

Universidade Tecnológica Federal do Paraná – Toledo
Engenharia da Computação – COENC

Sistemas Embarcados

Utilizando IDE Arduino **Interfaceando chaves mecânicas** **Debounce por software**

Tiago Piovesan Vendruscolo



Esta licença permite que outros remixem, adaptem e criem a partir do trabalho para fins não comerciais, desde que atribuam o devido crédito aos autores originais. [4.0 international](https://creativecommons.org/licenses/by-nc-nd/4.0/)

Interfaceando chaves mecânicas

Vin: alimentação externa (em conexão com o conector jack), a tensão deve ser de 7 até 12 V. Os pinos 3.3 e 5V possuem reguladores.

Entradas analógicas dos conversores AD

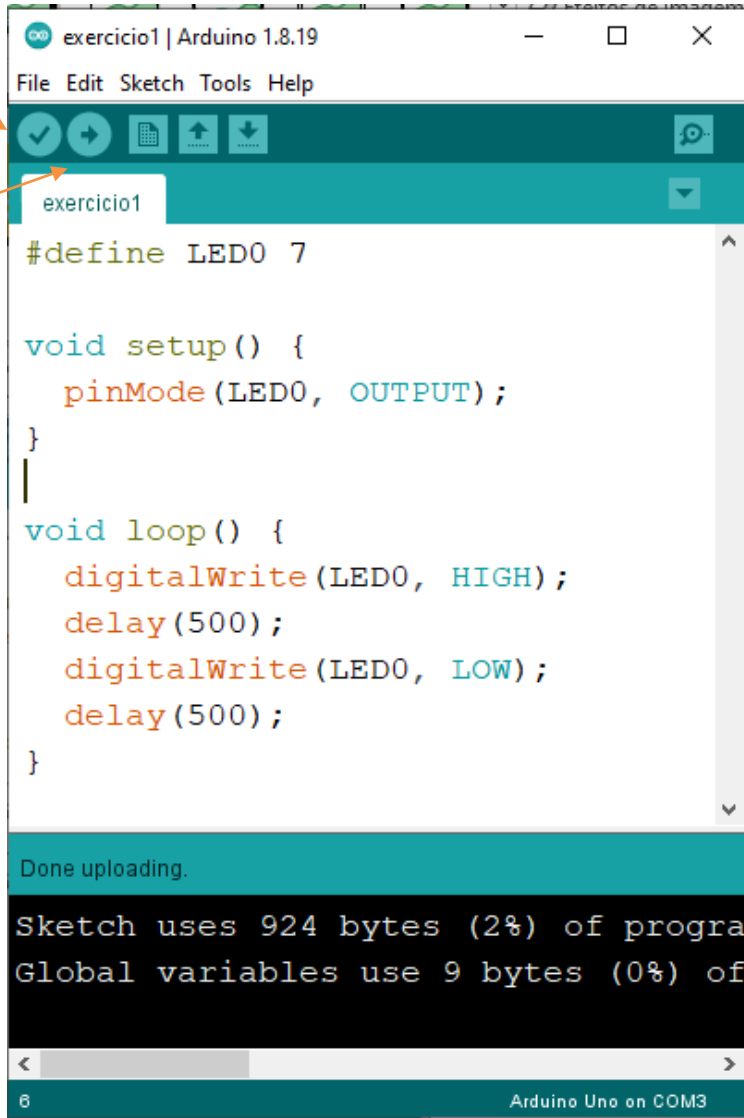


Interfaceando chaves mecânicas

- Exercício 1a: Faça um código que faça um LED no pino 7 piscar em 1Hz.

Compila

Grava



```
exercicio1 | Arduino 1.8.19
File Edit Sketch Tools Help

#define LED0 7

void setup() {
  pinMode(LED0, OUTPUT);
}

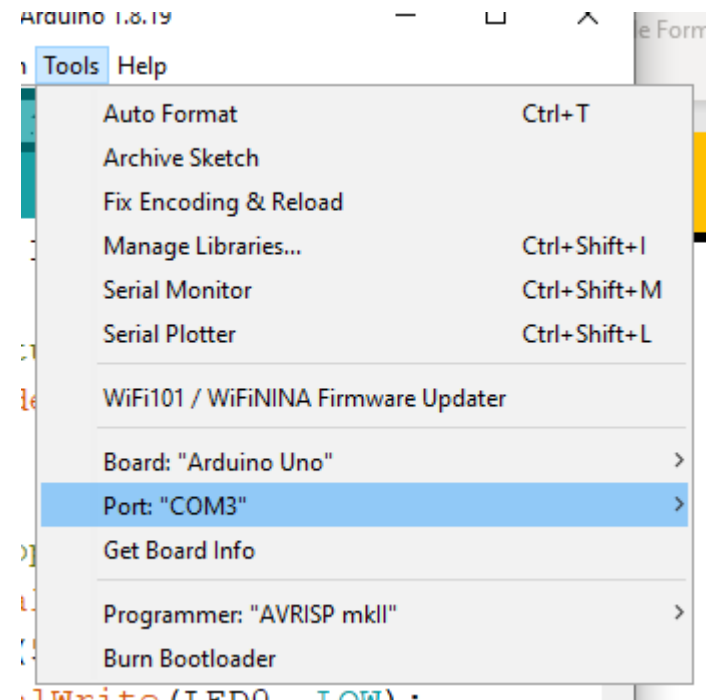
void loop() {
  digitalWrite(LED0, HIGH);
  delay(500);
  digitalWrite(LED0, LOW);
  delay(500);
}

Done uploading.

Sketch uses 924 bytes (2%) of program memory.
Global variables use 9 bytes (0%) of dynamic memory.

6 Arduino Uno on COM3
```

Verificar em Tools a board (Arduino UNO) e em Port a COM em que a placa está conectada



Interfaceando chaves mecânicas

- Exercício 1a: Faça um código que faça um LED no pino 7 piscar em 1Hz.

Definição de nomes para os pinos de entrada e saída

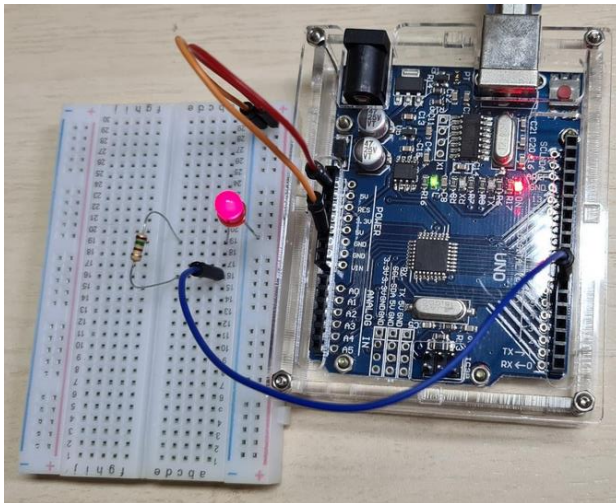
```
#define LED0 7
```

Configuração dos pinos e periféricos

```
void setup() {  
  pinMode(LED0, OUTPUT);  
}
```

Código que será executado

```
void loop() {  
  digitalWrite(LED0, HIGH);  
  delay(500);  
  digitalWrite(LED0, LOW);  
  delay(500);  
}
```



Componentes:

1 LED

1 Resistor 150 ohm

Interfaceando chaves mecânicas

- Exercício 1b: Como otimizar o código mantendo a mesma frequência?

```
#define LED0 7

void setup() {
    pinMode(LED0, OUTPUT);
}

void loop() {
    digitalWrite(LED0, HIGH);
    delay(500);
    digitalWrite(LED0, LOW);
    delay(500);
}
```



```
#define LED0 7

void setup() {
    pinMode(LED0, OUTPUT);
}

void loop() {
    digitalWrite(LED0, !digitalRead(LED0));
    delay(500);
}
```

Interfaceando chaves mecânicas

- Exercício 1c: Refaça o código sem utilizar a função delay.
 - Função millis() retorna o tempo desde que o microcontrolador foi ligado. Ocorre overflow após 49 dias.
 - Função micros() o overflow ocorre após 70 minutos.

```
#define LED0 7

int tempo_LED_ms = 500;
unsigned long ultima_execucao = 0;

void setup() {
    pinMode(LED0, OUTPUT);
}

void loop() {
    if(millis() - ultima_execucao >= tempo_LED_ms){
        digitalWrite(LED0, !digitalRead(LED0));
        ultima_execucao = millis();
    }
}
```

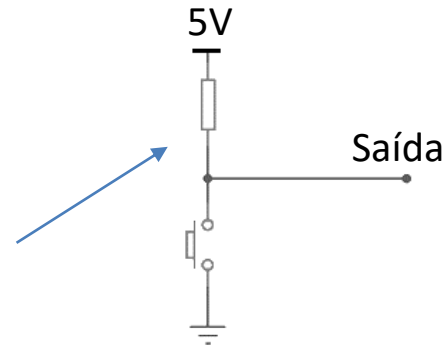
Interfaceando chaves mecânicas

- Exercício 2a: Faça um código que faça um LED no pino 7 inverter o nível a cada pressionamento do botão no pino 2. Utilize um resistor de pull up interno.

```
#define LED0 7  
#define botao 2
```

```
void setup() {  
    pinMode(LED0, OUTPUT);  
    pinMode(botao, INPUT_PULLUP);  
}
```

```
void loop() {  
    if (digitalRead(botao) == LOW) {  
        digitalWrite(LED0, !digitalRead(LED0));  
    }  
}
```



Funcionou corretamente?

Utilizando o Arduino – Serial

- Também podem ser feitas leituras ou escritas na porta de comunicação serial. Para isso, acrescente as seguintes funções no código:

```
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.println(val);
}
```

Velocidade da porta serial

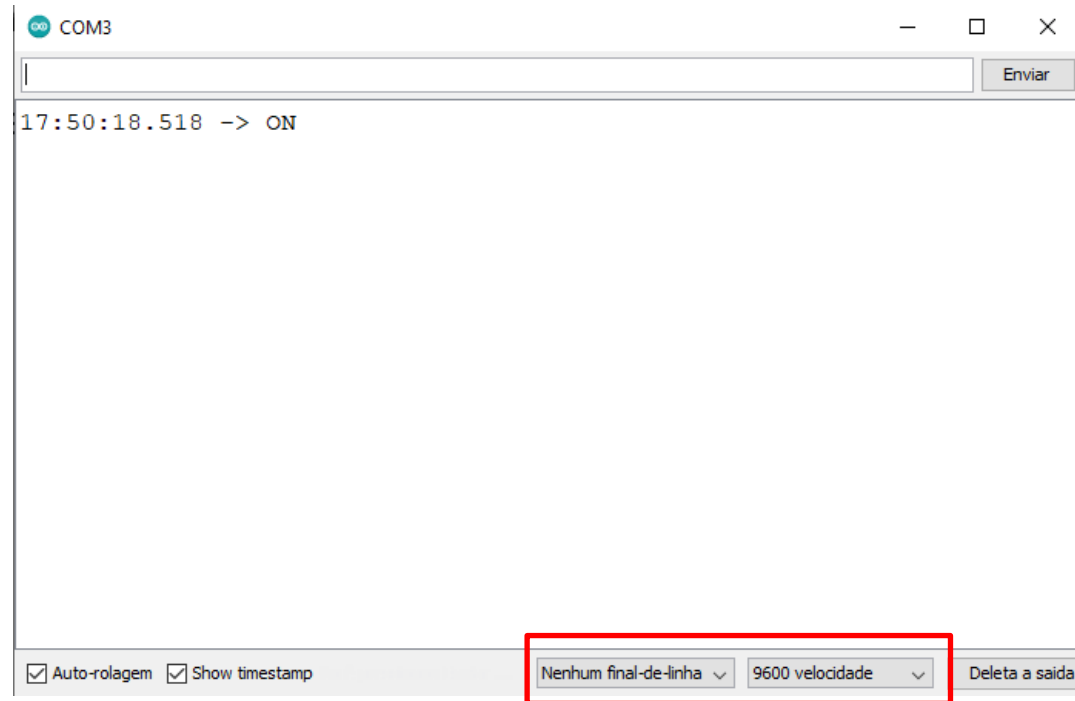
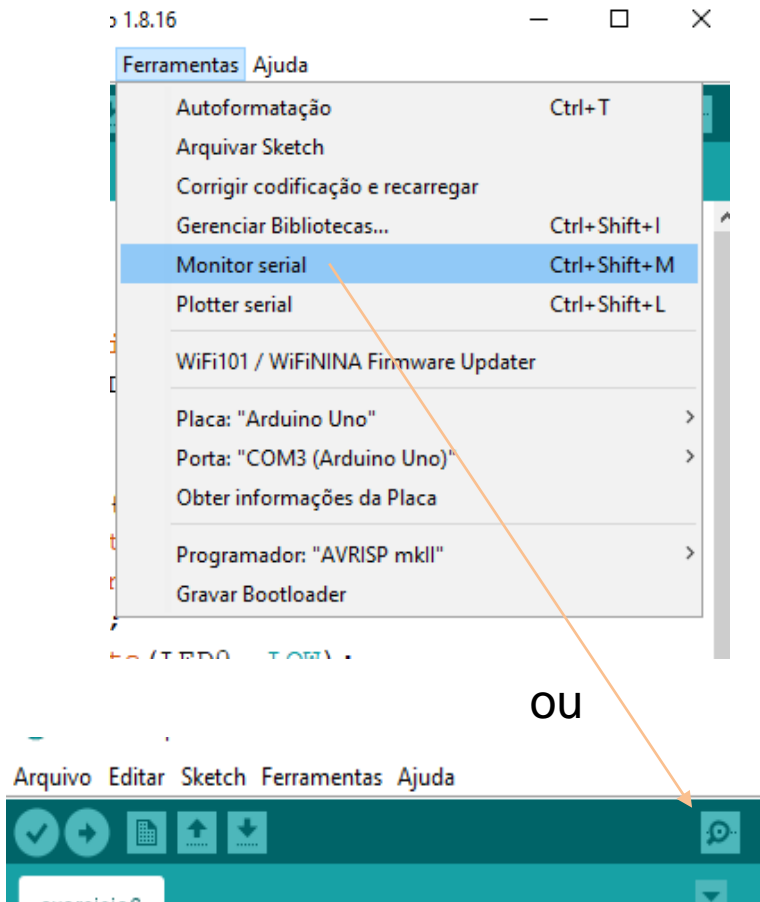


Variável impressa na serial



Utilizando o Arduino – Serial

- Para visualizar a comunicação serial, siga o caminho: Ferramentas > monitor serial
 - *Configure a velocidade de acordo com a colocada em `Serial.begin(...)`.*



Interfaceando chaves mecânicas

- Exercício 2b: Faça um código que faça um LED no pino 7 inverter o nível a cada pressionamento do botão no pino 2. Utilize um resistor de pull up interno, faça a contagem de cada pressionamento e imprima na serial.

```
#define LED0 7
#define botao 2

int contagem = 0;

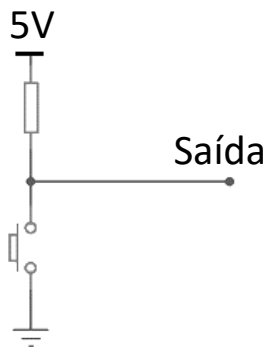
void setup() {
    pinMode(LED0, OUTPUT);
    pinMode(botao, INPUT_PULLUP);
    Serial.begin(9600);
}

void loop() {
    if (digitalRead(botao) == LOW) {
        contagem++;
        digitalWrite(LED0, !digitalRead(LED0));
        Serial.println(contagem);
    }
}
```

Interfaceando chaves mecânicas

- Atenuação de ruído via Software.

1. *Delay*: A forma mais simples é acrescentando um delay após a primeira leitura de um pulso na entrada, assim, os próximos pulsos (ruídos) serão ignorados pelo microcontrolador.
 - Problema: Se manter o botão pressionado, o software reconhecerá vários pressionamentos.
2. *While*: “Prende” a execução enquanto o nível do botão não altera.
 - Problema: Se manter o botão pressionado, o software ficará travado.



```
if (digitalRead(botao) == LOW) {  
    while(digitalRead(botao) == LOW) {}  
    delay(100);  
}
```

Desvantagem extra dos dois métodos anteriores: Qualquer ruído (EMI – interferência eletromagnética) na porta acaba sendo detectado como o pressionamento da chave.

Interfaceando chaves mecânicas

- Exercício 2c: Refaça o exercício 2b, utilizando while e delay. Faça primeiro apenas com o while e depois com o while e delay. Teste a contagem na serial.

```
#define LED0 7
#define botao 2

int contagem = 0;

void setup() {
    pinMode(LED0, OUTPUT);
    pinMode(botao, INPUT_PULLUP);
    Serial.begin(9600);
}

void loop() {
    if (digitalRead(botao) == LOW) {
        while(digitalRead(botao) == LOW) {}
        delay(100);
        contagem++;
        digitalWrite(LED0, !digitalRead(LED0));
        Serial.println(contagem);
    }
}
```

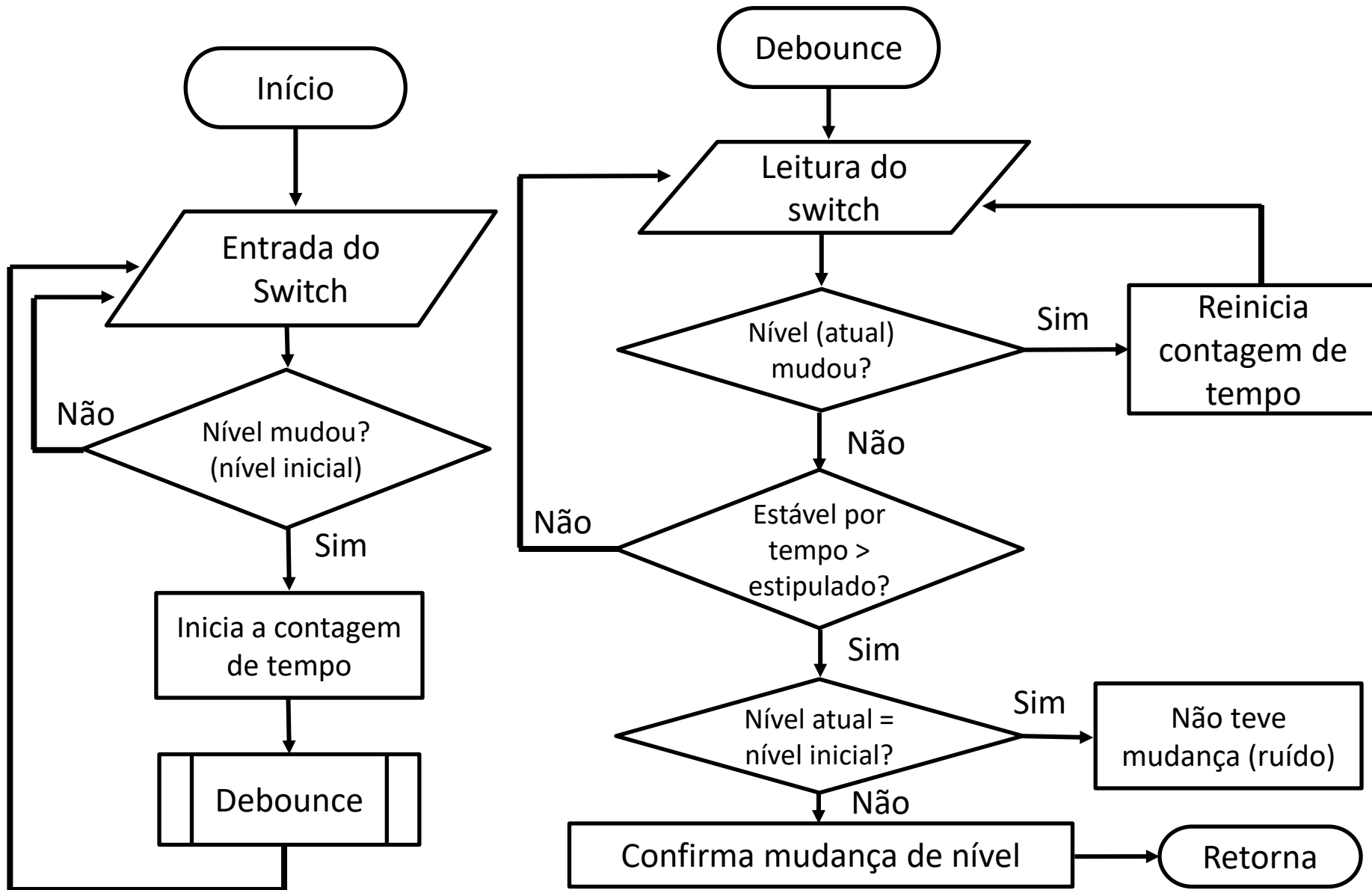
Interfaceando chaves mecânicas

- Sinal provável ao utilizar uma chave mecânica:



- Modelo Debounce: Utilizar contagem de tempo enquanto analisa os sinais. Ao mudar de nível, o sinal precisa ficar estável por um tempo maior que o estipulado pelo projetista.

Interfaceando chaves mecânicas



Interfaceando chaves mecânicas

- Exemplo 1: Faça um Código em que o nível do LED altera cada vez que o botão é pressionado.

```
#define LED0 7
#define botao 2

//Variaveis para analise do switch
bool nivel_passado_botao = HIGH;    // nivel passado de entrada do botao
bool nivel_antes_debounce_botao = HIGH;    // nivel antes de entrar na rotina

// variaveis para contagem de tempo
// ultimo valor de tempo (em millis) em que ocorreu o ultimo pressionamento (ou ruído) no botao
unsigned long ultima_mudanca_switch_entrada = 0;
// delay ate estabilizar o pressionamento (quando pressiona e quando solta)
int debounce_Delay = 100;

void setup() {
    pinMode(botao, INPUT_PULLUP);
    pinMode(LED0, OUTPUT);
}
```

Arquivo no MOODLE

Interfaceando chaves mecânicas

Modifique a variável `debounce_Delay` e analise o comportamento

```
void loop() {  
  // leitura do estado atual do botao  
  bool leitura_atual = digitalRead(botao);  
  // verifica se o estado do botao alterou (pode ser por pressionamento ou por ruído)  
  if (leitura_atual != nivel_antes_debounce_botao) {  
    // reseta a variavel de tempo  
    ultima_mudanca_switch_entrada = millis();  
  }  
  
  // atualiza o ultimo nivel com a leitura atual  
  nivel_antes_debounce_botao = leitura_atual;  
  
  // O sinal de entrada devera ficar estavel por no minimo "debounce_Delay"  
  if ((millis() - ultima_mudanca_switch_entrada) > debounce_Delay) {  
    // caso o switch tenha alterado o nivel, ele atualiza a variavel  
    if (leitura_atual != nivel_passado_botao) {  
      nivel_passado_botao = leitura_atual;  
      // altera o estado do LED quando pressiona o botão (LOW)  
      if (nivel_passado_botao == LOW) {  
        digitalWrite(LED0, !digitalRead(LED0));  
        //Coloque aqui o que deve acontecer/setar flags.  
      }  
    }  
  }  
}
```

Se alterar para HIGH, ele irá inverter o nível do LED quando soltar o botão.

Nota: Também pode-se utilizar o periférico TIMER

Interfaceando chaves mecânicas

- Exercício 3: Refaça o código anterior de forma que o LED mude seu estado a cada 4 pressionamentos do botão. Imprima a contagem na serial.

Interfaceando chaves mecânicas

```
#define LED0 7
#define botao 2

//Variaveis para analise do switch
bool nivel_passado_botao = HIGH;    // nivel passado de entrada do botao
bool nivel_antes_debounce_botao = HIGH;    // nivel antes de entrar na rotina

// variaveis para contagem de tempo
// ultimo valor de tempo (em millis) em que ocorreu o ultimo pressionamento (ou ruído) no botao
unsigned long ultima_mudanca_switch_entrada = 0;
// delay ate estabilizar o pressionamento (quando pressiona e quando solta)
int debounce_Delay = 100;
int contagem = 4;

void setup() {
    pinMode(botao, INPUT_PULLUP);
    pinMode(LED0, OUTPUT);
    Serial.begin(9600);
}

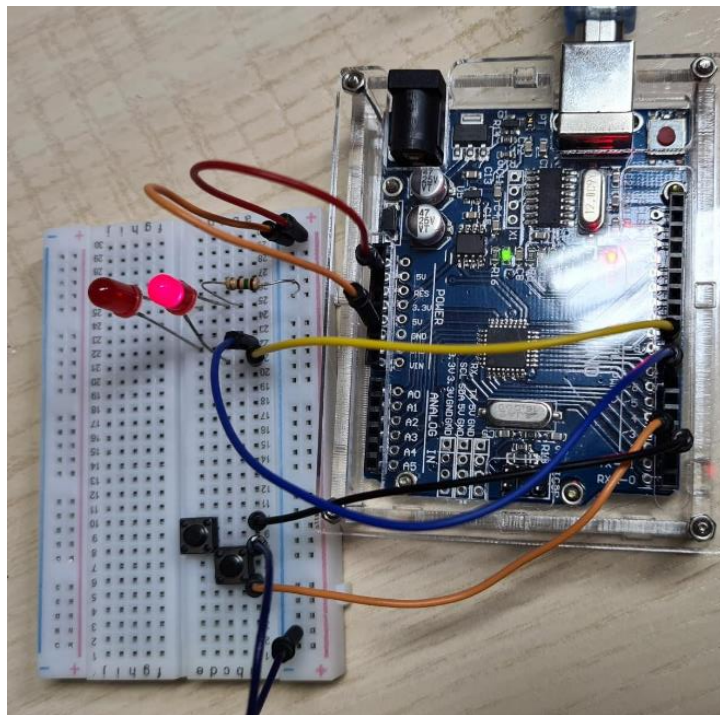
void loop() {
    // leitura do estado atual do botao
    bool leitura_atual = digitalRead(botao);
    // verifica se o estado do botao alterou (pode ser por pressionamento ou por ruído)
    if (leitura_atual != nivel_antes_debounce_botao) {
        // reseta a variavel de tempo
        ultima_mudanca_switch_entrada = millis();
    }

    // atualiza o ultimo nivel com a leitura atual
    nivel_antes_debounce_botao = leitura_atual;

    // O sinal de entrada devera ficar estavel por no minimo "debounce_Delay"
    if ((millis() - ultima_mudanca_switch_entrada) > debounce_Delay) {
        // caso o switch tenha alterado o nivel, ele atualiza a variavel
        if (leitura_atual != nivel_passado_botao) {
            nivel_passado_botao = leitura_atual;
            // altera o estado do LED quando pressiona o botão (LOW)
            if (nivel_passado_botao == LOW) {
                contagem = contagem-1;
                Serial.println(contagem);
                //Coloque aqui o que deve acontecer/setar flags.
            }
        }
    }
    if(contagem==0){
        digitalWrite(LED0, !digitalRead(LED0));
        contagem=4;
    }
}
```

Interfaceando chaves mecânicas

- Utilizando N chaves:
 - O contador de tempo (debounce) pode ser o mesmo para todas as chaves, no entanto, qualquer ruído que acontecer em uma chave irá afetar o debounce das outras, então o ideal é parametrizar.
- Passos:
 - Parametrizar as variáveis/contadores
- Exemplo 2: Refaça o exemplo 1 com 2 botões e 2 LEDs, sendo que cada botão controle um LED individualmente.



Arquivo no MOODLE

```
//Parametrizacao para a quantidade de botoes e LEDs
#define QUANTIDADE_BOTOES 2
const int botao[2] = {2, 3}; // Pinos especificos, nao é possivel parametrizar
const int LED[2] = {7, 8};

//Declaracao das variaveis parametrizadas
bool nivel_passado_botao[QUANTIDADE_BOTOES] = {0}; // nivel atual de entrada do botao[i]
bool nivel_antes_debounce_botao[QUANTIDADE_BOTOES] = {0}; // nivel lido anteriormente no botao[i]
bool leitura_atual[QUANTIDADE_BOTOES]={0};
unsigned long ultima_mudanca_botao_entrada[QUANTIDADE_BOTOES] = {0};
int debounce_Delay[QUANTIDADE_BOTOES] = {0};
```

Interfaceando chaves mecânicas

```
void setup() {
  for (int i = 0; i < QUANTIDADE_BOTOES; i++){
    //Inicializacao das variaveis parametrizadas
    nivel_passado_botao[i] = HIGH;      // nivel atual de entrada do botao[i]
    nivel_antes_debounce_botao[i] = HIGH;  // nivel lido anteriormente no botao[i]
    leitura_atual[i]=HIGH;
    ultima_mudanca_botao_entrada[i] = 0;
    debounce_Delay[i] = 100;

    pinMode(botao[i], INPUT_PULLUP);
    pinMode(LED[i], OUTPUT);
  }
}

void loop() {
  for (int i = 0; i < QUANTIDADE_BOTOES; i++){
    // leitura do estado atual do botao[i]
    leitura_atual[i] = digitalRead(botao[i]);
    if (leitura_atual[i] != nivel_antes_debounce_botao[i]) {
      ultima_mudanca_botao_entrada[i] = millis();
    }
    nivel_antes_debounce_botao[i] = leitura_atual[i];
    // O sinal de entrada devera ficar estavel por no minimo "debounce_Delay[i]"
    if ((millis() - ultima_mudanca_botao_entrada[i]) > debounce_Delay[i]) {
      // caso o botao tenha alterado o nivel, ele atualiza a variavel
      if (leitura_atual[i] != nivel_passado_botao[i]) {
        nivel_passado_botao[i] = leitura_atual[i];

        // altera o estado do LED quando pressiona o botão (LOW)
        if (nivel_passado_botao[i] == LOW) {
          digitalWrite(LED[i], !digitalRead(LED[i]));
        }
      }
    }
  }
}
```

Interfaceando chaves mecânicas

- Exercício 4: Refaça o exemplo 2 de forma que o estado de cada LED altere após o respectivo botão ser pressionado 4 vezes (exercício 3). Após, também inclua a impressão na serial da contagem de cada botão.

Interfaceando chaves mecânicas

```
//Parametrizacao para a quantidade de botoes e LEDs
#define QUANTIDADE_BOTOES 2
const int botao[2] = {2, 3}; // Pinos especificos, nao é possivel parametrizar
const int LED[2] = {7, 8};

//Declaracao das variaveis parametrizadas
bool nivel_passado_botao[QUANTIDADE_BOTOES] = {0}; // nivel atual de entrada do botao[i]
bool nivel_antes_debounce_botao[QUANTIDADE_BOTOES] = {0}; // nivel lido anteriormente no botao[i]
bool leitura_atual[QUANTIDADE_BOTOES]={0};
unsigned long ultima_mudanca_botao_entrada[QUANTIDADE_BOTOES] = {0};
int debounce_Delay[QUANTIDADE_BOTOES] = {0};
int contagem[QUANTIDADE_BOTOES] = {0};

void setup() {
for (int i = 0; i < QUANTIDADE_BOTOES; i++){
    //Inicializacao das variaveis parametrizadas
    nivel_passado_botao[i] = HIGH; // nivel atual de entrada do botao[i]
    nivel_antes_debounce_botao[i] = HIGH; // nivel lido anteriormente no botao[i]
    leitura_atual[i]=HIGH;
    ultima_mudanca_botao_entrada[i] = 0;
    debounce_Delay[i] = 100;
    contagem[i] = 4;
    pinMode(botao[i], INPUT_PULLUP);
    pinMode(LED[i], OUTPUT);
    Serial.begin(9600);
}
}
```

Interfaceando chaves mecânicas

```
void loop() {
  for (int i = 0; i < QUANTIDADE_BOTOES; i++){
    // leitura do estado atual do botao[i]
    leitura_atual[i] = digitalRead(botao[i]);
    if (leitura_atual[i] != nivel_antes_debounce_botao[i]) {
      ultima_mudanca_botao_entrada[i] = millis();
    }
    nivel_antes_debounce_botao[i] = leitura_atual[i];
    // O sinal de entrada devera ficar estavel por no minimo "debounce_Delay[i]"
    if ((millis() - ultima_mudanca_botao_entrada[i]) > debounce_Delay[i]) {
      // caso o botao tenha alterado o nivel, ele atualiza a variavel
      if (leitura_atual[i] != nivel_passado_botao[i]) {
        nivel_passado_botao[i] = leitura_atual[i];

        // altera o estado do LED quando pressiona o botão (LOW)
        if (nivel_passado_botao[i] == LOW) {
          contagem[i]--;
          Serial.println("Contagem do botao " + String(i) + ": " + String(contagem[i]));
        }
      }
      if(contagem[i]==0){
        digitalWrite(LED[i], !digitalRead(LED[i]));
        contagem[i]=4;
      }
    }
  }
}
```


Próxima aula

Uso de periféricos de microcontroladores
Sleep Mode