



Universidade Federal do Paraná
Campus Avançado de Jandaia do Sul
3ª Semana Integrada das Engenharias

Introdução ao Arduino

Ministrantes

Dr. Landir Saviniec

Dr. Marcelo Franco

Tópicos

- O que é o Arduino?
- Sensores, atuadores e controladores
- Componentes do arduino
- Como programar o arduino?
- Projeto 1: ligando e desligando leds
- Classes e objetos
- Projeto 2: criando uma classe para controlar leds
- Projeto 3: detectando obstáculos com sensores de distância
- Projeto 4: lendo sensores de umidade

O que é o arduino?

O arduino é um hardware contendo um microcontrolador programável para desenvolver projetos eletrônicos.



Sensores, atuadores e controladores

Sensores

Distância
Luminosidade
Temperatura
Umidade
Outros



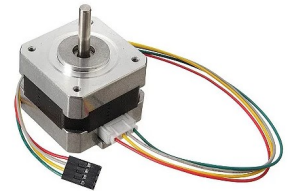
Ex: Sensor de distância

Ler o mundo físico

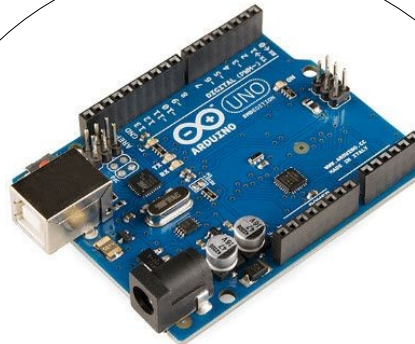
Executar ações

Atuadores

Motores
Lampadas
Válvulas
Outros

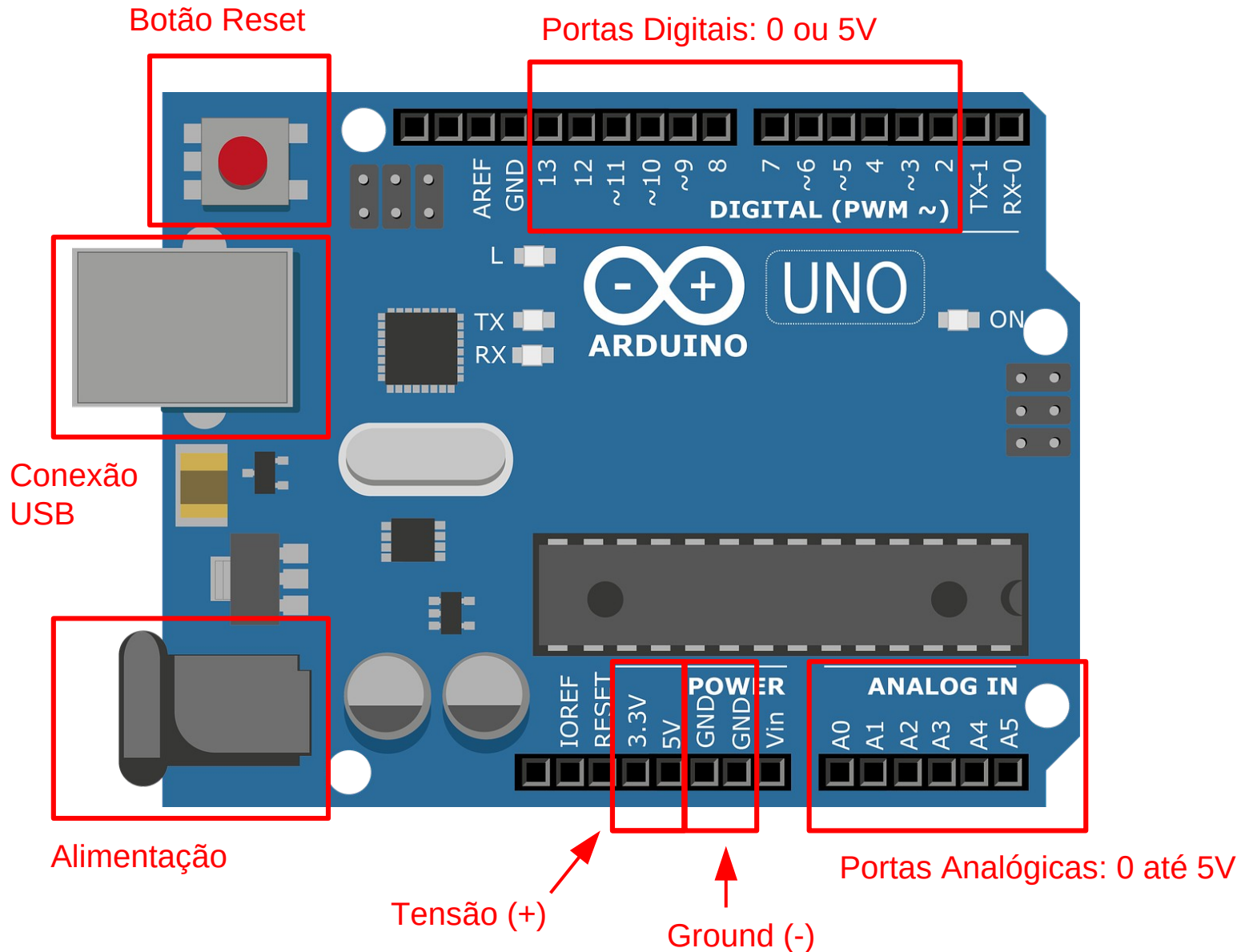


Ex: Motor de passo



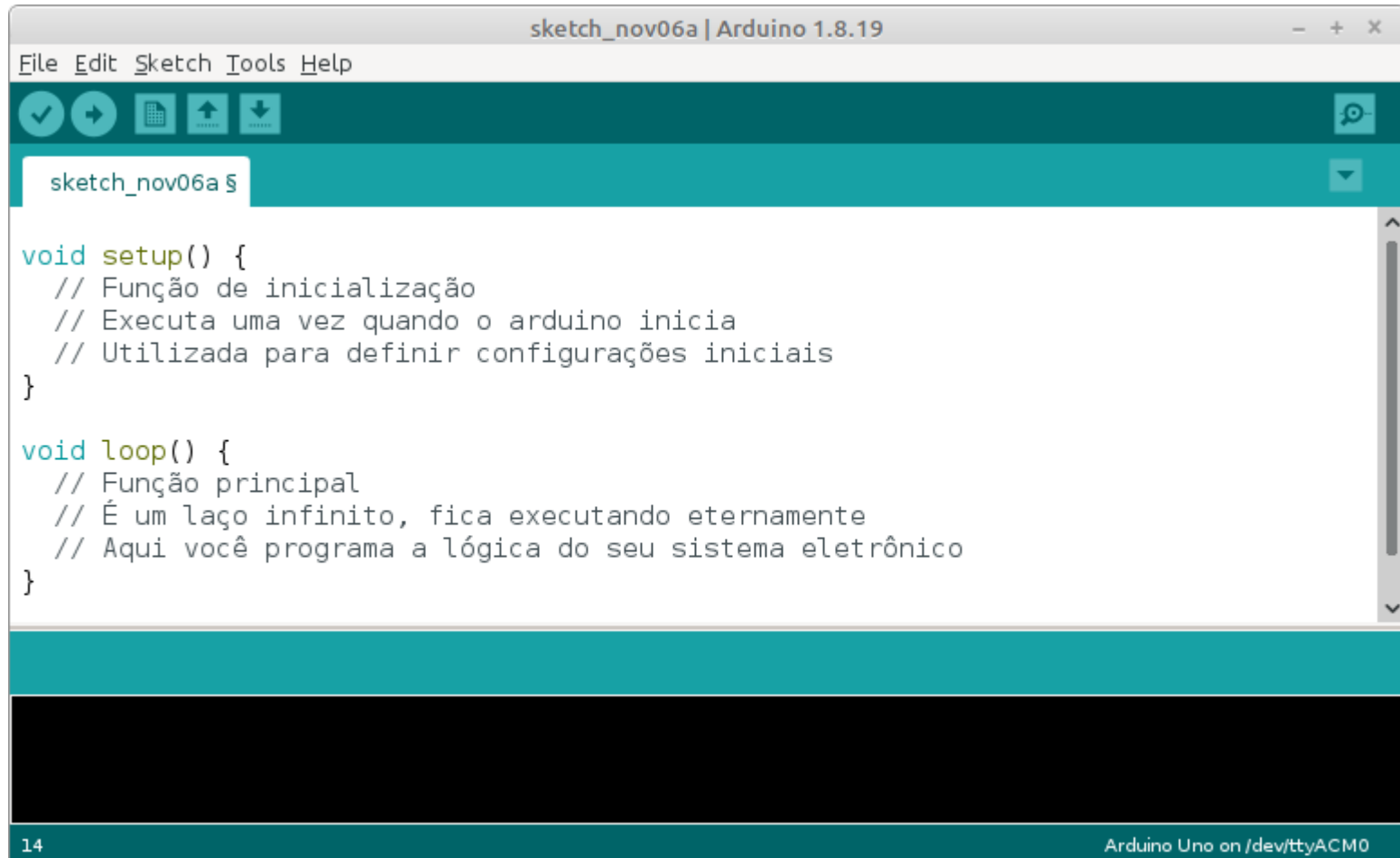
controladores

Componentes do arduino



Como programar o arduíno?

O arduino é programado via uma interface de programação (Arduino IDE). A linguagem utilizada é o C++.



The image shows a screenshot of the Arduino IDE (version 1.8.19) window. The title bar reads "sketch_nov06a | Arduino 1.8.19". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for checking, running, uploading, and downloading. The main editor area displays a C++ sketch for an Arduino Uno. The sketch includes comments in Portuguese explaining the functions of the `setup()` and `loop()` functions. The status bar at the bottom indicates "14" and "Arduino Uno on /dev/ttyACM0".

```
sketch_nov06a §

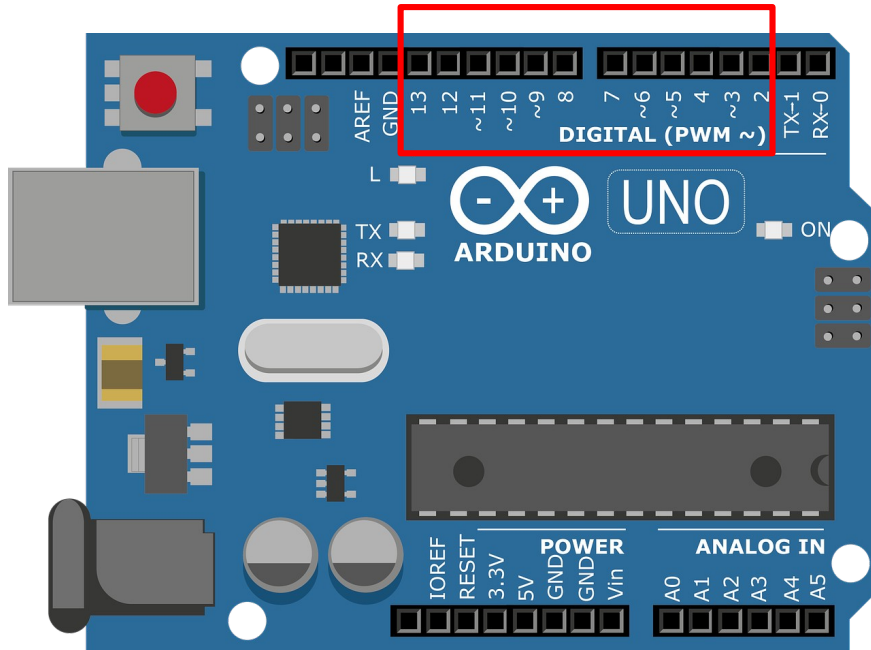
void setup() {
  // Função de inicialização
  // Executa uma vez quando o arduino inicia
  // Utilizada para definir configurações iniciais
}

void loop() {
  // Função principal
  // É um laço infinito, fica executando eternamente
  // Aqui você programa a lógica do seu sistema eletrônico
}
```

14 Arduino Uno on /dev/ttyACM0

Leitura e escrita de portas digitais

Portas Digitais: 0 ou 5V



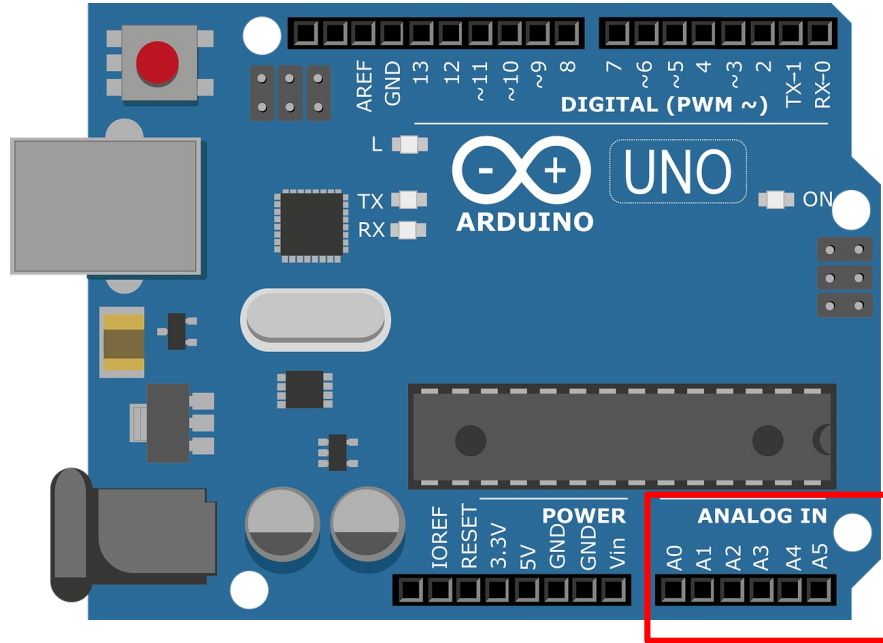
Configurando as portas:

- Configura a porta 12 para leitura:
`pinMode(12, INPUT);`
- Configura a porta 13 para escrita:
`pinMode(13, OUTPUT);`

Lendo e escrevendo nas portas:

- Lendo a porta 12:
`digitalRead(12);`
- Escrevendo na porta 13:
`digitalWrite(13, 1);` #ligando
`digitalWrite(13, 0);` #desligando

Leitura e escrita de portas analógicas



Portas Analógicas:
0 até 5V

Configurando as portas:

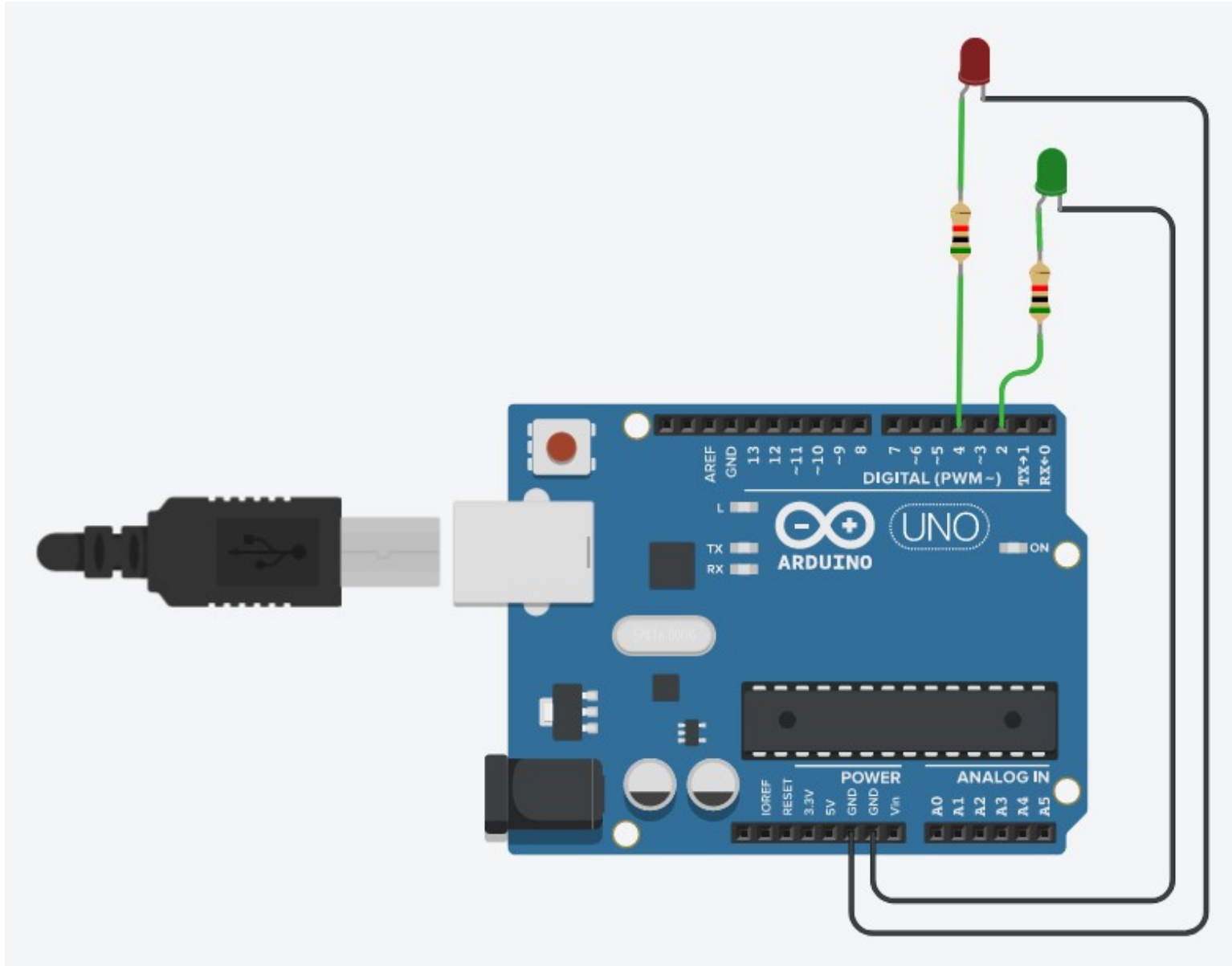
- Configura a porta A0 para leitura:
`pinMode(A0, INPUT);`
- Configura a porta A1 para escrita:
`pinMode(A1, OUTPUT);`

Lendo e escrevendo nas portas:

- Lendo a porta A0:
`analogRead(A0);`
- Escrevendo na porta A1:
`analogWrite(A1, valor);`

Projeto 1: ligando e desligando leds

Circuito do projeto 1



Código do projeto 1

projeto1

```
int pinoLedVerde = 2;
int pinoLedVermelho = 4;

void setup() {
  pinMode(pinoLedVerde, OUTPUT);
  pinMode(pinoLedVermelho, OUTPUT);
}

void loop() {
  digitalWrite(pinoLedVerde, 1);
  delay(100);
  digitalWrite(pinoLedVerde, 0);
  delay(1000);

  digitalWrite(pinoLedVermelho, 1);
  delay(100);
  digitalWrite(pinoLedVermelho, 0);
  delay(1000);
}
```

Classes e objetos

Classes e objetos

Classe: é um código reusável (um template).

Objeto: é uma instância da classe em tempo de execução.

Exemplo: O projeto arquitetônico de uma casa (classe) pode ser usado para construir várias casas (objetos).

Assim, o código de leitura de um sensor pode ser escrito uma única vez e usado para ler aquele tipo de sensor em vários projetos diferentes.

Para que serve: para organizar e reusar código. Bibliotecas de arduino são implementadas usando classes.

Estrutura de uma classe

```
class NomeDaClasse{  
  
    // Declaração de variáveis para guardar propriedades dos objetos  
  
    public:  
        NomeDaClasse(){  
            // Método construtor. Tem o mesmo nome da classe  
            // Executa uma vez quando o objeto é criado  
        }  
  
        void metodoA(){  
            //Função para executar alguma ação  
        }  
  
        int metodoB(){  
            // Uma classe pode ter várias funções  
            // Funções podem retornar valores ou não  
        }  
};
```

Projeto 2: criando uma classe para controlar leds

Classe “Led”

```
projeto2
class Led{
  int pinoDoLed;

  public:
    Led(int pino){
      pinMode(pino, OUTPUT);
      pinoDoLed = pino;
    }

    void ligar(){
      digitalWrite(pinoDoLed, 1);
    }

    void desligar(){
      digitalWrite(pinoDoLed, 0);
    }
};

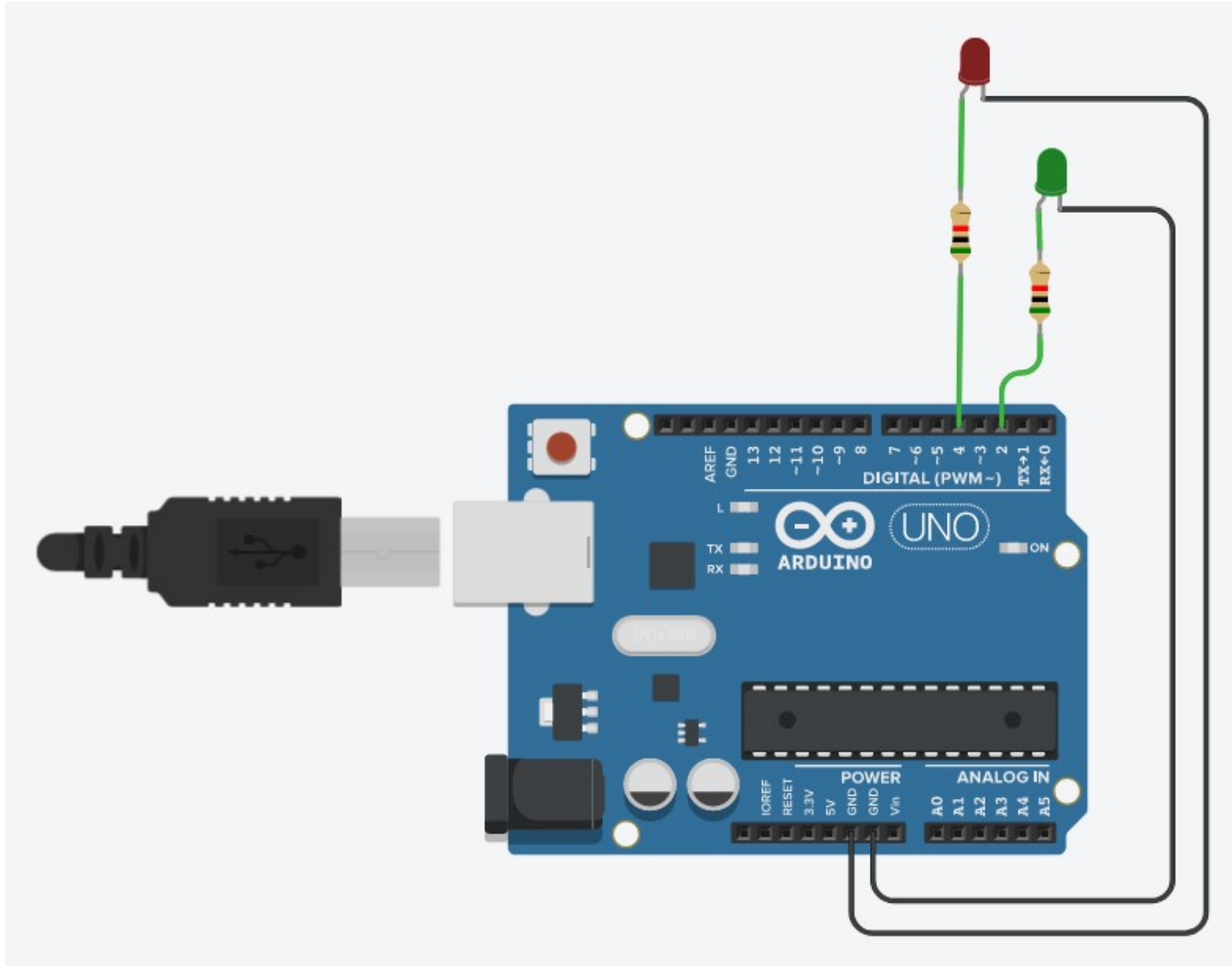
Led ledVerde(2);
Led ledVermelho(4);

void setup() {
}

void loop() {
  ledVerde.ligar();
  delay(100);
  ledVermelho.ligar();
  delay(1000);

  ledVerde.desligar();
  ledVermelho.desligar();
  delay(1000);
}
```


Circuito do projeto 2



Projeto 3: detectando obstáculos com sensores de distância

Descrição do projeto

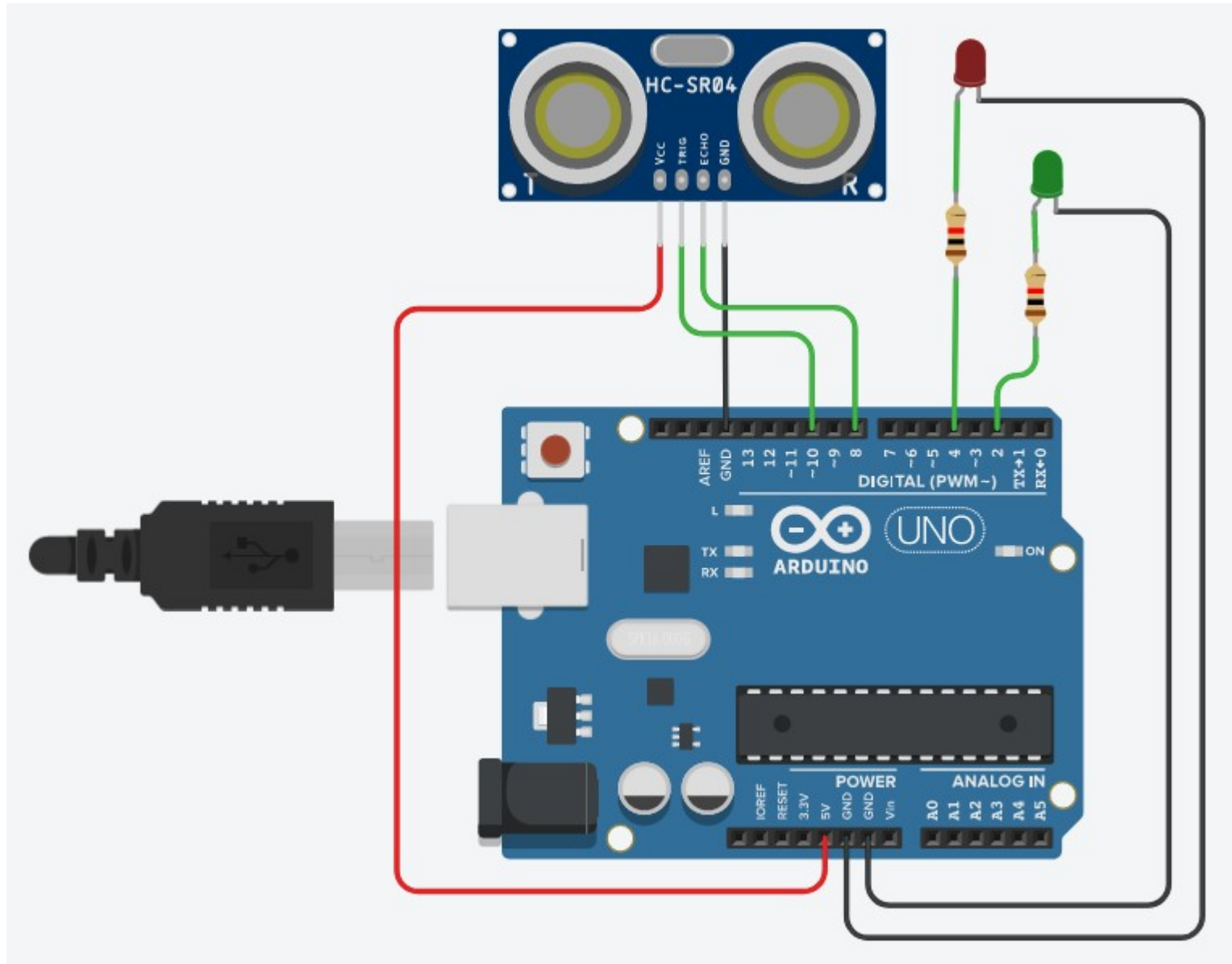
Usar um sensor de distância para detectar obstáculos. Enquanto nenhum obstáculo estiver próximo, um led verde permanecerá ligado. Quando um obstáculo estiver muito próximo (distância < 50 cm), o led verde desligará e um led vermelho será ligado, para indicar alerta.

Bibliotecas necessárias:

Utilizaremos a biblioteca Ultrasonic para ler o sensor.

Descompacte o arquivo Ultrasonic.zip e coloque a pasta Ultrasonic dentro da pasta LIBRARIES da IDE do Arduino.

Circuito do projeto 3



Código do projeto 3 (Parte 1)

```
projeto3
#include <Ultrasonic.h>
#define ECHO 8
#define TRIGGER 10

class Led{
    int pinoDoLed;

public:
    Led(int pino){
        pinMode(pino, OUTPUT);
        pinoDoLed = pino;
    }

    void ligar(){
        digitalWrite(pinoDoLed, 1);
    }

    void desligar(){
        digitalWrite(pinoDoLed, 0);
    }
};

Led ledVerde(2);
Led ledVermelho(4);
Ultrasonic sensor(TRIGGER, ECHO);

void setup() {
    //Habilita Comunicação Serial a uma taxa de 9600 bauds.
    Serial.begin(9600);
    ledVerde.ligar();
}
```

Código do projeto 3 (Parte 2)

```
void loop()
{
    double distancia = sensor.Ranging(CM); //retorna a distancia em centímetros
    Serial.print(distancia);
    Serial.println(" cm");

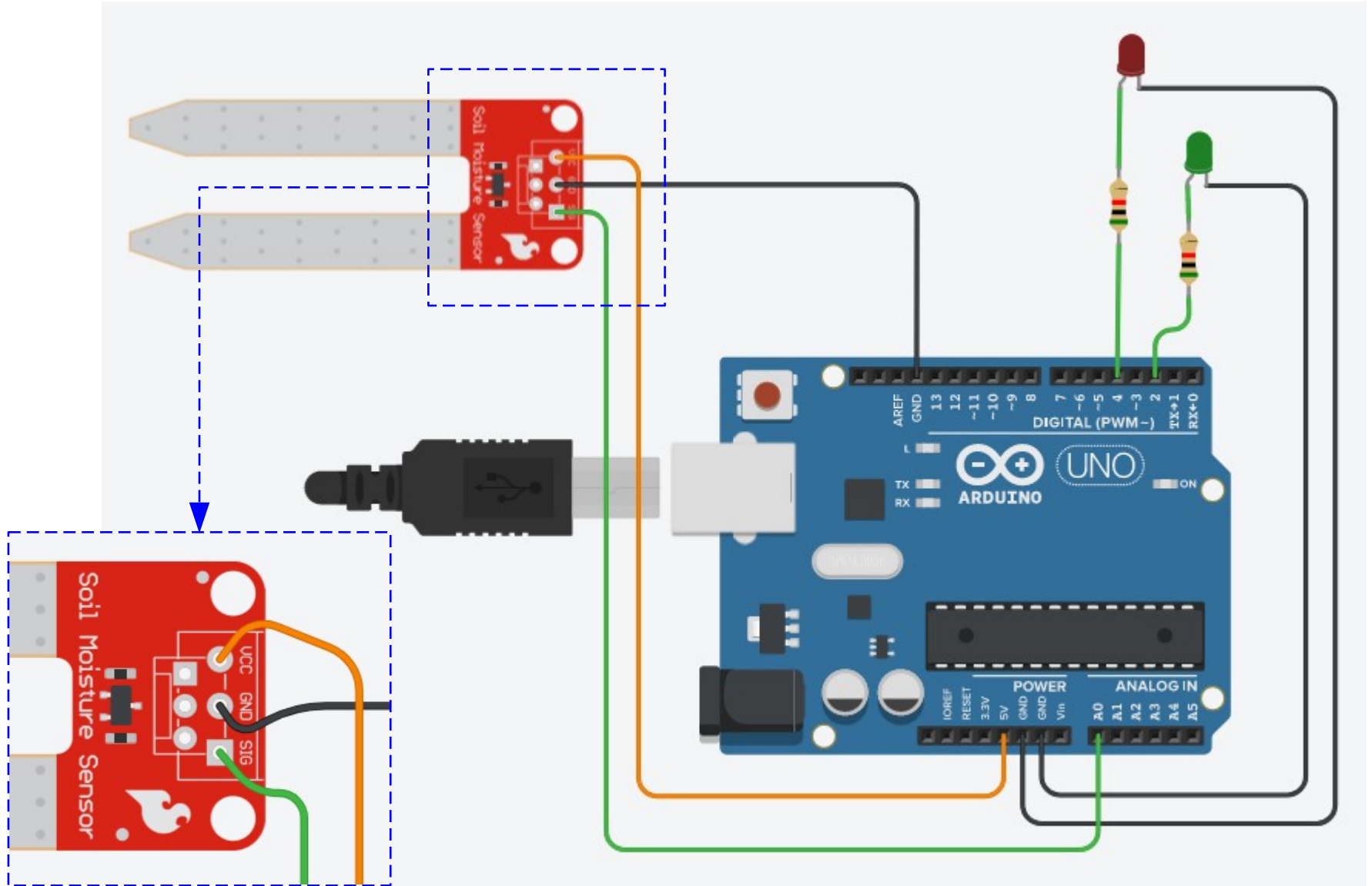
    if(distancia < 50){
        ledVermelho.ligar();
        ledVerde.desligar();
    }else{
        ledVerde.ligar();
        ledVermelho.desligar();
    }
    delay(100);
}
```

Projeto 4: lendo sensores de umidade

Descrição do projeto

Usar um sensor de umidade para detectar quando o solo está seco ou molhado. Quando a umidade for maior ou igual a 40%, um led verde permanecerá ligado. Quando a umidade estiver abaixo de 40%, o led verde desligará e um led vermelho será ligado, indicando que o solo está seco.

Circuito do projeto 4



Código do projeto 4 (Parte 1)

```
projeto4
class SensorUmidade{
    int pinoDoSinal;

    public:
        SensorUmidade(int pino){
            pinMode(pino, INPUT);
            pinoDoSinal = pino;
        }

        int getUmidade(){
            int umidade = analogRead(pinoDoSinal);
            return map(umidade, 0, 1023, 100, 0);
        }
};

class Led{
    int pinoDoLed;

    public:
        Led(int pino){
            pinMode(pino, OUTPUT);
            pinoDoLed = pino;
        }

        void ligar(){
            digitalWrite(pinoDoLed, 1);
        }

        void desligar(){
            digitalWrite(pinoDoLed, 0);
        }
};
```

Código do projeto 4 (Parte 2)

```
SensorUmididade sensor(A0);
Led ledVerde(2);
Led ledVermelho(4);

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

void loop() {

    int umidade = sensor.getUmididade();
    Serial.print("Umidade: ");
    Serial.print(umidade);
    Serial.println("%");

    if(umidade < 40){
        ledVermelho.ligar();
        ledVerde.desligar();
    }else{
        ledVerde.ligar();
        ledVermelho.desligar();
    }

    delay(5000);
}
```