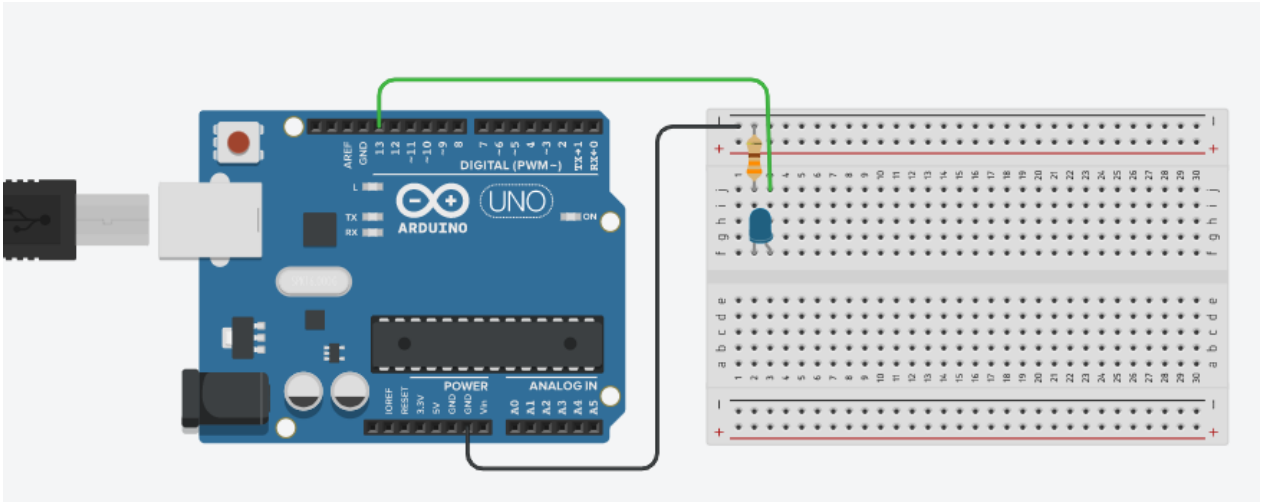
3.1. Primeira atividade:

Piscando um LED

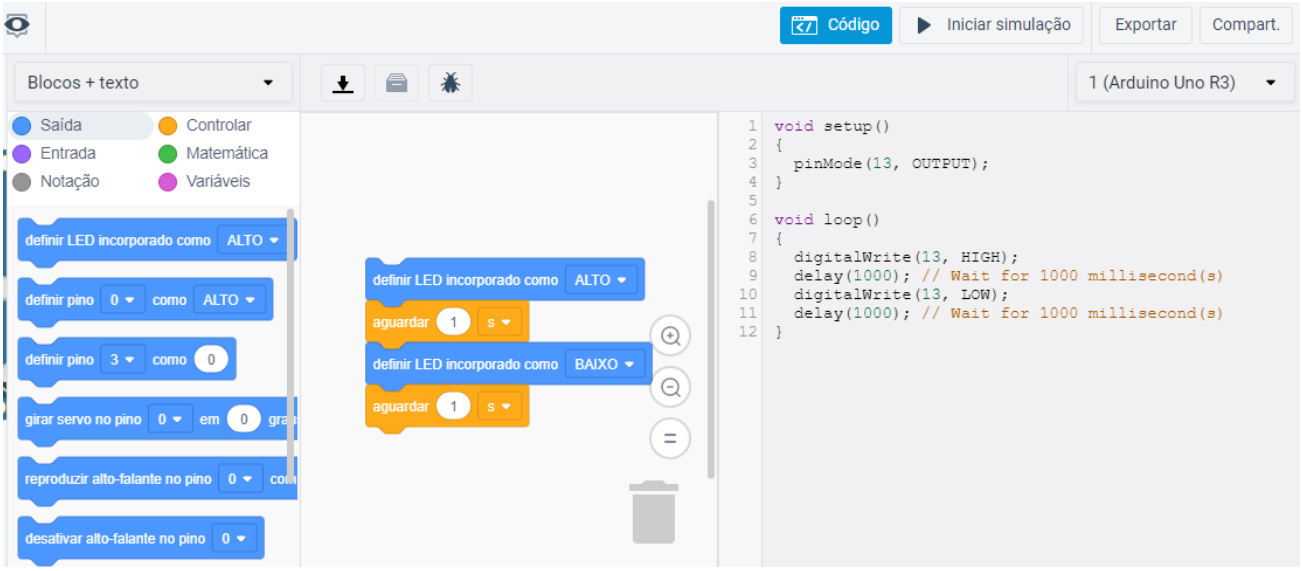
Material a ser utilizado:

1 Protoboard
1 Arduino Uno
1 Resistor 330 ohms
1 LED
2 Jumpers

Montagem:



Código gerado:



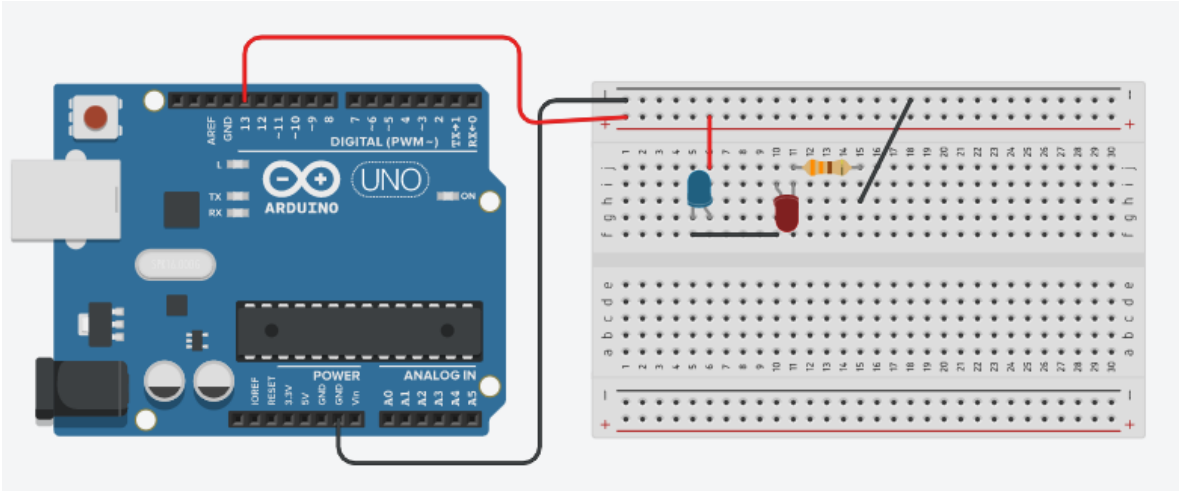
3.2. Segunda atividade:

Piscando dois LEDs

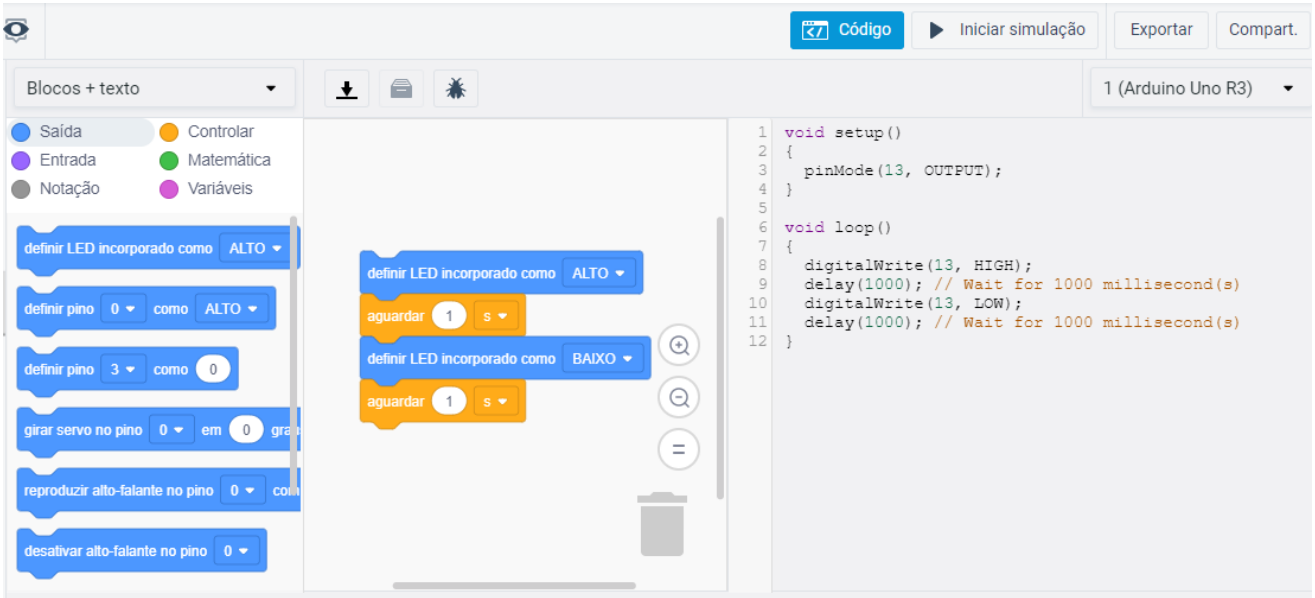
Material a ser utilizado:

1 Protoboard
1 Arduino Uno
1 Resistor 330 ohms
2 LED
5 Jumpers

Montagem:



Código gerado:



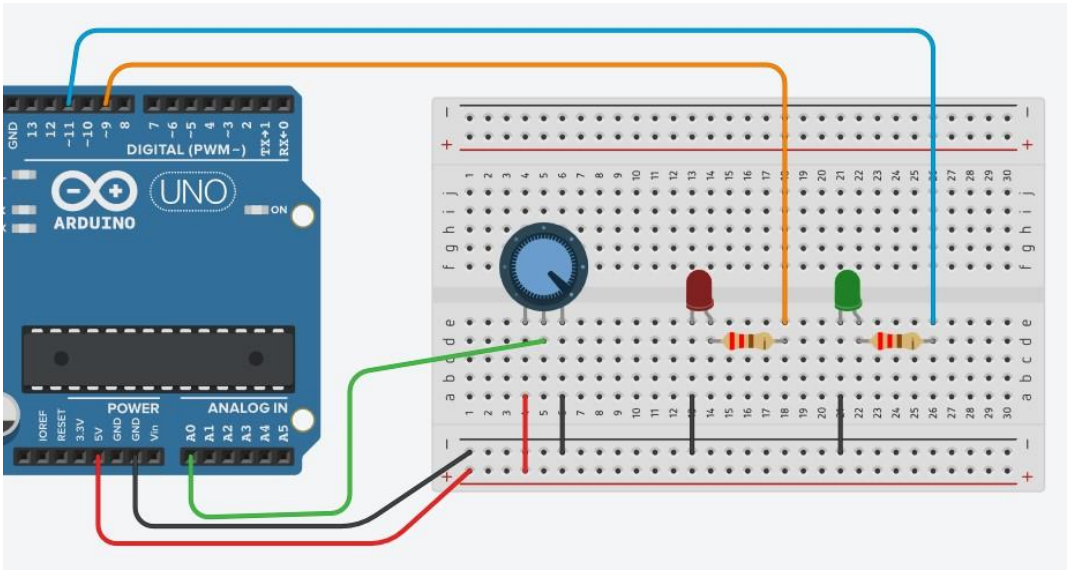
3.3. Terceira atividade:

Potenciômetro controlando os LEDs

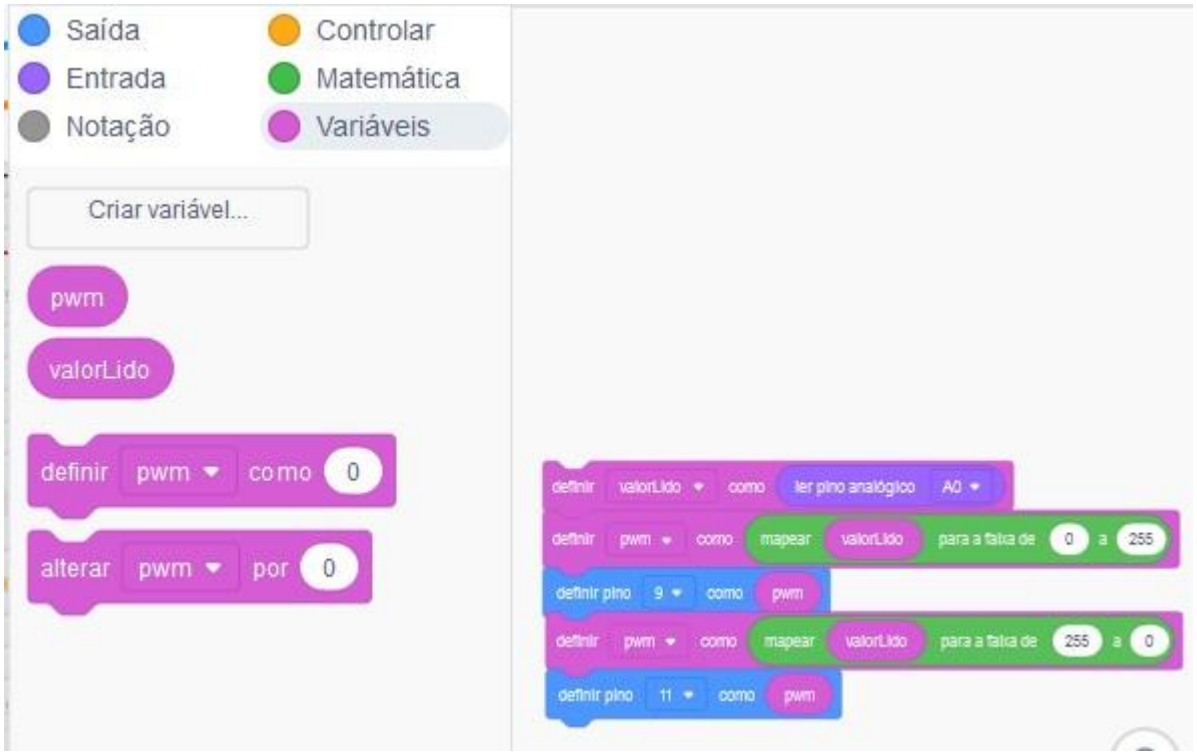
Material a ser utilizado:

1 Protoboard	2 Resistor 330 ohms
1 Arduino Uno	2 LED
1 Potenciômetro	7 Jumpers

Montagem:



Código gerado:



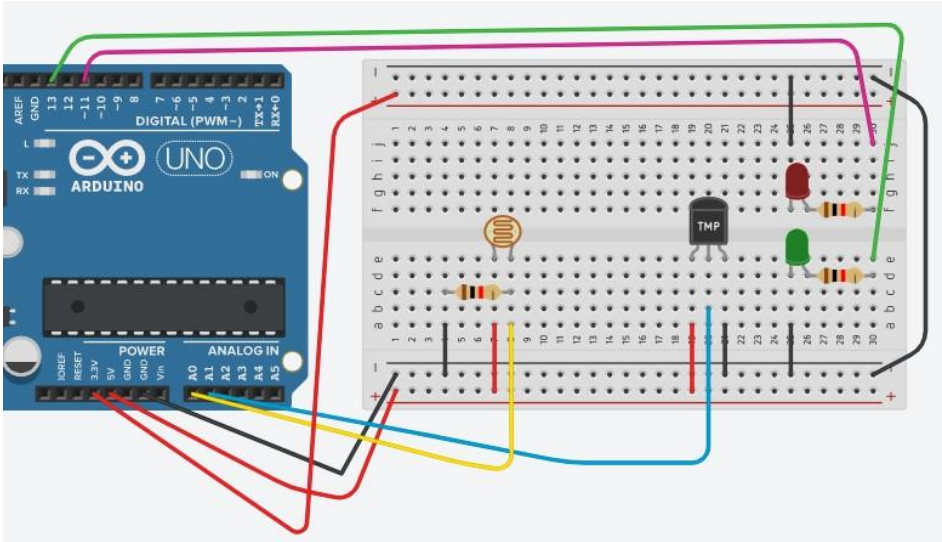
3.4. Quarta atividade:

Usando Sensores

Material a ser utilizado:

1 Protoboard
1 Arduino Uno
1 Sensor de Luminosidade
1 Sensor de Temperatura
3 Resistor 330 ohms
2 LED
14 Jumpers

Montagem:



Código gerado:

```
int lightSensor = A0;
int tempSensor = A1;

void setup() //É executado uma vez no início
da operacao
{
 pinMode(tempSensor, INPUT);
 pinMode(lightSensor, INPUT);
 Serial.begin(9600);
}

void loop()
{
 //imprimir uma linha para a leitura
 int lightReading = analogRead(lightSensor);
 Serial.print("Leitura Luz: ");
 Serial.println(lightReading);

 int tempReading = analogRead(tempSensor);
 // valor entre 0 e 750
 //Usando entrada de 5v
 float voltage = tempReading * 5.0;

// Dividindo por 1024
voltage /= 1024.0;

//Convertendo em intervalo de 10mv
float tempC = (voltage - 0.5) * 100;

// testa a temperatura e controla lampadas
if (tempC < 30.){
   digitalWrite(13,HIGH);    //led verde
} else {
   digitalWrite(13,LOW);
}
if (tempC > 50.){
   digitalWrite(11,HIGH);    //led vermelho
} else {

digitalWrite(11,LOW);
}

Serial.print(tempC);
Serial.println(" ºC");   //imprimindo o valor

  delay(100); //Usando intervalo de tempo
 }
```

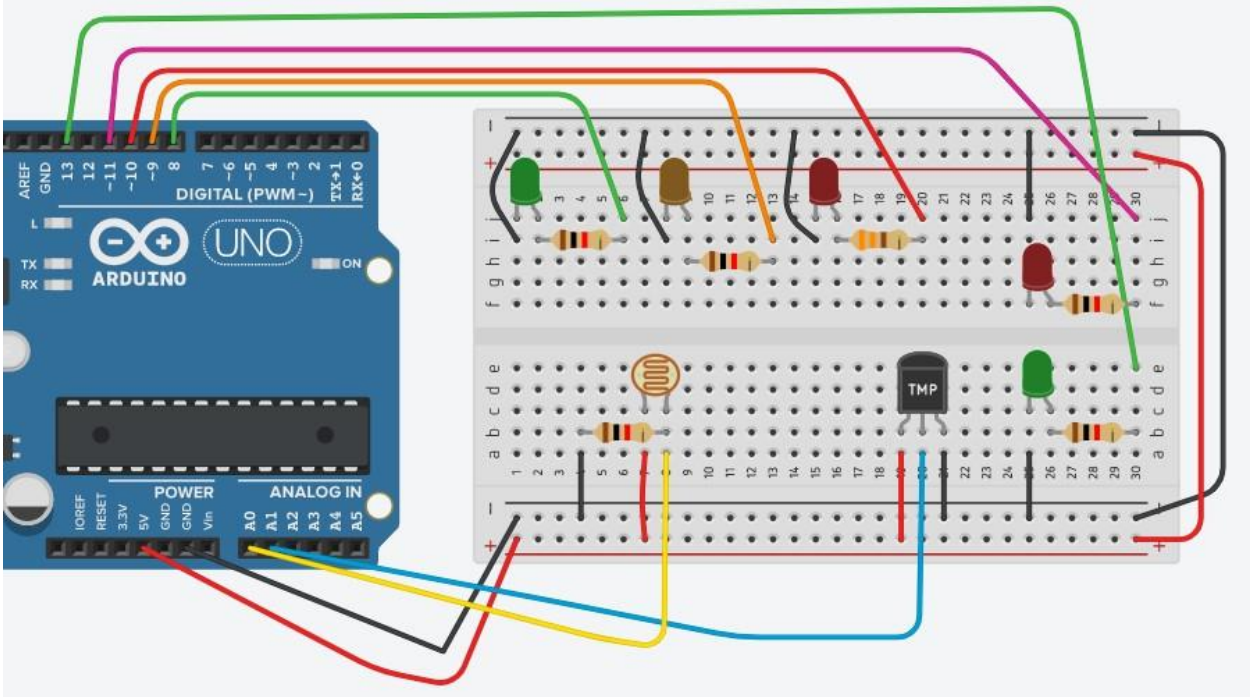
3.5. Quinta atividade:

Sensores controlando os LEDs

Material a ser utilizado:

1 Protoboard	6 Resistor 330 ohms
1 Arduino Uno	5 LED
1 Sensor de Luminosidade	20 Jumpers
1 Sensor de Temperatura	

Montagem:



Código gerado: