*PROFESSOR DANILO* *PROJETOS DE CIÊNCIAS – 17/03/2022*

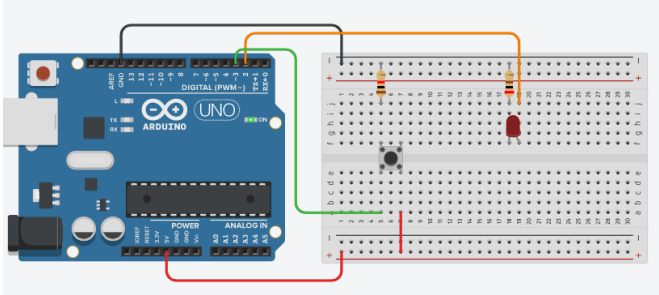
**AULA 07**

Na aula de hoje faremos uso da tabela verdade, dos condicionais if e else e entenderemos a entrada digital.

Vamos começar com o circuito, depois a análise do código. Lembre-se que o site do professor possui alguns vídeos para te ajudar.

## MONTAGEM DO CIRCUITO

Veja se o esquema do tinkercad é suficiente:



Cuidado com o push button, uma vez que ele possui quatro terminas, porém dois deles são iguais.

O resistor usado neste circuito leva a tensão na entrada para zero, desde que o botão não esteja sendo pressionado. Por "empurrar a tensão para baixo" dizemos que este resistor atual como "pull down". Se colocássemos ele no 5V, ele funcionaria como "pull up".

## SKETCH 1

Abaixo, você vê o programa. Procure digitar tudo, prestando atenção nas letras quando maiúscula e quando minúscula. É muito importante que você digite todo o código, pois isso te ajudará muito no aprendizado.

```
int botao = 3;
int led = 2;

void setup(){
  pinMode(botao, INPUT);
  pinMode(led, OUTPUT);
}

void loop(){
  if(digitalRead(botao) == 1){
    digitalWrite(led, HIGH);
  }
  else{digitalWrite(led, LOW);}
}
```

## ANÁLISE DO CÓDIGO

Vamos analisar o código linha por linha.

```
int botao = 3;
int led = 2;
```

Nestas duas linhas declaramos duas variáveis do tipo inteiro. Vamos sempre proceder dessa maneira, declarando no início do código todas as portas que iremos utilizar.

```
void setup(){
  pinMode(botao, INPUT);
  pinMode(led, OUTPUT);
}
```

Na configuração (setup), usamos as variáveis "botao" e "led" criadas anteriormente para determinar qual porta irá atuar como entrada e qual irá atuar como saída. Note que a porta na qual conectaremos o LED é a porta 2, por isso atribuímos o valor 2 para a variável "led". Como saída, temos a porta digital 3, por isso "botao" recebe o valor 3.

No futuro veremos que se as variáveis fossem colocadas dentro de alguma função, como dentro da função setup, elas seriam variáveis locais, isto é, só funcionaria dentro da função setup e não dentro da função loop. Como declaramos as variáveis fora das funções, elas são ditas variáveis globais, pois pode ser usada dentro de qualquer função.

```
void loop(){
  if(digitalRead(botao) == 1){
    digitalWrite(led, HIGH);
  }
  else{digitalWrite(led, LOW);}
}
```

Dentro do loop, temos o if. Dentro do if, o Arduino lê a porta digital onde está o botão (digitalRead(botao)). O duplo símbolo de igual serve para comparar, verificando se o que é lido é igual à HIGH, ou 1 (digitalRead(botao) == 1). Se isso for verdadeiro, ou seja, se o botão estiver sendo pressionado, então o LED será aceso (digitalWrite(led, HIGH);).

Caso contrário (else), o LED será desligado: digitalWrite(led, LOW).

## MEDINDO TEMPO DE REAÇÃO

Vamos modificar o programa para medir o tempo de reação. A ideia esperar um tempo aleatório, ascender o LED e medir o tempo até que o usuário aperte o botão. Vamos usar o circuito anterior, apresentando o resultado do tempo de reação no monitor serial. Faremos como anteriormente: apresentaremos o código e depois analisaremos linha por linha.

## SKETCH 2

```
int led = 2, botao = 3;
int espera;
int tempo_inicial, tempo_final;

void setup() {
  pinMode(led,OUTPUT);
  pinMode(botao,INPUT);
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

void loop() {
  digitalWrite(led,LOW);
  Serial.println("Pressione o botão para
  iniciar");
  Serial.println("");
  Serial.println("");
  while(digitalRead(botao) == 0){
    //fica esperando apertar o botão
    //para iniciar o jogo
    digitalWrite(led,HIGH);
    delay(250);
    digitalWrite(led,LOW);
    delay(250);
```

```
}
  espera=random(500, 5000);
  delay(espera);
  digitalWrite(led,HIGH);
  tempo_inicial=millis();
  while(digitalRead(botao) == 0){
    //fica esperando apertar o botão
    //para saber o tempo de reação
  }
  digitalWrite(led,LOW);
  tempo_final=millis();
  Serial.print("O tempo de reação foi de:
  ");
  Serial.println(tempo_final-
  tempo_inicial);
  Serial.println("");
  delay(1000);
}
```

## ANÁLISE DO CÓDIGO

```
int led = 2, botao = 3;
int espera;
int tempo_inicial, tempo_final;
```

Nas três primeiras linhas foram declaradas as variáveis globais que ficarão disponíveis para todas as funções.

```
void setup() {
  pinMode(led,OUTPUT);
  pinMode(botao,INPUT);
  Serial.begin(9600);
  randomSeed(analogRead(0));
}
```

No setup, informamos quais são as portas de saída e entrada, respectivamente, nas duas primeiras linhas. Na terceira linha, após o colchete, iniciamos a comunicação serial com uma taxa de 9600 bits por segundo. A quarta linha tem uma novidade: usaremos uma função chamada random para gerar números pseudo-aleatórios. Um computador não é capaz de gerar números aleatórios: o que temos é na verdade uma lista muito grande de números. Quando pedimos para o computador gerar um número aleatório, ele seleciona um número nesta lista. Geralmente, o número selecionado é sempre sequencial, começando pelo primeiro, porém podemos "sortear" um número nesta lista usando randomSeed(analogRead(0)). Note que usamos o valor da entrada do sinal em uma porta analógica desconectada, uma vez que sempre haverá um ruído nesta porta. Isso permite simular razoavelente bem um número aleatório, pois selecionamos um número na lista que existe na memória do computador. Esta velor lido na porta analógica é chamado de semente, por isso Seed.

```
void loop() {
  digitalWrite(led,LOW);
  Serial.println("Pressione o botão para
  iniciar");
  Serial.println("");
  Serial.println("");
```

No loop não temos muita novidade no início: o LED conectado na porta 2 é desligado (isso foi feito para garantir que o LED sempre inicie o loop desligado, pois o programa apresentou alguns problemas no num versão sem esta função). Na degunda linha, depois do colchete, imprimimos no monitor serial a frase “Pressione o botão para iniciar”, levando o cursor para a linha de baixo. Depois, pula-se mais duas linhas e, finalmente, entranos em um novo laço.

```
  while(digitalRead(botao) == 0){
    //fica esperando apertar o botão
    //para saber o tempo de reação
    digitalWrite(led,HIGH);
    delay(250);
    digitalWrite(led,LOW);
    delay(250);
}
```

O laço while garante que o que está entre colchetes seja executado se o que estiver dentro do parêntesis for verdadeiro. Note que no caso do nosso programa o que está entre colchetes fica sendo executado enquanto ninguém pressionar o botão na porta 3, pois se ninguém pressionar o botão então digitalRead(botao) será igual a zero e se alguém pressionar o botão, então digitalRead(botao) fica igual a um e o loop será finalizado.

Enquanto ninguém pressionar o botão, o LED fica piscando. Quando pressionado, o programa sai do loop.

```
  espera=random(500, 5000);
  delay(espera);
  digitalWrite(led,HIGH);
  tempo_inicial=millis();
```

Na sequência, um número aleatório entre 500 e 5000 será gerado para que o programa espere por um tempo entre 0,5 segundo e 5 segundos. Passado este tempo, o LED acende e o usuário deverá pressionar o botão. Quando o Arduino é ligado, um cronômetro muito preciso é ligado, e o valor deste cronômetro pode ser acessado pela função millis(), ou seja, a variável tempo_inicial recebe o valor do cronômetro no instante atual.

```
  while(digitalRead(botao) == 0){
    //fica esperando apertar o botão
    //para saber o tempo de reação
   }
```

O próximo loop fica esperando o usuário pressionar o botão. Quando o botão for pressionado, o programa sai do loop while.

```
  digitalWrite(led,LOW);
  tempo_final=millis();
```

Após sair do programa, o LED será apagado e o tempo atual do cronômetro interno do Arduino é salvo na variável tempo_final.

Note que o tempo de reação do usuário será tempo_final – tempo_incial.

```
  Serial.print("O tempo de reação foi de:
  ");
  Serial.println(tempo_final-
  tempo_inicial);
  Serial.println("");
  delay(1000);
}
```

Finalmente, imprime-se no monitor serial a frase “O tempo de reação foi de: ” e, ainda na mesma linha, imprime o valor do tempo de reação que é igual a tempo_final – tempo_incial.

No final do programa, esperamos mais um segundo apenas para não correr o risco de iniciarmos o ciclo novamente. Passado este segundo, retornamos ao início do programa e o Arduino fica esperando para que o botão seja pressionado novamente.

Se julgar necessário, veja o vídeo que o seu professor postou no site. Ele começa com o funcionamento do programa.