



Residência
Tecnológica
em Sistemas
Embarcados

Interrupções

Unidade 4 | Capítulo 4

Executores:



Coordenação:



Iniciativa:



Sumário



- Objetivos
- Revisão
- Técnicas de entrada e saída
- E/S controlada por interrupções
- Interrupções no RP2040
- Interrupções do módulo GPIO
- Exemplo de Código
- Principais Pontos
- Conclusão

Objetivos

Ao final da aula de hoje, teremos alcançado os seguintes objetivos:

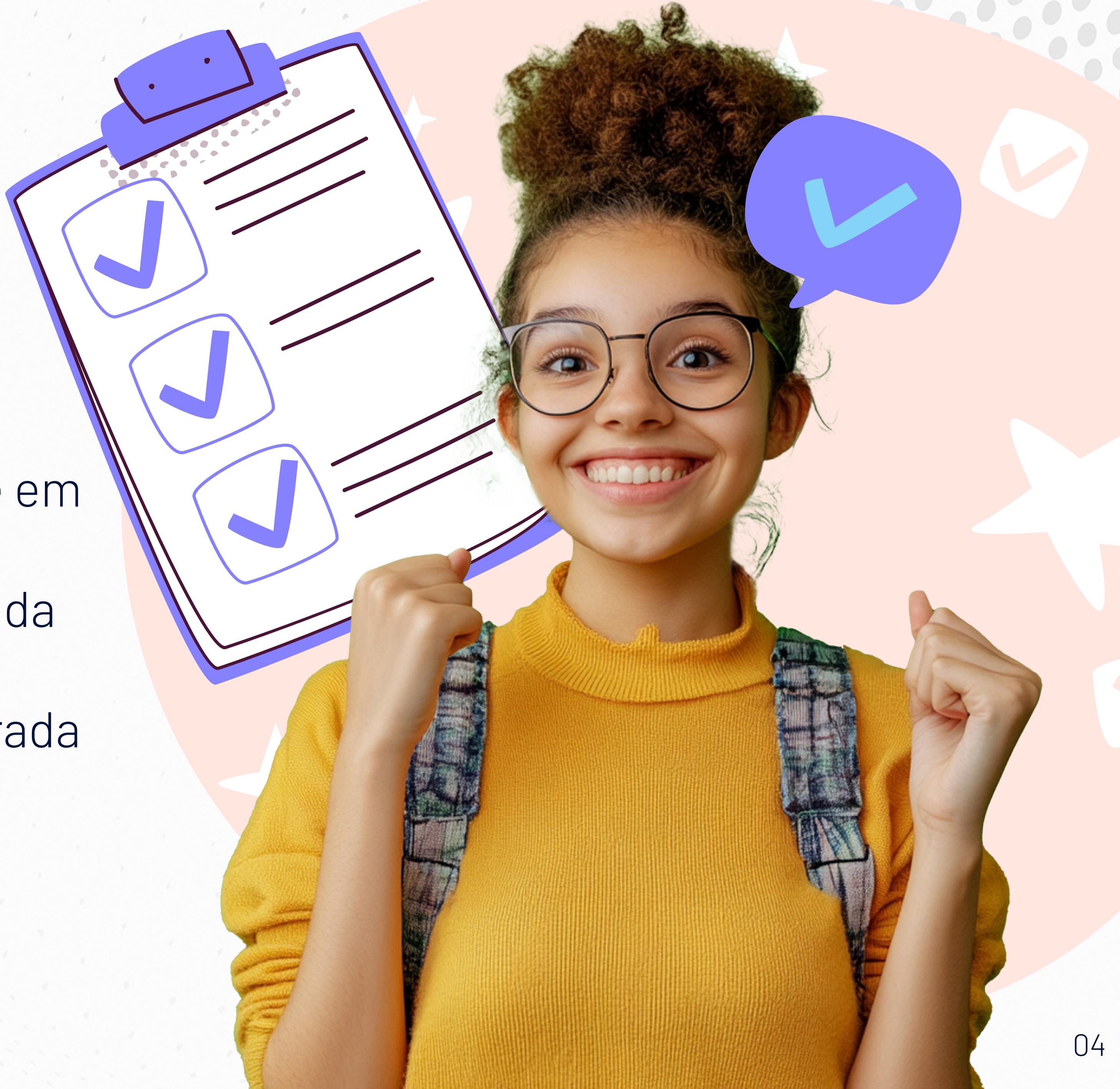
- Compreender o funcionamento do mecanismo de interrupções.
- Configurar o módulo de interrupções e rotinas de tratamento de interrupções.
- Desenvolver código de utilização de interrupções e seus diferentes modos.



Revisão

Nas últimas aulas
aprendemos:

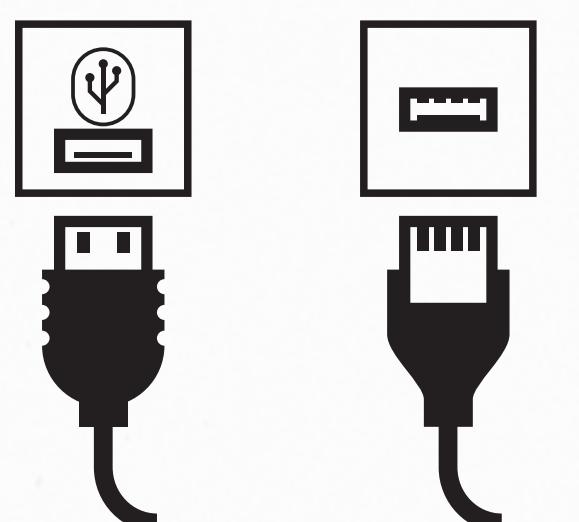
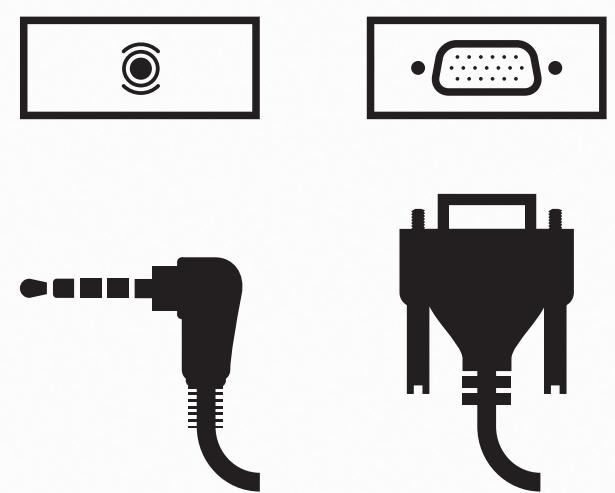
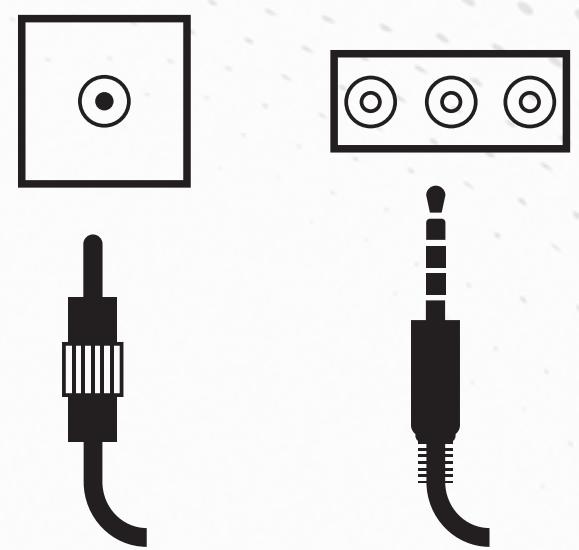
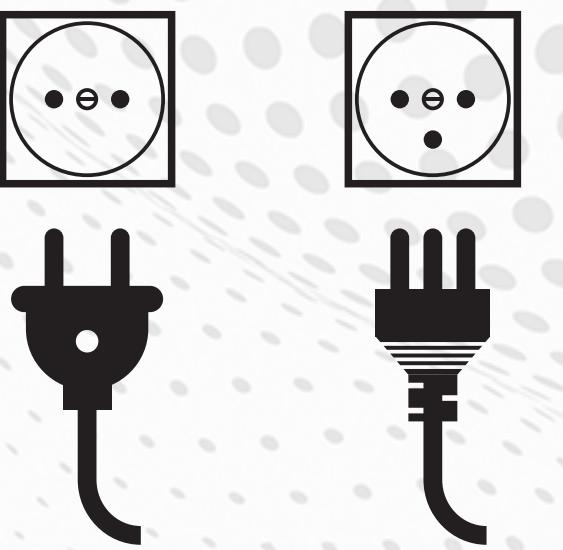
- Desenvolvimento de software em microcontroladores
 - » Entrada e Saída programada
 - * Rotinas de delay
 - » Controlar os pinos de entrada e saída (GPIO)
 - * Acionamento de LEDs
 - * Leitura de Botões



Técnicas de Entrada e Saída

E/S Programada

- Processador tem controle direto das operações de E/S
 - » Monitoramento do estado do periférico
 - * Espera por eventos
 - Processador monitora o dispositivo para identificar a ocorrência de eventos
- Envio de comandos e/ou configurações
- Transferência de dados
- Desperdício do tempo do processador



Técnicas de Entrada e Saída

E/S controlada por interrupção

- O periférico gera uma interrupção quando um evento de E/S acontece
 - » A execução do processador é interrompida, para atender à solicitação
 - * A execução é direcionada para uma rotina de tratamento de interrupções
 - * Ao fim, o processador retoma do ponto em que foi interrompido
 - » O processador não precisa monitorar periodicamente o periférico

Técnicas de Entrada e Saída

E/S controlada por interrupção



Técnicas de Entrada e Saída

Acesso Direto à Memória (DMA)

- Nas técnicas anteriores, o processador ainda se ocupa de transferir dados entre o periférico e a memória principal
- DMA
 - » Co-processador específico para controlar as transferências de dados entre periféricos e a memória
 - * Trata as interrupções dos periféricos
 - * Movimenta dados
 - » RP2040
 - * Possui um módulo DMA
 - Funciona para periféricos que geram dados
 - > Interfaces de comunicação
- configuração desse módulo não é o nosso objetivo de hoje!

Interrupções - RP2040

O microcontrolador possui um módulo ARM NVIC para cada núcleo

- 32 interrupções
 - » 26 são usadas

IRQ	Fonte	IRQ	Fonte	IRQ	Fonte	IRQ	Fonte	IRQ	Fonte
0	TIMER_IRQ_0	6	XIP_IRQ	12	DMA_IRQ_1	18	SPI0_IRQ	24	I2C1_IRQ
1	TIMER_IRQ_1	7	PIO0_IRQ_0	13	IO_IRQ_BANK0	19	SPI1_IRQ	25	RTC_IRQ
2	TIMER_IRQ_2	8	PIO0_IRQ_1	14	IO_IRQ_QSPI	20	UART0_IRQ		
3	TIMER_IRQ_3	9	PIO1_IRQ_0	15	SIO_IRQ_PROC0	21	UART1_IRQ		
4	PWM_IRQ_WRAP	10	PIO1_IRQ_1	16	SIO_IRQ_PROC1	22	ADC_IRQ_FIFO		
5	USBCTRL_IRQ	11	DMA_IRQ_0	17	CLOCKS_IRQ	23	I2C0_IRQ		

Interrupções - RP2040

Múltiplas Interrupções

- Níveis de prioridade
 - » Existem quatro níveis de prioridade
- Interrupções aninhadas
 - » Uma interrupção de menor prioridade pode ser interrompida por uma de maior prioridade
 - » Para interrupções de mesma prioridade
 - * A de menor número de IRQ vence



Interrupções do módulo GPIO

Cada pino do GPIO possui quatro interrupções:

- Níveis de prioridade
 - » GPIO_IRQ_LEVEL_LOW
 - » GPIO_IRQ_LEVEL_HIGH
 - » GPIO_IRQ_EDGE_FALL
 - » GPIO_IRQ_EDGE_RISE
 - » Normalmente, nós usamos as duas últimas que indicam
 - * Borda de Descida
 - * Borda de Subida
 - » Existe apenas uma interrupção que atende todos os eventos de GPIO
 - * De todos os pinos!

Interrupções do módulo GPIO



Funções do SDK

- O SDK fornece um tratador de interrupções
 - » gpio_irq_handler
 - * Que faz as configurações necessárias e chama uma rotina escrita pelo usuário
 - CallbackFunction
 - > Existe apenas uma para o módulo GPIO
- O programador precisa
 - » Configurar a interrupção
 - * Portas e Eventos
 - » Designar uma função de callback



Interrupções do módulo GPIO



Funções do SDK

- gpio_set_irq_enable_with_callback
 - » gpio
 - * Endereço da porta
 - » events
 - * Máscara do evento
 - * GPIO_IRQ_EDGE_RISE | GPIO_IRQ_EDGE_FALL
 - » enabled
 - * Valor booleano
 - * True - habilitado
 - * False - desabilitado
 - » callback
 - * Ponteiro para função de callback
 - * Chamada quando a interrupção ocorre

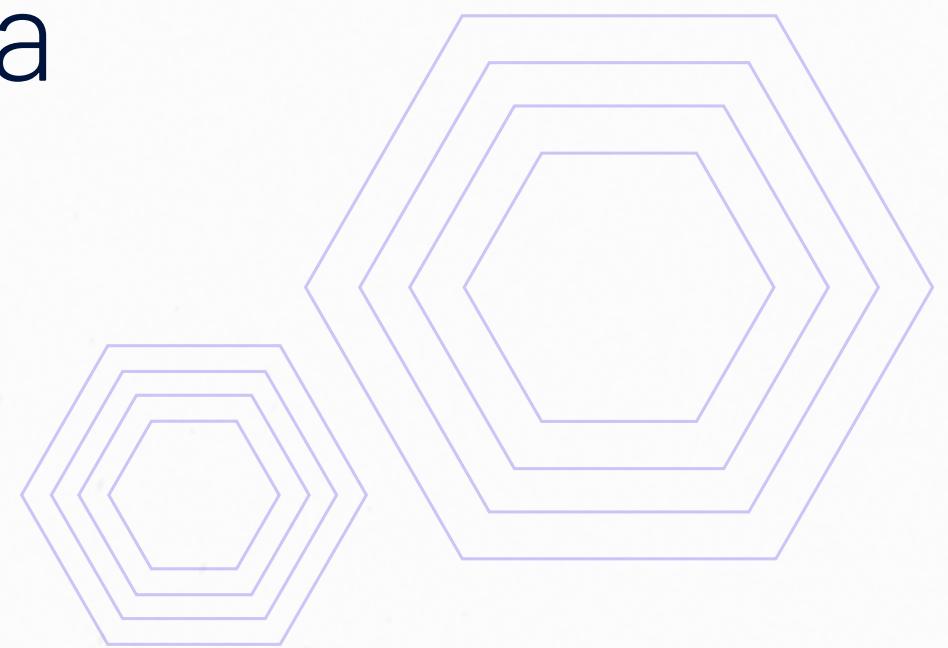


Interrupções do módulo GPIO



Orientações para funções de callback de interrupções

- Escreva funções simples e pequenas
 - » Se o evento for complexo, use um ‘flag’ para sinalizá-lo e o trate no ‘super loop’
- Cuidado com o acesso a variáveis e estruturas de dados globais
 - » Problemas de Concorrência
 - * Condições de Corrida e Starvation
 - » Variáveis devem ser declaradas como ‘volatil’
- Evite chamar funções de E/S ou muito demoradas
- Interrupções funcionam bem para eventos de baixa frequência
 - » Eventos de alta frequência podem atrapalhar a execução do seu código!



Exemplo de Código

```
#include <stdio.h>
#include "pico/stdlib.h"
const uint led_pin_red = 12;
const uint button_0 = 6;
static volatile uint a = 1;
static void gpio_irq_handler(uint gpio, uint32_t events);
int main(){
    stdio_init_all();
    gpio_init(led_pin_red);
    gpio_init(button_0);
    gpio_set_dir(button_0, GPIO_IN);
    gpio_set_dir(led_pin_red, GPIO_OUT);
    gpio_pull_up(button_0);
    gpio_set_irq_enabled_with_callback(button_0, GPIO_IRQ_EDGE_FALL, true, &gpio_irq_handler);
    while (true);
}
```

Fig. 1. Exemplo de Código.

Fonte: imagem do autor

A imagem mostra um trecho de código escrito em C para uma placa Raspberry Pi Pico. Abaixo está uma descrição detalhada do código:

Cabeçalhos incluídos: O código começa incluindo duas bibliotecas: #include <stdio.h>: usado para funções de entrada e saída padrão. #include "pico/stdlib.h": usado para interagir com o hardware da placa Pico, como GPIO (pinos de entrada/saída geral).

Declaração de constantes: const uint led_pin_red = 12;: define o pino GPIO 12 para controlar um LED vermelho. const uint button_0 = 6;: define o pino GPIO 6 para um botão.

Variável global: static volatile uint a = 1;: uma variável volátil chamada com valor inicial 1, que pode ser usada para monitoramento em intervalos.

Função de interrupção: static void gpio_irq_handler(uint gpio, uint32_t events);: função que lida com interrupções quando o botão é pressionado, mas o corpo da função ainda não está implementado.

Função main(): Configurações iniciais do hardware: stdio_init_all();: inicializa todas as interfaces de entrada/saída padrão. gpio_init(led_pin_red);: inicializa o pino do LED. gpio_init(button_0);: inicializa o pino do botão.

Configurações de direção e pull-up: gpio_set_dir(button_0, GPIO_IN);: configura o pino do botão como entrada. gpio_set_dir(led_pin_red, GPIO_OUT);: configura o pino do LED como saída.

Ativação de interruptão: gpio_set_irq_enabled_with_callback(button_0, GPIO_IRQ_EDGE_FALL, true, &gpio_irq_handler);: ativa a interrupção quando o botão é pressionado, chama a função gpio_irq_handler.

Loop infinito: while(true);: o programa entra em um loop infinito, aguardando interrupção.

Resumidamente, o código configura um botão e um LED em uma placa Raspberry Pi Pico. Quando o botão é pressionado, uma interrupção é gerada, que seria tratada pela função gpio_irq_handler.

Exemplo de Código

```
void gpio_irq_handler(uint gpio, uint32_t events)
{
    a++;
    if (a % 2 == 0)
        gpio_put(led_pin_red, true);
    else
        gpio_put(led_pin_red, false);
}
```

Fig. 2. Exemplo de Código.

Fonte: imagem do autor

A imagem mostra a implementação da função 'gpio_irq_handler', que lida com interrupções geradas quando um botão é pressionado. Aqui está uma descrição detalhada:

1. Declaração da função:
 - A função é chamada 'gpio_irq_handler' e recebe dois parâmetros:
 - 'uint gpio': o pino GPIO associado à interrupção.
 - 'uint32_t events': os eventos que causaram a interrupção.
2. Corpo da função:
 - 'a++': a variável global 'a' é incrementada sempre que a função é chamada (ou seja, toda vez que o botão é pressionado).
 - Estrutura condicional:
 - 'if (a % 2 == 0)': verifica se o valor de 'a' é par (se 'a' dividido por 2 tem resto 0).
 - Se 'a' for par, o LED vermelho (controlado pelo pino 'led_pin_red') é ligado: 'gpio_put(led_pin_red, true)'.
 - 'else': se o valor de 'a' for ímpar, o LED vermelho é desligado: 'gpio_put(led_pin_red, false)'.

Resumindo, a função alterna o estado do LED vermelho toda vez que o botão é pressionado. Se 'a' for par, o LED acende; se for ímpar, o LED apaga.

Principais Pontos

- Existem três técnicas de Entrada e Saída
- E/S controlada por interrupções é uma técnica importante para sistemas microcontrolados
 - » Agiliza o tratamento de eventos
- O módulo GPIO possui quatro eventos que geram interrupções
 - » A função `gpio_set_irq_enable_with_callback` configurar as interrupções desse módulo
- É preciso ter cuidado ao usar interrupções
 - » Erros em interrupções são muito difíceis de identificar!
 - » Escreva funções simples!

Interrupções

Unidade 4 | Capítulo 4

