Aluno do Embarcatech_37 no IFMA

Nome: Manoel Felipe Costa Furtado

Matrícula: 20251RSE.MTC0086

Atividade – Referente ao capítulo 07 da unidade 01

Tema do Capítulo – PIO - Programmable Input/Output

Prazo dia 15/06/2025 as 23:59

Enunciado: Complementação do Projeto NeoControlLab Utilizando uma técnica de interrupção em conjunto com o botão A, refaça o código de modo que a geração de números aleatórios ocorrerá quando pressionado o botão A.

Qual a melhoria que deve ser realizada no novo projeto:

- Utilizando uma técnica de interrupção em conjunto com o botão A, refaça o código de modo que a geração de números aleatórios ocorrerá quando pressionado o botão A.
- Atualmente, a geração dos números aleatórios está vinculada ao iniciar do BitDogLab, tornar funcional ao pressionar o botão A e gerar uma sequência de números aleatórios.

- Nome do arquivo principal: “Atividade_cap_07.c” → Na pasta src.
- Vídeo mostrando o seu funcionamento
  Link: https://youtu.be/lyEr6pB4Kwc
- GitHub:
  https://github.com/ManoelFelipe/Embarcatech_37/tree/main/Unidade_01/Cap_07/Atividade_07

Organização do projeto: Código foi modularizado.

Na Pasta src:

- Atividade_cap_07.c
  Arquivo principal com main() – Onde está desenvolvida a exigência da atividade.
- numeros_neopixel.c e numeros_neopixel.h
  Sua única tarefa é desenhar os números de 1 a 6.
- efeito_curva_ar.c e efeito_curva_ar.h
  (Demonstra a capacidade do projeto de ir além de imagens estáticas. Ele implementa uma animação complexa de uma curva que se move, baseada em um modelo matemático (autorregressivo))
- testes_cores.c e testes_cores.h
  Fornece rotinas para verificar se a matriz de LEDs está funcionando corretamente.

Na Pasta libs:

Biblioteca: LabNeoPixel

- neopixel_driver.c e neopixel_driver.h
  Esta é a camada mais baixa do software. Ela "esconde" a complexidade de controlar os LEDs. Ela lida diretamente com o periférico PIO (I/O Programável) do RP2040 para gerar os sinais de temporização exatos que o protocolo NeoPixel (WS2812B) exige.
- efeitos.c e efeitos.h
  Fornece uma biblioteca de animações prontas, como efeitoEspiral e efeitoOndaVertical. Também inclui funções auxiliares como acenderFileira e acenderColuna.
- util.c e util.h
  Oferece funções genéricas que poderiam ser usadas em qualquer parte do projeto, como a geração de números aleatórios. (No seu arquivo principal, você implementou uma função própria, sorteia_entre, mas este módulo existe como uma biblioteca de utilidades separada).

Benefícios do código modularizado:

- Main enxuta: A função principal fica clara e fácil de entender.
- Modularidade: Cada funcionalidade está separada em seu próprio módulo.
- Reusabilidade: Os módulos podem ser usados em outros projetos.
- Manutenibilidade: Fácil de fazer alterações em áreas específicas.
- Testabilidade: Cada módulo pode ser testado isoladamente.

Como entender o projeto:

1. Comece pelo main() em Atividade_cap_07.c para ver a ordem em que tudo acontece.
2. Vá abrindo os headers .h para saber o que cada módulo oferece.
3. Leia os .c se quiser entender os detalhes internos.
4. Todos os arquivos estão comentados com Doxygen em português para facilitar.

- Código: src/Atividade_02_.c

```c
/**
 * @file Atividade_cap_07.c
 * @brief Versão 3: Arquivo principal com interrupção e tratamento de debounce robusto.
 *
 * Este programa demonstra o uso da matriz de LEDs NeoPixel 5x5 (LabNeoPixel).
 * Ele aguarda o pressionar do Botão A (GPIO 5). Quando uma interrupção é detectada,
 * o programa utiliza uma técnica de debounce robusta (inspirada na Atividade 4)
 * para confirmar o pressionamento. Se confirmado, gera uma sequência de números
 * aleatórios e os exibe na matriz.
 *
 * @author Modificado por Manoel Furtado
 * @date 12 de Junho de 2025
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "pico/stdlib.h"
#include "pico/time.h"
#include "hardware/gpio.h" // Necessário para interrupções de GPIO
#include "hardware/sync.h" // Para funções de sincronização como __wfi()

#include "libs/LabNeoPixel/neopixel_driver.h"
#include "src/numeros_neopixel.h"
#include "src/testes_cores.h"
#include "libs/LabNeoPixel/efeitos.h"
#include "src/efeito_curva_ar.h"


// === Definições de hardware e debounce ===
#define BOTAO_A 5 // Botão A conectado ao GPIO5
/**
 * @brief Tempo de debounce em milissegundos.
 * O programa aguardará este tempo após a detecção inicial da interrupção
 * para que o ruído elétrico do botão se estabilize.
 */
#define DEBOUNCE_MS 50
```

```c
/**
 * @brief Flag volátil para comunicação entre a interrupção e o loop principal.
 */
volatile bool botao_a_pressionado = false;

/**
 * @brief Função de Callback para a interrupção do GPIO com lógica de debounce.
 *
 * Esta função é chamada pelo hardware na borda de descida (pressionar do botão).
 * Passo 1: Desabilita imediatamente a interrupção para o pino do botão para evitar "bounces".
 * Passo 2: Sinaliza ao loop principal (setando a flag) que um evento precisa ser verificado.
 *
 * @param gpio O número do pino que causou a interrupção.
 * @param events O tipo de evento que ocorreu.
 */
void botao_a_callback(uint gpio, uint32_t events) {
    // Desabilita a interrupção para este pino. O loop principal irá reabilitá-la
    // após o tratamento completo do evento.
    gpio_set_irq_enabled(BOTAO_A, GPIO_IRQ_EDGE_FALL, false);
    // Sinaliza ao loop principal que o botão foi pressionado.
    botao_a_pressionado = true;
}

/**
 * @brief Configura e inicializa o sistema, a matriz NeoPixel e a interrupção do botão.
 */
void setup() {
    // Inicializa a comunicação serial via USB.
    stdio_init_all();
    sleep_ms(1000);

    // Inicializa o driver da matriz NeoPixel.
    npInit(LED_PIN);
    // Define a semente para o gerador de números aleatórios.
    srand(time_us_32());

    // --- Configuração da Interrupção do Botão A ---
    gpio_init(BOTAO_A);
    gpio_set_dir(BOTAO_A, GPIO_IN);
    gpio_pull_up(BOTAO_A);

    // Habilita a interrupção para o pino BOTAO_A e registra a função de callback.
    gpio_set_irq_enabled_with_callback(BOTAO_A, GPIO_IRQ_EDGE_FALL, true, &botao_a_callback);
}

/**
 * @brief Gera um número inteiro aleatório dentro de um intervalo inclusivo.
 */
```

```c
int sorteia_entre(int min, int max) {
    return rand() % (max - min + 1) + min;
}

/**
 * @brief Exibe na matriz NeoPixel o número sorteado (de 1 a 6).
 */
void mostrar_numero_sorteado(int numero) {
    switch (numero) {
        case 1: mostrar_numero_1(); break;
        case 2: mostrar_numero_2(); break;
        case 3: mostrar_numero_3(); break;
        case 4: mostrar_numero_4(); break;
        case 5: mostrar_numero_5(); break;
        case 6: mostrar_numero_6(); break;
    }
}

/**
 * @brief Função principal (ponto de entrada) do programa.
 *
 * Orquestra a execução do projeto, implementando a lógica de debounce no loop principal.
 */
int main() {
    // Executa a rotina de configuração inicial.
    setup();

    printf("NeoControlLab pronto! Pressione o Botao A para sortear um numero.\n");

    // Loop infinito que constitui o programa principal.
    while (true) {
        // Verifica se a flag foi setada pela interrupção.
        if (botao_a_pressionado) {
            // Reseta a flag imediatamente.
            botao_a_pressionado = false;

            // Passo 3: Aguarda o tempo de debounce para o ruído mecânico do botão cessar.
            sleep_ms(DEBOUNCE_MS);

            // Passo 4: Re-lê o estado do pino. Se ainda estiver pressionado, confirma o clique.
            if (!gpio_get(BOTAO_A)) {

                // --- AÇÃO PRINCIPAL (Executada somente em um clique confirmado) ---

                // Sorteia quantas vezes o "dado" será lançado.
                int vezes = sorteia_entre(100, 500);
                printf("Botao A pressionado! Mostrando %d numeros aleatorios...\n", vezes);
```

```
                int sorteado = 0;
                // Loop que simula os lançamentos do dado.
                for (int i = 1; i <= vezes; i++) {
                    int n = sorteia_entre(1, 6);
                    // Imprime o progresso e o número sorteado em uma única linha
                    printf("Sorteio %d de %d: O numero sorteado foi: %d\n", i, vezes, n);

                    mostrar_numero_sorteado(n);
                    sorteado = n;
                    sleep_ms(10);
                }

                printf("\n--- Fim da Sequencia ---\n");
                printf("Total de %d numeros sorteados. Ultimo numero sorteado: %d\n\n", vezes, sorteado);
            }

            // Passo 5: Reabilita a interrupção para o pino do botão,
            // independentemente de ter sido um clique válido ou apenas ruído.
            // O sistema está pronto para a próxima detecção.
            gpio_set_irq_enabled(BOTAO_A, GPIO_IRQ_EDGE_FALL, true);
        }

        // Coloca o processador em modo de baixo consumo até a próxima interrupção.
        __wfi();
    }

    return 0; // Esta linha nunca será alcançada.
}
```