Introdução ao Arduino

Sistemas embarcados

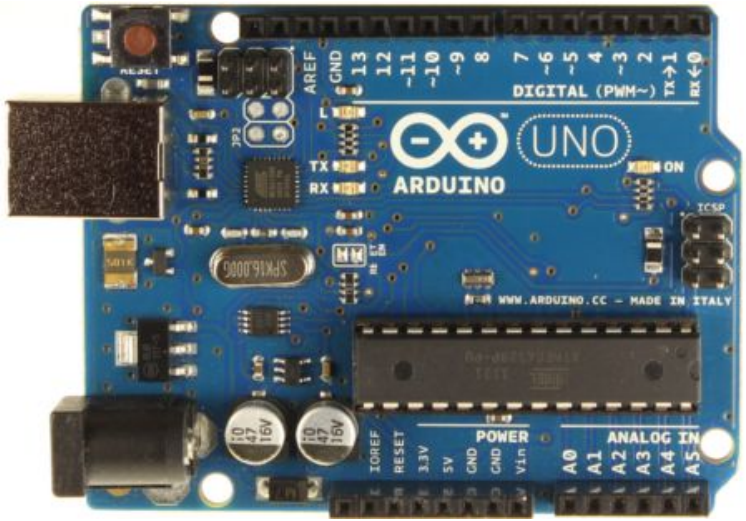
Prof. Allan Rodrigo Leite

Plataforma Arduino

- Plataforma de prototipação aberta
  - Fornece um conjunto de interfaces bem definido
  - Acesso aos componentes conectados ao circuito integrado
  - Dispõe de GPIO (*General Purpose Input/Output*)
  - Utiliza como base linguagem de programação C
    - Pode ser utilizada outras linguagens como Lua, Javascript, etc.

- Vantagens da plataforma
  - Baixo custo para prototipação e impulsão do movimento *maker*
  - Facilidade para construção de sistemas embarcados
    - Requer pouco conhecimento sobre circuitos digitais e programação

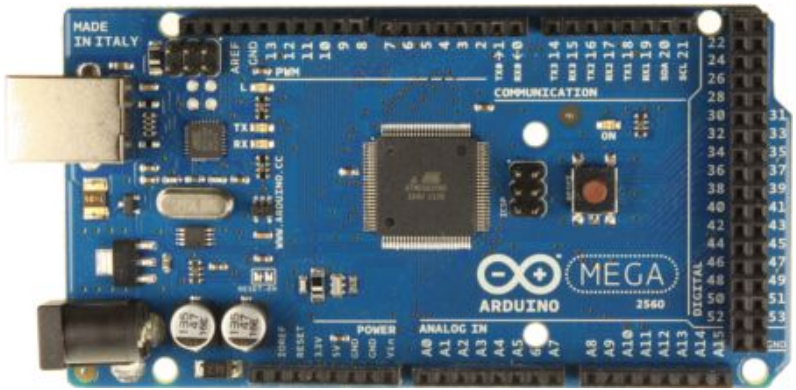
Plataforma Arduino



Arduíno UNO

Microcontrolador:	ATmega328
Pinos E/S digitais:	14 (6 com saída PWM)
Pinos entrada analógicos:	6
Memória Flash:	32 KB
SRAM:	2 KB
EEPROM:	1 KB
Velocidade clock:	16MHz

Plataforma Arduino



Arduíno MEGA

Microcontrolador:	ATmega2560
Pinos E/S digitais:	54 (15 com saída PWM)
Pinos entrada analógicos:	16
Memória Flash:	256 KB
SRAM:	8 KB
EEPROM:	4 KB
Velocidade clock:	16MHz

Plataforma Arduino

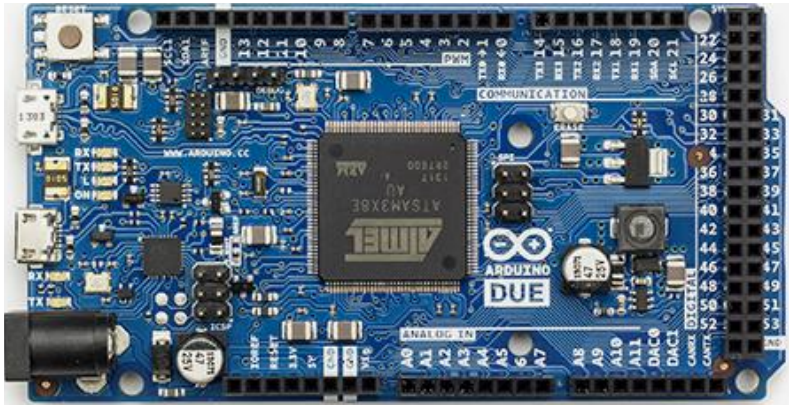
Arduíno NANO

Microcontrolador:	ATmega328
Pinos E/S digitais:	14 (6 com saída PWM)
Pinos entrada analógicos:	8
Memória Flash:	32 KB
SRAM:	2 KB
EEPROM:	1 KB
Velocidade clock:	16MHz

Plataforma Arduino

Arduíno Yún		Linux
Microcontrolador:	ATmega32U4	Atheros AR9331
Pinos E/S digitais:	20	
Pinos entrada analógicos:	12	
Memória Flash:	32 KB	
SRAM:	2,5 KB	64Mb
EEPROM:	1 KB	
Velocidade clock:	16MHz	400Mhz

Plataforma Arduino



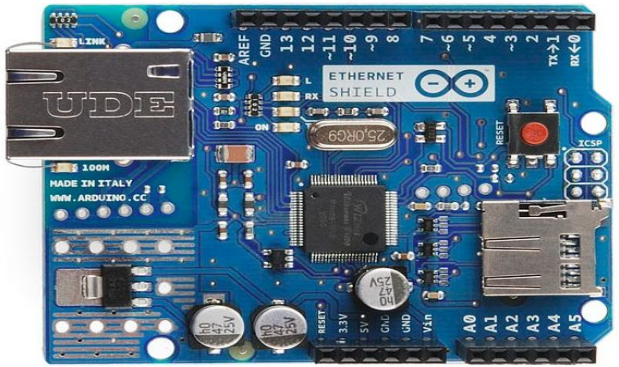
Arduíno DUE

Microcontrolador:	AT91SAM3X8E
Pinos E/S digitais:	54
Pinos entrada analógicos:	12
Pinos saída analógicos:	2 (DAC)
Memória Flash:	512 KB
SRAM:	96 KB
EEPROM:	1 KB
Velocidade clock:	84MHz

Plataforma Arduino

- Shields Arduino
  - São placas com funcionalidades adicionais
  - Possuem pinos e voltagens prontas para serem “encaixadas” no Arduino
  - Normalmente vem acompanhada de bibliotecas de fácil utilização

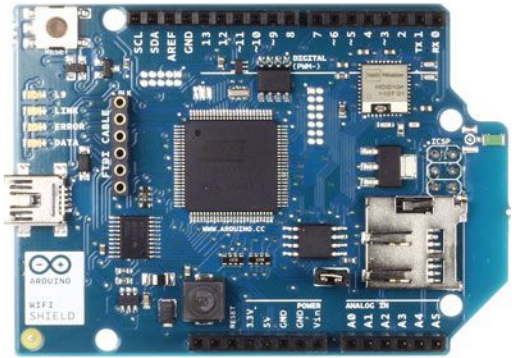
Comunicação com a rede de celular GSM



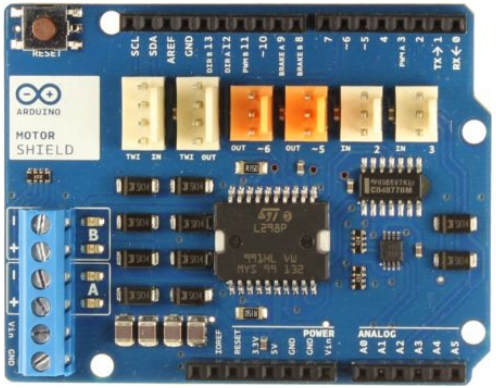
Comunicação Ethernet

Plataforma Arduino

- Shields Arduino (cont.)



Comunicação rede sem fio Wifi



Controlador de motores Motorshield

Display 16x2 LSD Microbot

Ferramentas de desenvolvimento

- Ambientes de desenvolvimento
  - Arduino IDE: https://www.arduino.cc
  - Arduino Editor (online): https://create.arduino.cc/editor

- Ambientes de prototipação
  - TinkerCad (online): https://www.tinkercad.com
  - Circuits.io (online): http://circuits.io

Circuitos elétricos

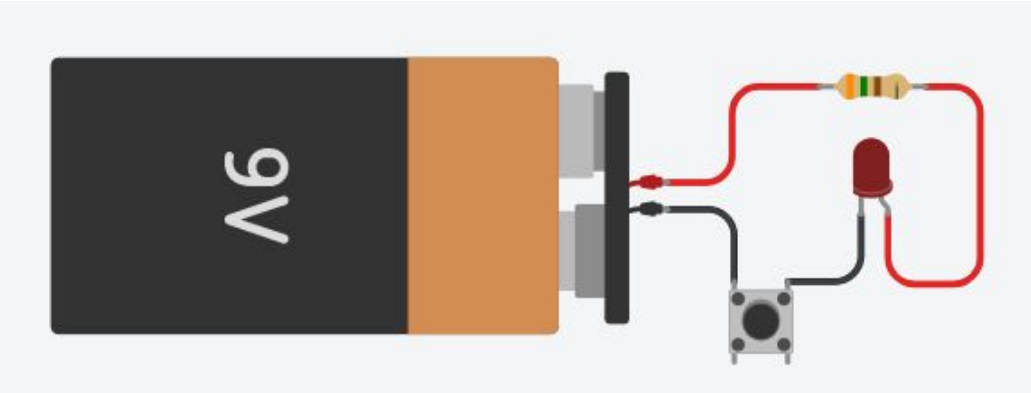
- Ligação de componentes elétricos por meio de fios condutores
  - Formam um caminho fechado que produz uma corrente elétrica

- São minimamente formados por
  - Um gerador elétrico
  - Um condutor em circuito fechado
  - Um elemento capaz de utilizar a energia produzida pelo gerador

- Exemplos de componentes elétricos mais comuns
  - Geradores, receptores, resistores, capacitores e interruptores

Circuitos elétricos

- Exemplo de um circuito elétrico construído no TinkerCad



Diodos

- São dispositivos elétricos para controlar o sentido da corrente elétrica
  - Útil para proteção contra danos por uma corrente excessiva

- Um diodo possui duas polaridades que indicam o sentido da corrente
  - As polaridades são chamadas de anodo e catodo
  - Neste sentido, a corrente flui do anodo para o catodo

- Um LED (*Light-Emitting Diode*) é um tipo de diodo
  - Todo diodo consome um pouco de energia para funcionar
  - No caso de um LED, este consumo refere-se a luz e ao calor emitido

Resistores

- São dispositivos elétricos para alterar o potencial da corrente elétrica
  - Útil para proteção contra danos por uma corrente excessiva

- Para calcular o resistor adequado para um dispositivo, precisa saber
  - A tensão da fonte de alimentação (volts)
  - A tensão de trabalho que o dispositivo suporta (volts)
  - A corrente que o dispositivo suporta (amperes)

- Estas informações são encontradas no *datasheet* do dispositivo
  - O *datasheet* é especificação técnica disponibilizado pelo fabricante

Resistores

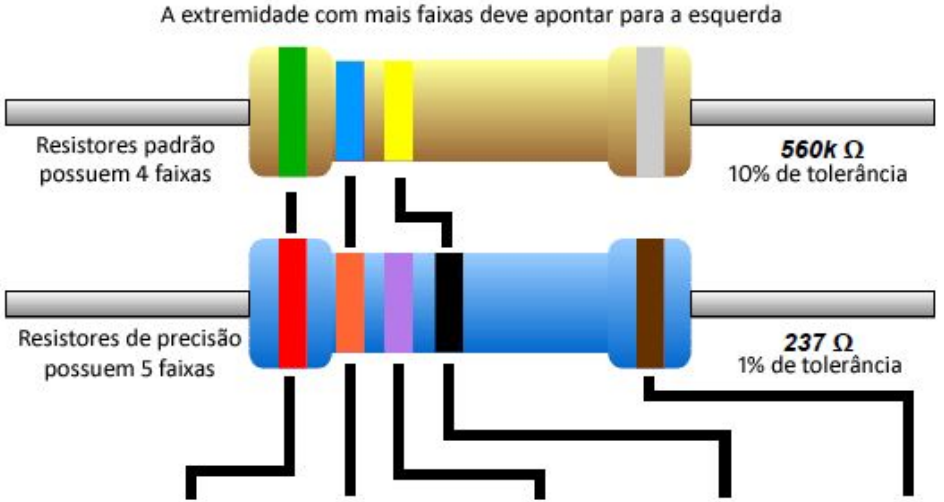
- Cálculo do resistor para um dispositivo
  - $(V_{al} - V_d) / I$
    - V_{al}: tensão da alimentação
    - V_d: tensão do dispositivo
    - I: corrente do dispositivo

- Exemplo de como calcular o resistor adequado para um LED
  - Considere uma bateria com tensão de 9V
  - Considere um LED com tensão de 2V e corrente de 20 miliamperes

$(9 - 2) / 0{,}020 = 350\ \Omega$

Resistores

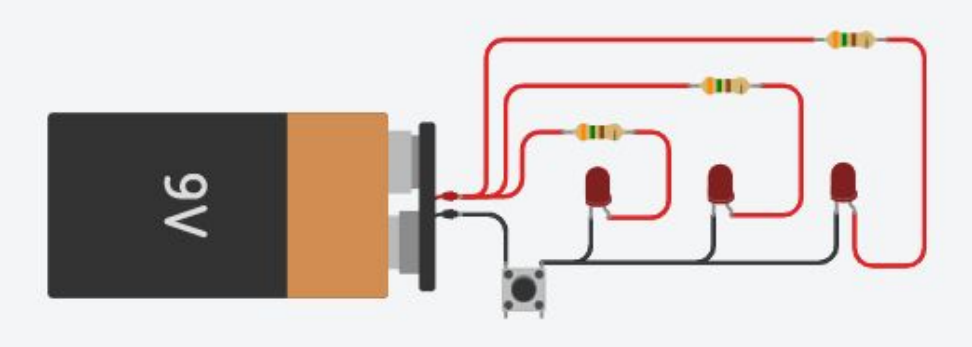
- Tabela de cores
  - Identifica a resistência a ser aplicada sobre o potencial da corrente
  - Resistores padrão possuem 4 faixas de cores
  - Resistores de precisão possuem 5 faixas de cores



Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador	Tolerância
Preto	0	0	0	x 1 Ω	
Marrom	1	1	1	x 10 Ω	+/- 1%
Vermelho	2	2	2	x 100 Ω	+/- 2%
Laranja	3	3	3	x 1K Ω	
Amarelo	4	4	4	x 10K Ω	
Verde	5	5	5	x 100K Ω	+/- .5%
Azul	6	6	6	x 1M Ω	+/- .25%
Violeta	7	7	7	x 10M Ω	+/- .1%
Cinza	8	8	8		+/- .05%
Branco	9	9	9		
Dourado				x .1 Ω	+/- 5%
Prateado				x .01 Ω	+/- 10%

Resistores

- Exemplo de um circuito elétrico com dispositivos ligados em paralelo



Resistores

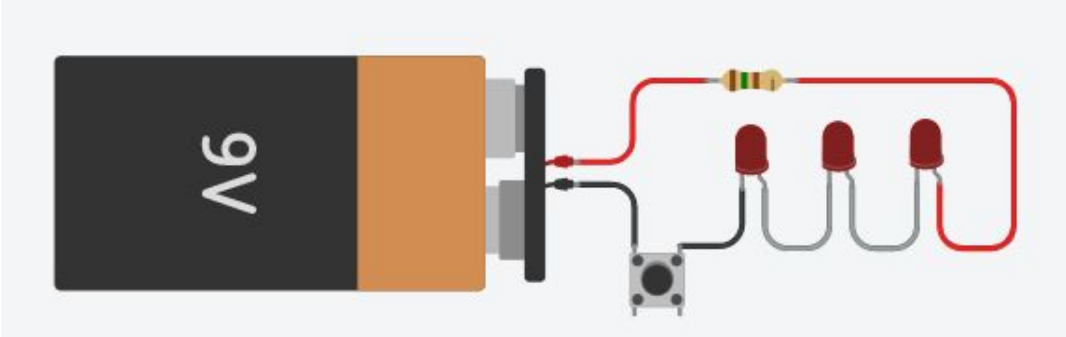
- Cálculo do resistor para dispositivos ligados em série
  - $(V_{al} - V_d) / I$
    - V_{al}: tensão da alimentação
    - V_d: tensão somada dos dispositivos
    - I: corrente do dispositivo

- Exemplo de como calcular o resistor adequado para um LED
  - Uma bateria com tensão de 9V
  - Três LEDs em série com tensão de 2V e corrente de 20 miliamperes

$(9 - 6) / 0{,}020 = 150\ \Omega$

Resistores

- Exemplo de um circuito elétrico com dispositivos ligados em série



Transistores

- São dispositivos para amplificar ou atenuar uma corrente elétrica
  - Exemplos de uso de transistores
    - Amplificar a corrente elétrica (microfone)
    - Impedir a passagem da corrente (interruptor)

- São componentes fundamentais na computação
  - Quando ligado (alta corrente), o computador lê o bit 0
  - Quando desligado (alta baixa), o computador lê o bit 1

- Transistores de junção bipolar
  - São os transistores BJT mais comuns (*Bipolar Junction Transistor*)
  - Formados por camadas de silício com diferentes cargas

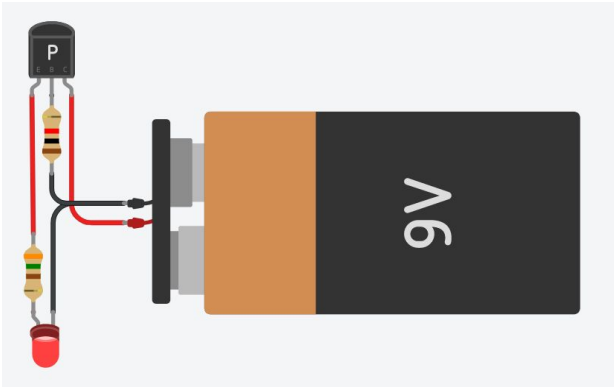
Transistores

- Transistores de junção bipolar (cont.)
  - São formados por: coletor, base e emissor
  - O emissor e o coletor estão ligados em série a um circuito elétrico
    - Este circuito pode ou não ser alimentado por uma corrente
  - A base controla o estado do transistor
    - Ou seja, a base determina se o circuito estará aberto ou fechado

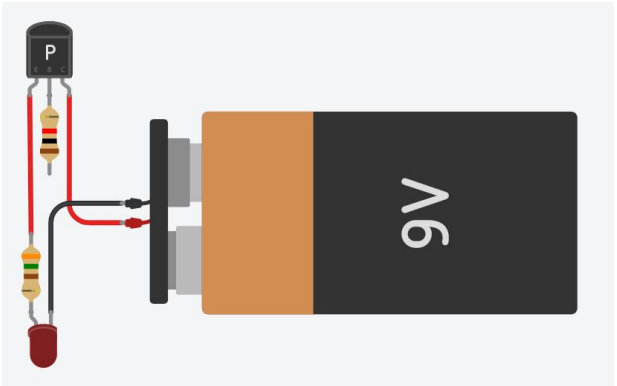
- Um transistor BJT pode ser NPN ou PNP
  - O tipo NPN utiliza carga positiva na base para fechar o circuito
  - O tipo PNP utiliza carga negativa na base para fechar o circuito

Transistores

- Exemplo de transistor como interruptores
  - É comum o uso de um resistor na sua base
    - Devido ao fato do transistor amplificar o sinal
  - No exemplo foi utilizado uma resistência de 1K ohms



Circuito fechado, a base está sendo alimentada



Circuito aberto, a base não está sendo alimentada

Exercícios

1. Monte um circuito elétrico contendo um três botões e três LEDs de cores diferentes. Quando cada botão for pressionado, apenas um LED deve acender. O circuito deve ser montado com os LEDs ligados em paralelo.
    - Exemplo: LEDs vermelho, azul e verde e botões 1, 2 e 3
        - Botão 1 acende LED vermelho
        - Botão 2 acende LED azul
        - Botão 3 acende LED verde

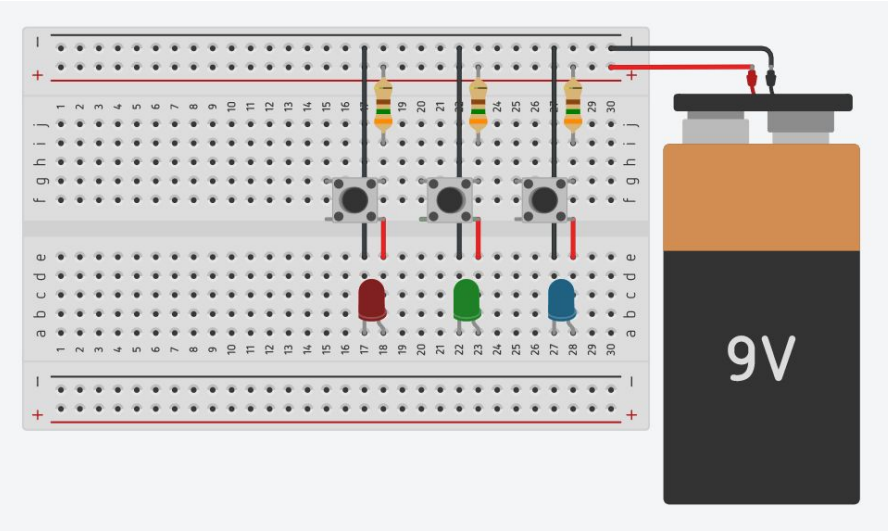
2. Monte um circuito elétrico contendo um dois botões e três LEDs de cores diferentes. Ao pressionar um dos botões, dois LEDs devem acender. Ao pressionar o outro botão, apenas um LED deve acender. O circuito deve ser montado com dois LEDs ligados em série e um em paralelo.
    - Exemplo: LEDs vermelho, azul e verde e botões 1 e 2
        - Botão 1 acende LED vermelho e azul (ligados em série)
        - Botão 2 acende LED verde (ligado em paralelo)

Exercícios

3. Monte um circuito elétrico contendo um dois botões e três LEDs de cores diferentes. Ao pressionar um dos botões, os dois primeiros LEDs devem acender. Ao pressionar o outro botão, os dois últimos LED devem acender. O circuito deve ser montado com duas ligações de LEDs em série.
    - Exemplo: LEDs vermelho, azul e verde e botões 1 e 2
        - Botão 1 acende LEDs vermelho e azul
        - Botão 2 acende LEDs azul e verde

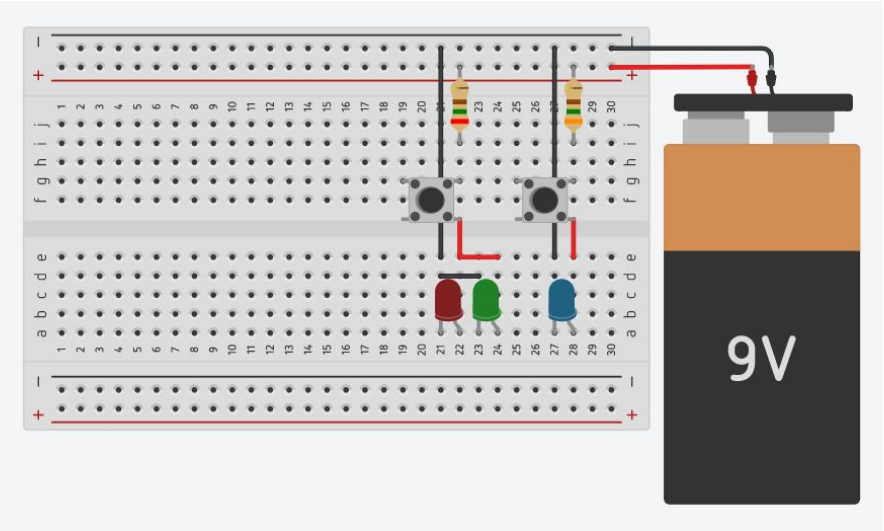
4. Monte um circuito elétrico contendo um três botões e três LEDs de cores diferentes. Ao pressionar o primeiro botão, o primeiro LED deve acender. Ao pressionar o segundo botão, os dois primeiros LEDs devem acender. Por fim, ao pressionar o último botão, todos os LEDs devem acender.
    - Exemplo: LEDs vermelho, azul e verde e botões 1, 2 e 3
        - Botão 1 acende LED vermelho
        - Botão 2 acende LEDs vermelho e azul
        - Botão 3 acende LEDs vermelho, azul e verde

Exercícios



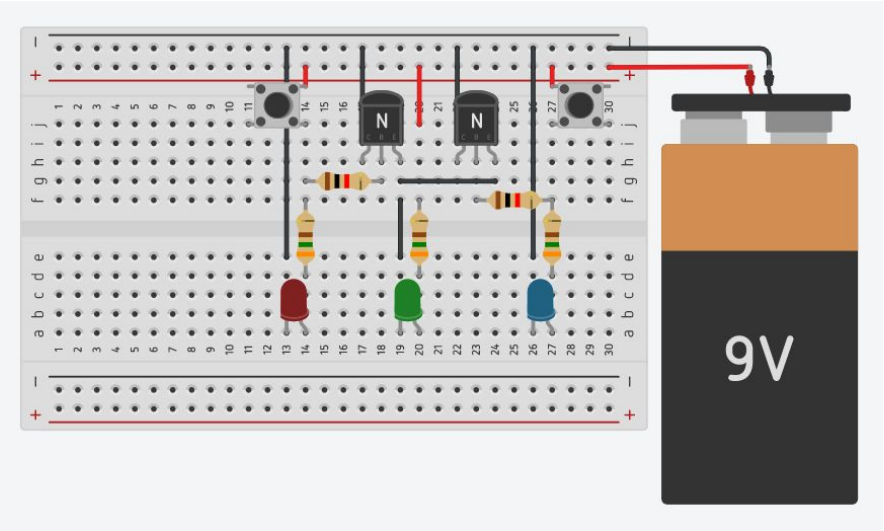
Exercício 1

Exercícios



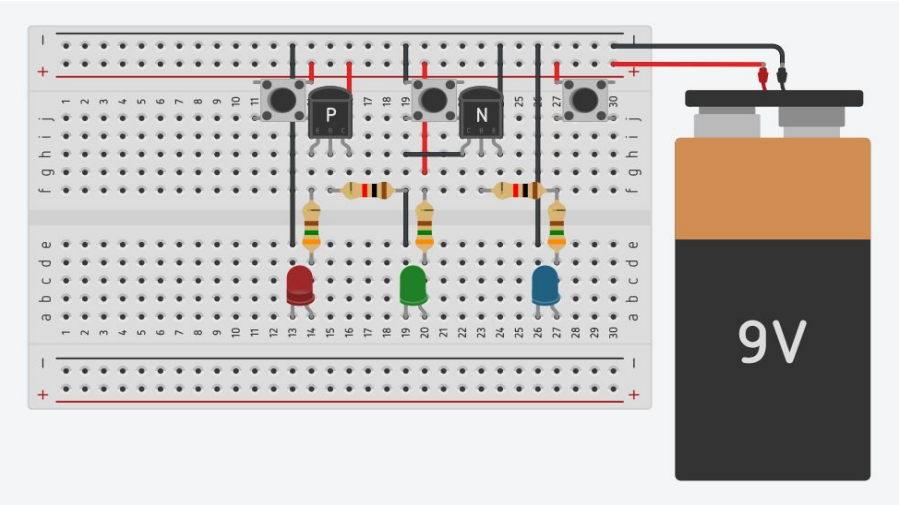
Exercício 2

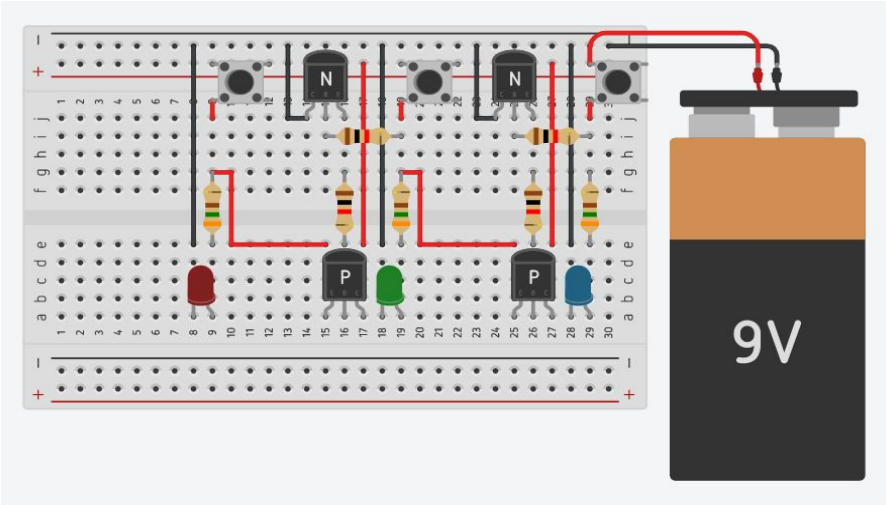
Exercícios



Exercício 3

Exercícios

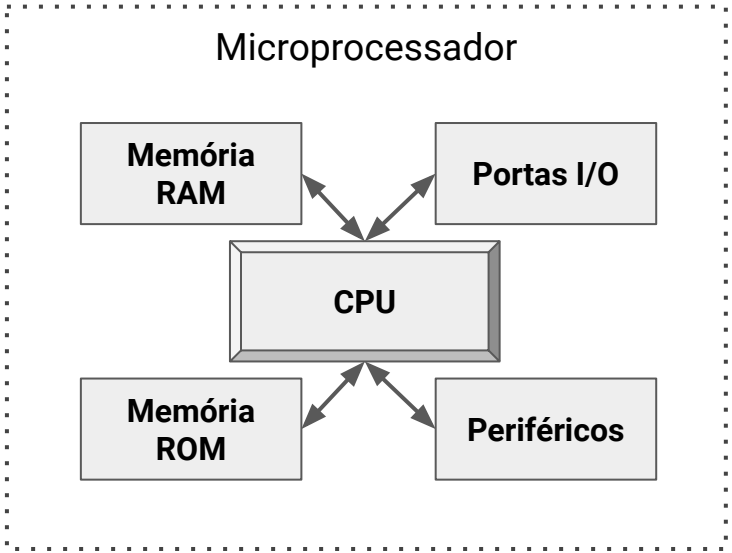


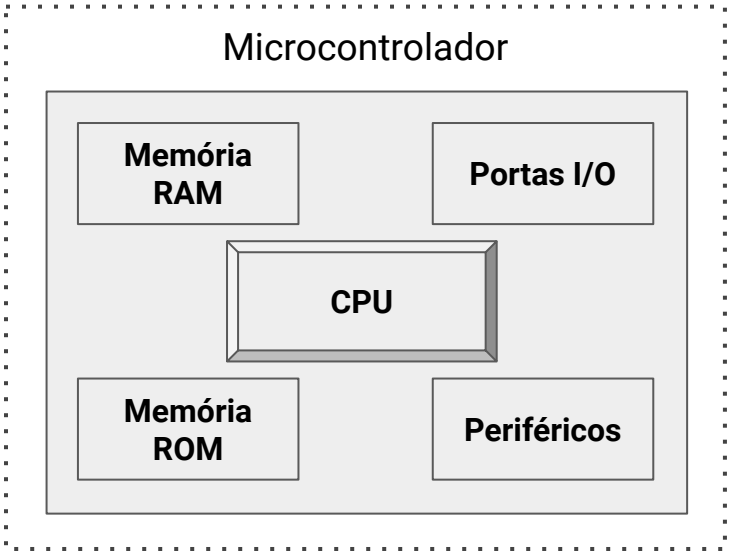


Exercício 4

Arquitetura Arduino

- Processador e componentes estão em um único circuito integrado
  - Em um microprocessador, os componentes também estão conectados
  - Porém, não fazem parte do mesmo circuito integrado



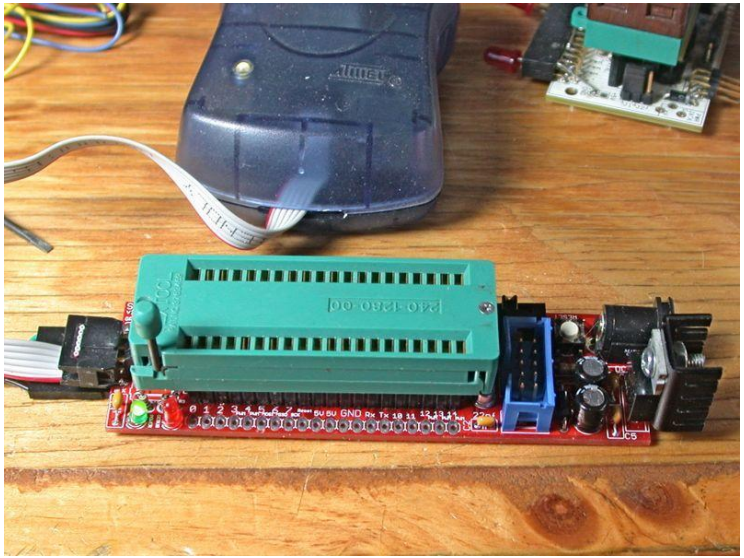


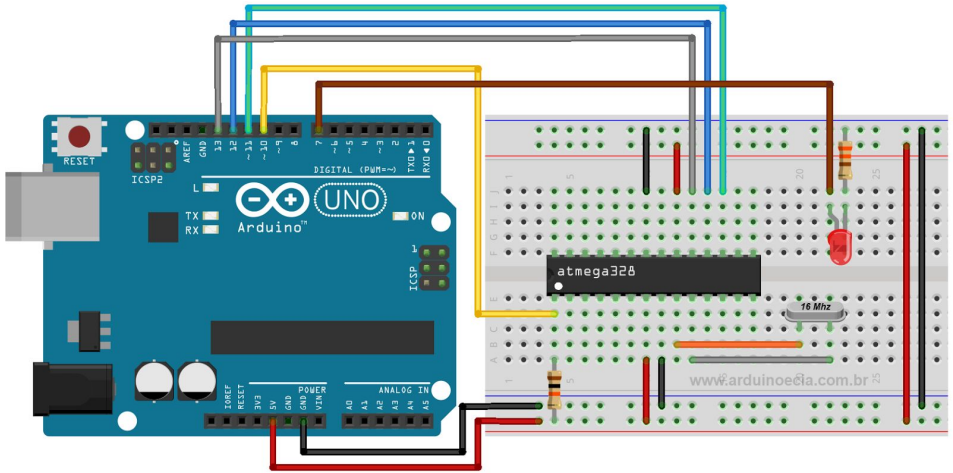
Arquitetura Arduino

- *Bootloader*
  - É um programa residente no microcontrolador
  - O *bootloader* é responsável por
    - Realizar algumas verificações e atualizações do microcontrolador
    - Fazer a comunicação serial com o PC para envio de novos programas
    - Iniciar o programa carregado, após o microcontrolador ser ligado
  - É necessário um *hardware* especial para gravar um programa em um microcontrolador (*ATMega*) sem usar o Arduino
    - Normalmente chamado de *bootburner*
    - É possível usar outro Arduino como gravador
    - Os microcontroladores do Arduino já vem com um programa pré gravado

Arquitetura Arduino

- *Bootloader* (cont.)





IDE do Arduino

- O programa carregado no Arduino é um programa escrito em C/C++
  - A include `Arduino.h` contém endereços e códigos em hexadecimal para todas as constantes usadas nos programas

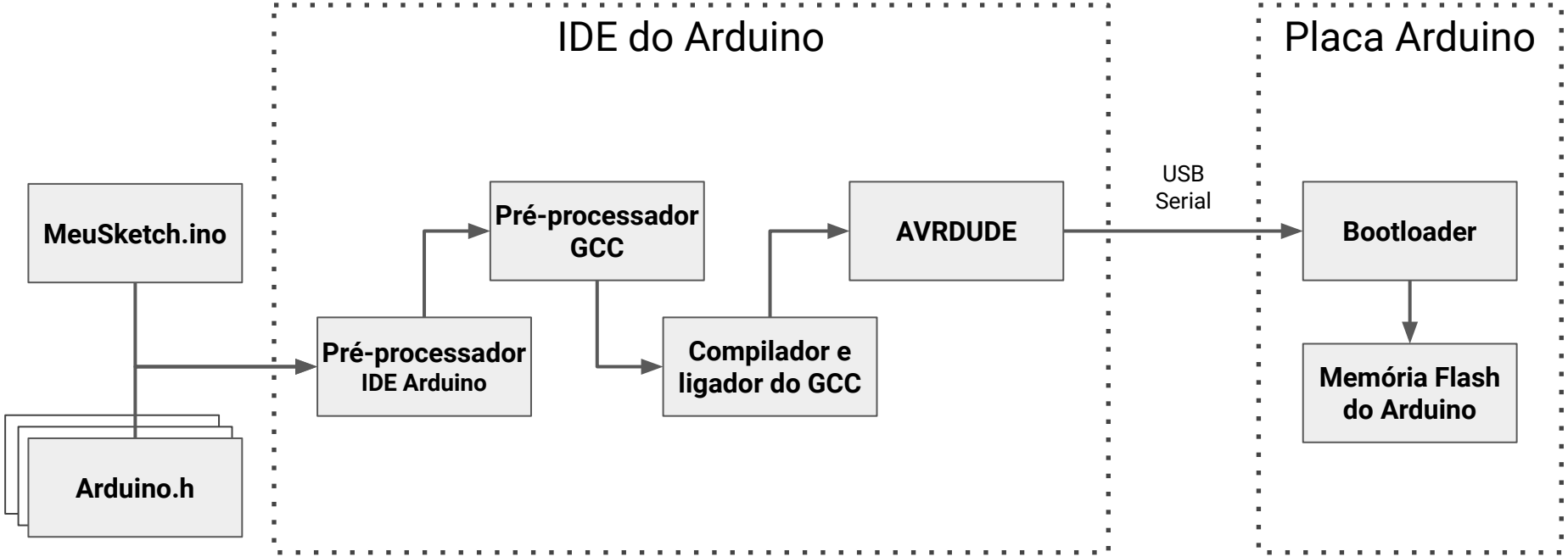
  - Dentro do diretório onde a IDE do Arduino foi instalada encontra-se
    - Os arquivos com extensão `.h` e `.cpp` utilizados pelo projeto
    - O programa principal com extensão `.ino`

  - A IDE do Arduino é responsável por
    - Pré-processar, compilar e ligar os códigos do projeto
    - Carregar o programa no Arduino

  - Um projeto na IDE Arduino é chamado de *sketch*

IDE do Arduino

- Gravação de um *scketch* pela IDE do Arduino



Programa básico no Arduino

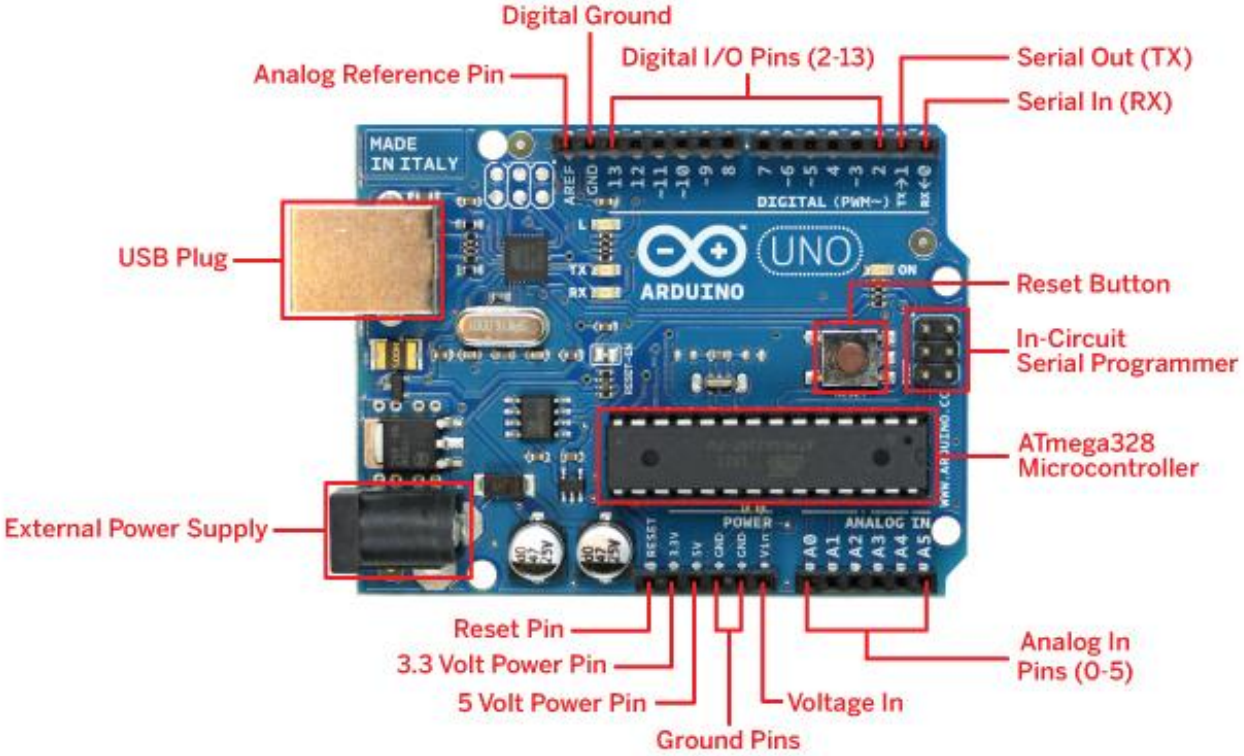
- Após a inicialização o *bootloader* executa o programa carregado
  - O Arduino armazena apenas um programa a ser invocado pelo *bootloader*
    - O programa é executado indefinidamente ou até ser interrompido
  - O programa é escrito em C ou C++
    - Um programa pode usar e chamar outros arquivos e classes em C ou C++
    - Isto ajuda a organizar o código, tornando-o mais legível
  - Todo programa deve conter duas funções: `setup` e `loop`
    - `setup`: chamada uma única vez no momento que o programa é executado
    - `loop`: chamada repetidamente a cada ciclo do microcontolador

Programa básico no Arduino

- Exemplo de um programa básico no Arduino

```
void setup() {
    //Código de inicialização do programa carregado
}

void loop() {
    //Código do programa carregado
}
```

Componentes do Arduino



Portas digitais no Arduino

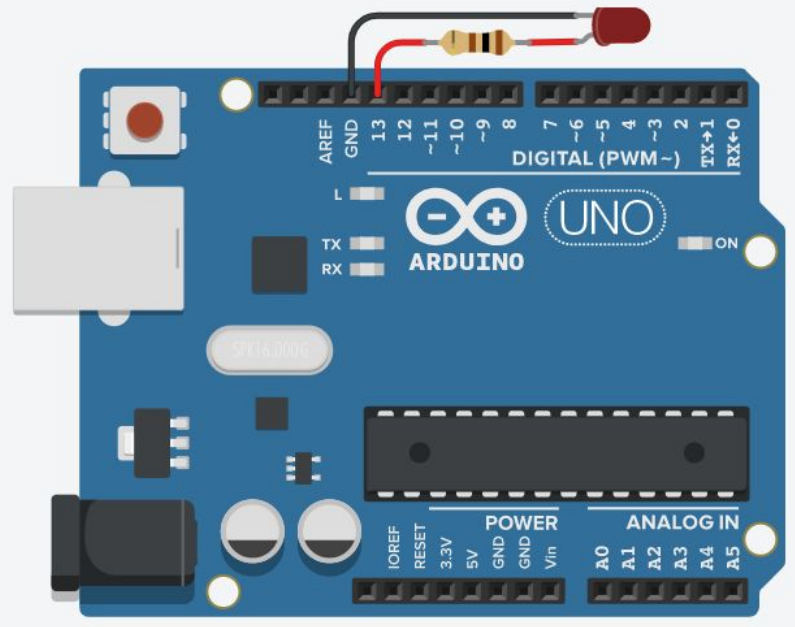
- Portas digitais para entrada e saída
  - Funcionam de forma binária
    - Totalmente ligadas (5v)
    - Totalmente desligadas (0v)

  - Função `pinMode(<pino>, INPUT | OUTPUT)`
    - Configura um pino digital como entrada (`INPUT`) ou saída (`OUTPUT`)
    - `INPUT` e `OUTPUT` são constantes declaradas na include `Arduino.h`

  - Função `digitalWrite(<pino>, HIGH | LOW)`
    - Permite colocar um pino de saída como 0v (`LOW`) ou 5v (`HIGH`)
    - `HIGH` e `LOW` são constantes declaradas na include `Arduino.h`

Portas digitais para saída

- Exemplo de um programa que utiliza uma porta digital de saída



Portas digitais para saída

- Exemplo de um programa que utiliza uma porta digital de saída

```
int pinoLed = 13;

void setup() {
    pinMode(pinoLed, OUTPUT); //Configura o pino 13 como saída
}

void loop() {
    digitalWrite(pinoLed, HIGH); //Coloca o pino 13 como 5v
    delay(500); //Aguarda 500 ms

    digitalWrite(pinoLed, LOW); //Coloca o pino 13 como 0v
    delay(500); //Aguarda 500 ms
}
```

Exercícios

5. Crie um *sketch* no Arduino para controlar um circuito elétrico contendo três LEDs de cores diferentes. Cada LED deve ficar aceso por 1 segundo e, em seguida, o próximo LED do circuito deve ser acendido. Deve haver somente 1 LED aceso por vez.
    - Exemplo: LEDs vermelho, azul e verde
        - Desliga todos os LEDs e acende LED vermelho por 1 segundo
        - Desliga todos os LEDs e acende LED azul por 1 segundo
        - Desliga todos os LEDs e acende LED verde por 1 segundo

6. Crie um *sketch* no Arduino para controlar um circuito elétrico contendo três LEDs de cores diferentes. O primeiro e o segundo LED devem ficar acesos por 1 segundo e, em seguida, o terceiro LED do circuito deve ser acendido. Utilize três portas digitais, isto é, uma para cada LED.
    - Exemplo: LEDs vermelho, azul e verde
        - Desliga todos os LEDs e acende os LEDs vermelho e azul por 1 segundo
        - Desliga todos os LEDs e acende LED verde por 1 segundo

7. Faça o exercício 6 utilizando apenas duas portas digitais.

Exercícios

8. Crie um *sketch* no Arduino para controlar um circuito elétrico contendo três LEDs de cores diferentes. O primeiro e o segundo LED devem ficar acesos por 1 segundo e, em seguida, o segundo e o terceiro LED do circuito devem ser acendidos. Utilize três portas digitais, isto é, uma para cada LED.
    - Exemplo: LEDs vermelho, azul e verde
        - Desliga todos os LEDs e acende os LEDs vermelho e azul por 1 segundo
        - Desliga todos os LEDs e acende os LEDs azul e verde por 1 segundo

9. Faça o exercício 8 utilizando duas portas digitais.

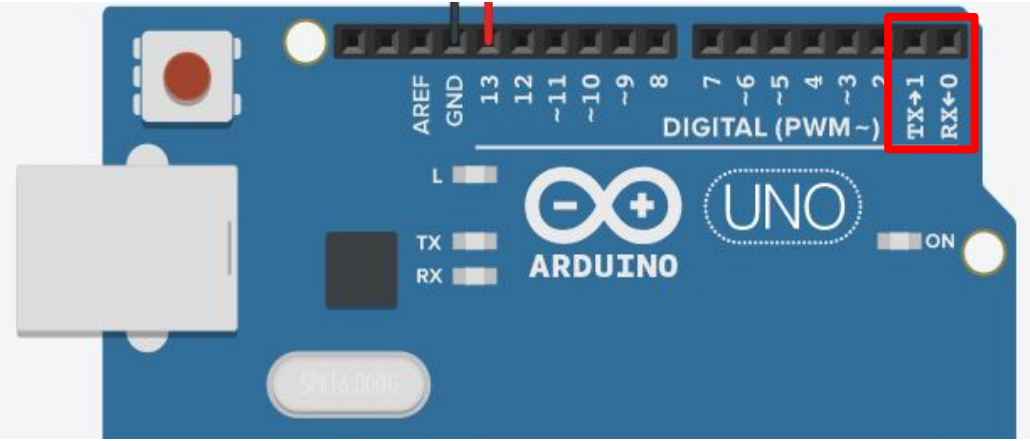
10. Crie um *sketch* no Arduino para controlar um circuito elétrico contendo três LEDs de cores diferentes. O primeiro LED deve ficar aceso por 1 segundo. Em seguida, os dois primeiros LEDs devem ficar acesos por 1 segundo. Por fim, todos os LEDs devem ficar acesos por 1 segundo.
    - Exemplo: LEDs vermelho, azul e verde
        - Desliga todos os LEDs e acende LED vermelho por 1 segundo
        - Desliga todos os LEDs e acende LED vermelho e azul por 1 segundo
        - Desliga todos os LEDs e acende LED vermelho, azul e verde por 1 segundo

Comunicação serial

- É possível realizar uma comunicação serial de entrada a saída
  - Comunicação UART (*Universal asynchronous receiver-transmitter*)
    - É um protocolo *full-duplex*
  - É através deste canal que são carregados os programas no Arduino
    - Porém, este canal é muito utilizado para envio e recebimento de dados
  - A IDE do Arduino dispõe do Monitor Serial que utiliza comunicação serial
    - Permite enviar e receber dados do Arduino sem a necessidade de uso de uma ferramenta ou componente externo

Comunicação serial

- A comunicação serial utiliza um sinal TTL (*Transistor-Transistor Logic*)
  - Este canal está ligado aos pinos digitais 0 (RX) e 1 (TX)
    - RX é o pino receptor
    - TX é o pino trasmissor



Comunicação serial

- Principais funções para comunicação serial
  - `Serial.begin(<velocidade>)`
    - Utilizada para configurar o canal comunicação
    - Deve ser executada apenas uma vez (dentro do `setup`)
    - Deve ser especificada a velocidade de comunicação (taxa de envio)
  - `Serial.read()`
    - Captura o primeiro *byte* caso algum dado tenha sido enviado para o Arduino
  - `Serial.readString()`
    - Semelhante ao `Serial.read()`
    - Porém, já retorna os dados convertidos para o tipo `String`

Comunicação serial

- Principais funções para comunicação serial (cont.)
  - `Serial.println(<mensagem>)`
    - Envia uma mensagem do Arduino para o receptor, quebrando uma linha
  - `Serial.print(<mensagem>)`
    - Envia uma mensagem do Arduino para o receptor
    - Mas sem quebrar uma linha
  - `Serial.write()`
    - Envia dados para o Arduino para o receptor em formato binário

Comunicação serial

- Exemplo de comunicação serial para capturar e emitir mensagem

```
String dados;

void setup() {
    Serial.begin(9600); //Inicia o canal com velocidade de 9600 bps
    while (!Serial); //Aguarda a comunicação serial começar
}

void loop() {
    if (Serial.available() > 0) { //Verifica se a comunicação está disponível
        dados = Serial.readString(); //Captura uma string

        Serial.print("O Arduino recebeu: "); //Emite uma string
        Serial.println(dados); //Emite uma string
    }
}
```

Comunicação serial

- Exemplo de comunicação serial para controlar um LED

```
int pinoLed = 13;
bool liga = false;

String mensagem = "";

void setup() {
    pinMode(pinoLed, OUTPUT); //Configura o pino 13 como saída

    Serial.begin(9600);
    delay(50);

    Serial.println("Iniciando o Arduino");
}
```

Comunicação serial

- Exemplo de comunicação serial para controlar um LED (cont.)

```
void loop() {
    if (Serial.available() > 0){
        mensagem = Serial.readString();
        Serial.print("Recebido: ");
        Serial.println(mensagem);

        if (mensagem == "LIGA"){
            liga = true;
        } else if (mensagem == "DESLIGA") {
            liga = false;
        }
    }
    digitalWrite(pinoLed, liga ? HIGH : LOW);
}
```

Exercícios

11. Refaça o projeto do exercício 1 para controlar os LEDs pelo Arduino, substituindo os botões por comandos via comunicação serial.

12. Refaça o projeto do exercício 2 para controlar os LEDs pelo Arduino, substituindo os botões por comandos via comunicação serial.

13. Refaça o projeto do exercício 3 para controlar os LEDs pelo Arduino, substituindo os botões por comandos via comunicação serial.

14. Refaça o projeto do exercício 4 para controlar os LEDs pelo Arduino, substituindo os botões por comandos via comunicação serial.

15. Crie um *sketch* no Arduino para reproduzir o comportamento de um semáforo utilizando 3 LEDs. O programa deve permitir a configuração do tempo de cada estágio do semáforo a partir de uma comunicação serial.

Portas digitais para entrada

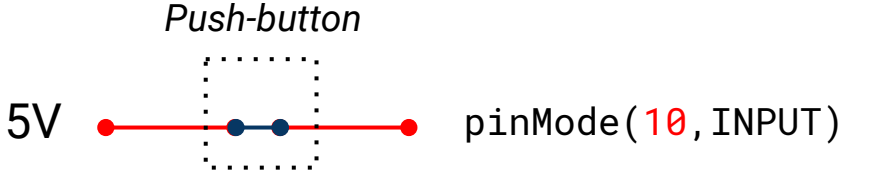
- Permitem detectar alterações de tensão em portas digitais
  - Como porta digital é binária, os possíveis valores são `HIGH` e `LOW`
    - `HIGH` quando a porta recebe 5v
    - `LOW` quando a porta recebe 0v

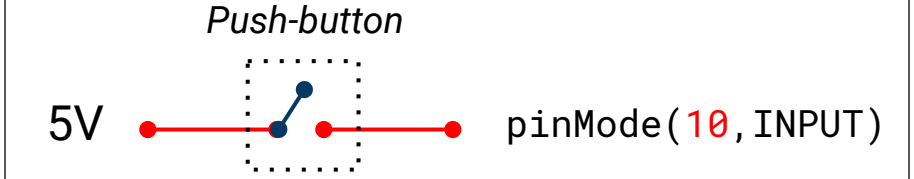
- As portas digitais funcionam tanto como entrada quanto como saída
  - É necessário especificar o comportamento da porta na função `setup`
  - A porta não pode assumir entrada ou saída ao mesmo tempo

```
int pinoLed = 10;

void setup() {
    pinMode(pinoLed, INPUT); //Configura o pino 10 como uma entrada digital
}
```

Resistores *pull-up* e *pull-down*

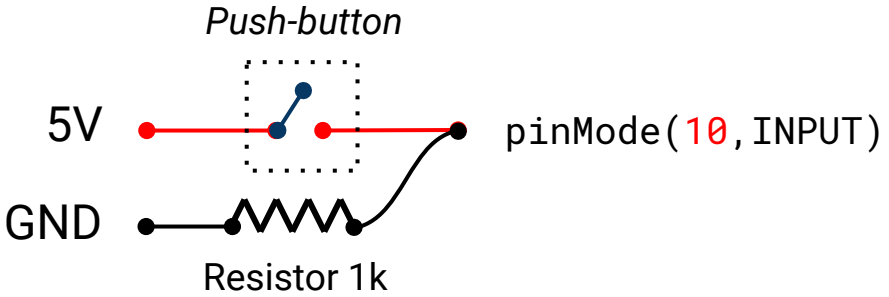
- Ao ligar um botão ou outro dispositivo no circuito, as extremidades precisam estar conectado no negativo (*ground*) e no positivo
  - Um dispositivo não deve estar conectado apenas em um polo
  - Se estiver pode ocorrer uma oscilação entre positivo e negativo
    - Isto faz com que a leitura do estado do dispositivo oscile aleatoriamente
    - Este comportamento é chamado de *three-state*





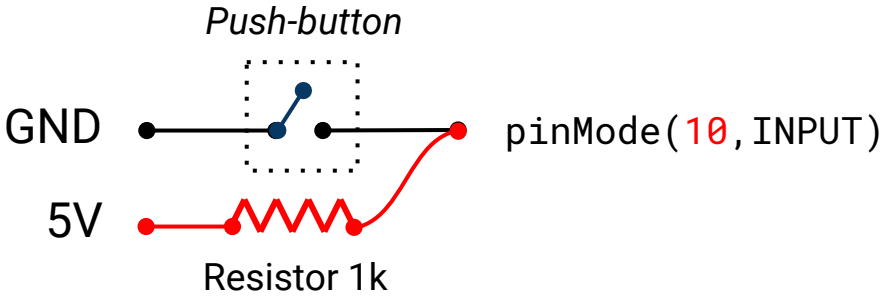
Resistores *pull-up* e *pull-down*

- Para evitar este estado usa-se resistores *pull-down* ou *pull-up*
  - Resistor *pull-down*
    - Botão não pressionado: porta efetuará a leitura 0v (`LOW`)
    - Botão pressionado: porta efetuará a leitura de 5v (`HIGH`)



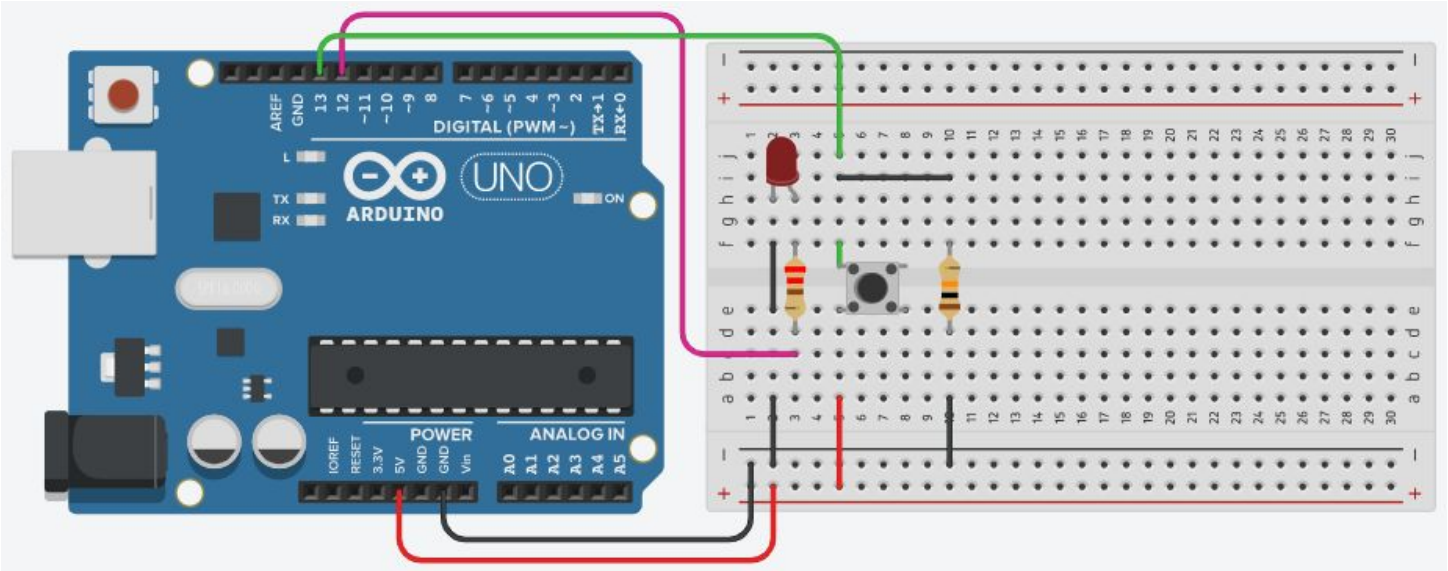
Resistores *pull-up* e *pull-down*

- Para evitar este estado usa-se resistores *pull-down* ou *pull-up*
  - Resistor *pull-up*
    - Botão não pressionado: porta efetuará a leitura 5v (HIGH)
    - Botão pressionado: porta efetuará a leitura de 0v (LOW)



Resistores *pull-up* e *pull-down*

- Exemplo de resistor *pull-down*



Resistores *pull-up* e *pull-down*

- Exemplo de resistor *pull-down* (cont.)

```
int pinoBotao = 13;
int pinoLed = 12;

void setup() {
    pinMode(pinoBotao, INPUT);
    pinMode(pinoLed, OUTPUT);

    Serial.begin(9600);
}
```

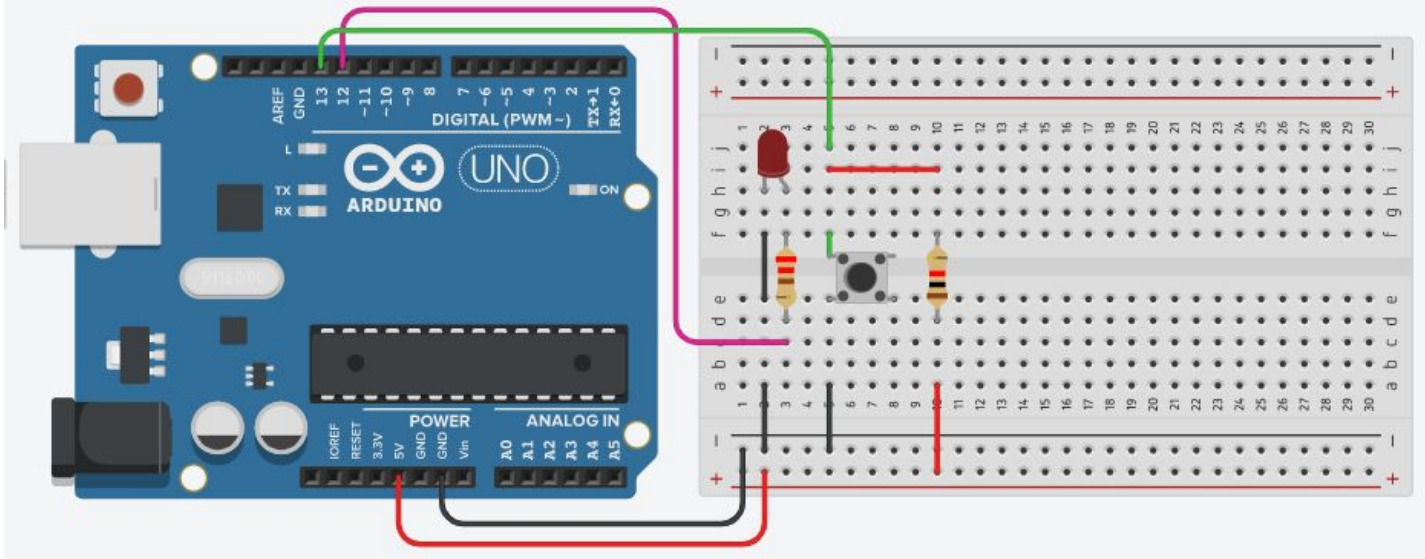
Resistores *pull-up* e *pull-down*

- Exemplo de resistor *pull-down* (cont.)

```
void loop() {
    if (digitalRead(pinoBotao) == HIGH){ //Tensão alta, botão pressionado
        Serial.println("Botao pressionado");
        digitalWrite(pinoLed, HIGH);
    } else {
        Serial.println("Botao nao pressionado");
        digitalWrite(pinoLed, LOW);
    }
}
```

Resistores *pull-up* e *pull-down*

- Exemplo de resistor *pull-up*



Resistores *pull-up* e *pull-down*

- Exemplo de resistor *pull-up* (cont.)

```
int pinoBotao = 13;
int pinoLed = 12;

void setup() {
    pinMode(pinoBotao, INPUT);
    pinMode(pinoLed, OUTPUT);

    Serial.begin(9600);
}
```

Resistores *pull-up* e *pull-down*

- Exemplo de resistor *pull-down* (cont.)

```
void loop() {
    if (digitalRead(pinoBotao) == LOW){ //Tensão baixa, botão pressionado
        Serial.println("Botao pressionado");
        digitalWrite(pinoLed, HIGH);
    } else {
        Serial.println("Botao nao pressionado");
        digitalWrite(pinoLed, LOW);
    }
}
```

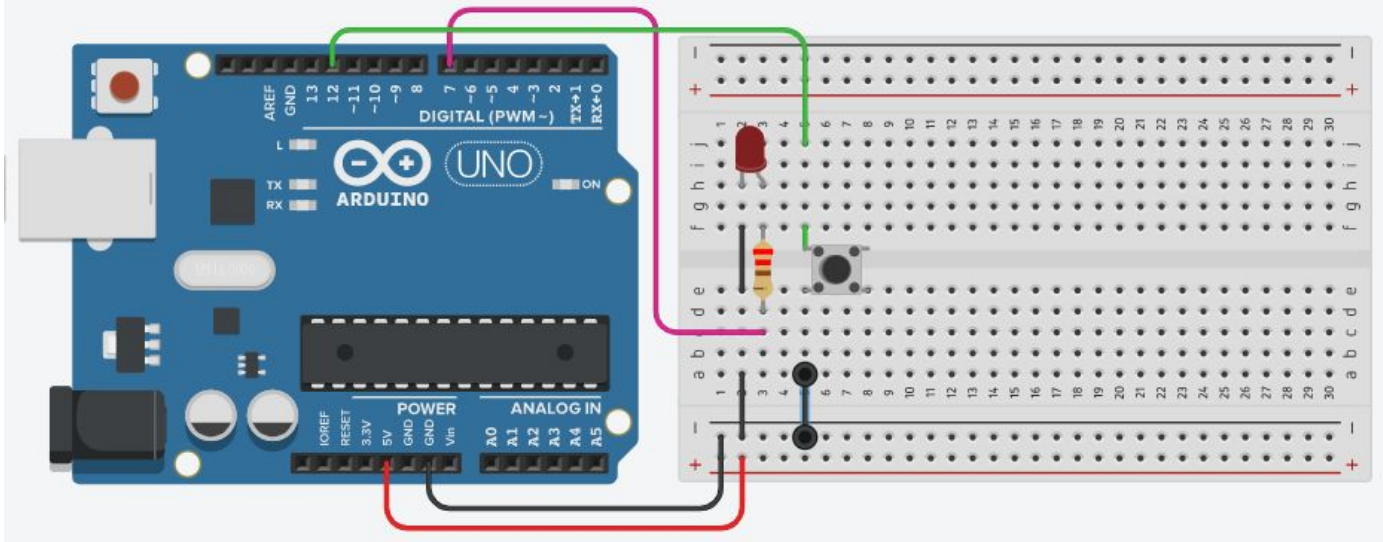
Resistores *pull-up* e *pull-down*

- O microprocessador dispõe de resistores *pull-up* em portas digitais
  - Para ativar, basta configurar a porta digital com o modo `INPUT_PULLUP`
    - Desta forma não é necessário adicionar um resistor no circuito elétrico
    - Mas atenção, nem todos os pinos possuem este resistor *built-in*

```
void setup() {
    pinMode(12, INPUT_PULLUP);
}
```

Resistores *pull-up* e *pull-down*

- Exemplo do modo `INPUT_PULLUP`



Resistores *pull-up* e *pull-down*

- Exemplo do modo `INPUT_PULLUP` (cont.)

```
int pinoBotao = 12;
int pinoLed = 7;

void setup() {
    pinMode(pinoBotao, INPUT_PULLUP);
    pinMode(pinoLed, OUTPUT);

    Serial.begin(9600);
}
```

Resistores *pull-up* e *pull-down*

- Exemplo do modo `INPUT_PULLUP` (cont.)

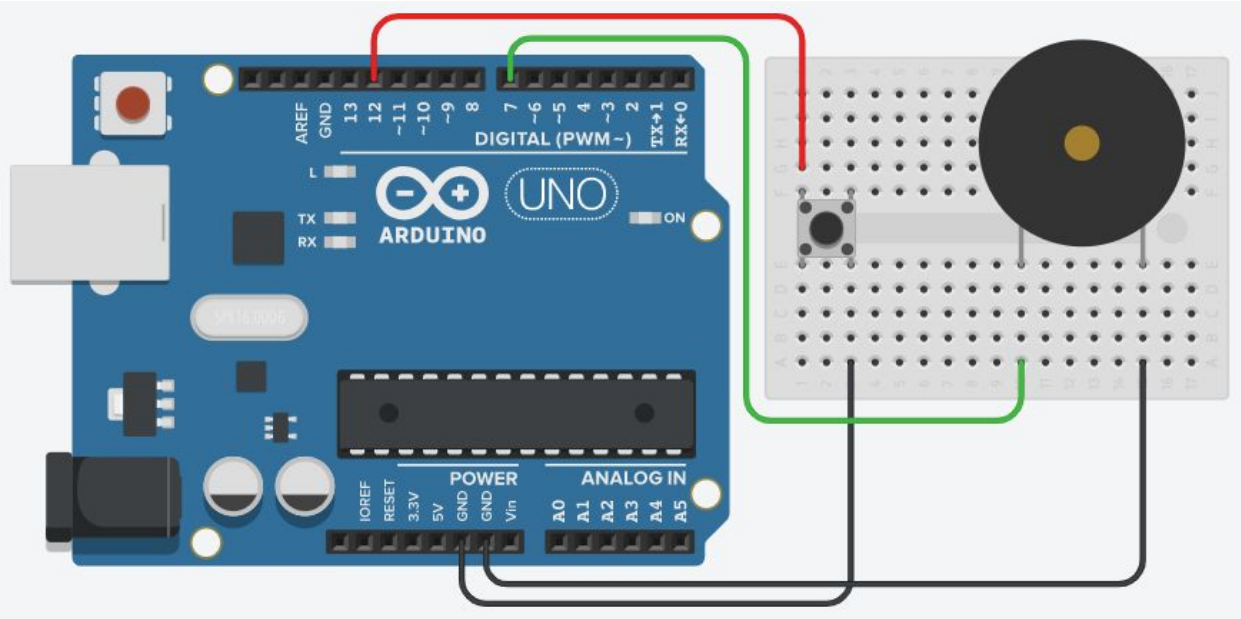
```
void loop() {
    if (digitalRead(pinoBotao) == LOW){ //Tensão baixa, botão pressionado
        Serial.println("Botao pressionado");
        digitalWrite(pinoLed, HIGH);
    } else {
        Serial.println("Botao nao pressionado");
        digitalWrite(pinoLed, LOW);
    }
}
```

Resistores *pull-up* e *pull-down*

- Como identificar a mudança de estado de uma porta digital?
  - Uma forma bastante comum é utilizar uma variável auxiliar
    - Esta variável detém o último estado lido da porta digital
  - Neste caso, é detectada uma mudança no estado da porta quando
    - For realizada uma nova leitura e esta for diferente do último estado
  - Esta técnica é útil para dispositivos que não precisam ser acionados continuamente, enquanto o botão estiver pressionado, por exemplo

Resistores *pull-up* e *pull-down*

- Exemplo de identificação do estado da porta digital



Resistores *pull-up* e *pull-down*

- Exemplo de identificação do estado da porta digital (cont.)

```
int pinoBotao = 12;
int pinoBuzzer = 7;

int tempoBuzzer = 500;      //Tempo que o buzzer ficará acionado, quando ativo
int frequenciaBuzzer = 100; //Frequência em Hz do buzzer

int ultimoEstadoBotao = HIGH; //Variável com o estado anterior da porta 12
int estadoBotao = HIGH;       //Variável com o estado atual da porta 12

void setup() {
    pinMode(pinoBuzzer, OUTPUT);
    pinMode(pinoBotao, INPUT_PULLUP);

    Serial.begin(9600);
}
```

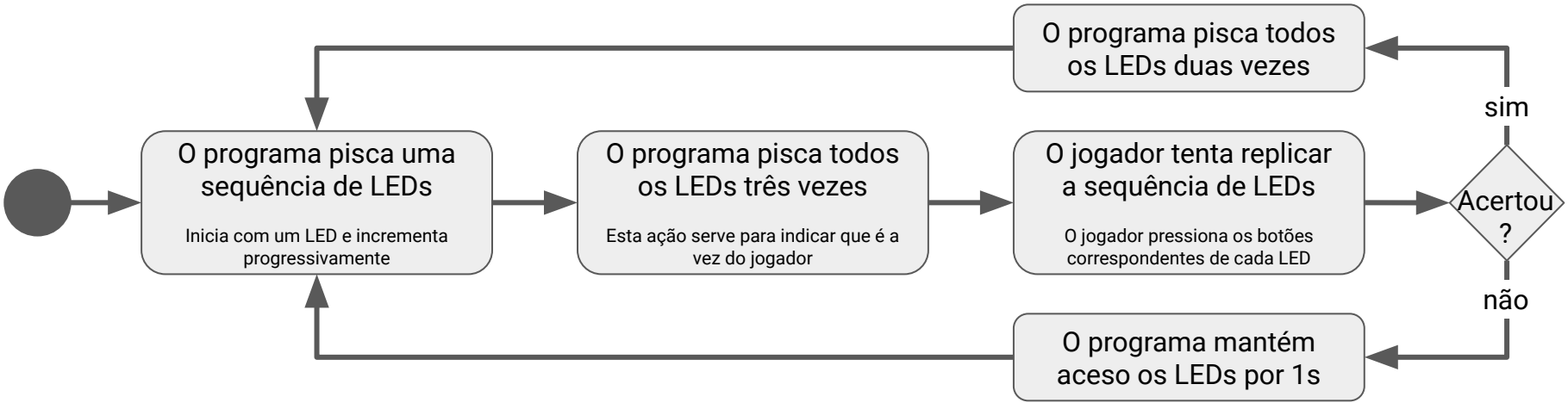
Resistores *pull-up* e *pull-down*

- Exemplo de identificação do estado da porta digital (cont.)

```
void loop() {
    estadoBotao = digitalRead(pinoBotao);

    Serial.println(estadoBotao == LOW ? "Pressionado" :  "Não pressionado");

    if (estadoBotao == LOW && ultimoEstadoBotao == HIGH){
        //Como os estados estão diferentes, será acionado o buzzer
        tone(pinoBuzzer, frequenciaBuzzer, tempoBuzzer);
    }

    ultimoEstadoBotao = estadoBotao; //Atualiza o último estado
}
```

Exercícios

16. Crie um jogo da memória (Genius) com o Arduino. O projeto deve conter 4 LEDs e 4 botões e um buzzer. O programa carregado deve controlar os LEDs, acionando uma sequência de luzes aleatoriamente. O jogo inicia com o acionamento de um LED e, nas jogadas sucessivas, deve aumentar a sequência sucessivamente. O aumento ocorre conforme o jogador acerta a sequência de cores acionadas. O jogador deve observar a sequência das luzes, memorizá-la e, ao término da sequência, replicar a sequência das luzes utilizando os botões.
    - Funcionamento do jogo: https://www.youtube.com/watch?v=ttJ2XpZIs3E



Introdução ao Arduino

Sistemas embarcados

Prof. Allan Rodrigo Leite