Documentação da Biblioteca ButtonPi

Documentação da Biblioteca ButtonPi

Introdução

A biblioteca **ButtonPi** foi desenvolvida para simplificar o gerenciamento de botões em projetos que utilizam o Raspberry Pi Pico. Ela fornece uma interface fácil de usar para inicializar botões, ler seu estado e configurar funções de callback que são executadas quando o botão é pressionado. Essa biblioteca é especialmente útil em projetos que envolvem interações com o usuário, como controles de dispositivos, interfaces simples ou qualquer aplicação que necessite de entrada por botões.

A biblioteca abstrai a complexidade de configurar pinos GPIO, gerenciar resistores de pull-up e lidar com interrupções, permitindo que o desenvolvedor se concentre na lógica do projeto em vez de detalhes de hardware.

Utilidade da Biblioteca

A biblioteca **ButtonPi** é útil em diversas situações, como:

1. **Projetos com Entrada de Usuário:**
    - Em projetos onde o usuário precisa interagir com o sistema por meio de botões, como em controles remotos, interfaces de usuário simples ou menus navegáveis.
2. **Detecção de Eventos em Tempo Real:**
    - A biblioteca permite a configuração de callbacks que são executados imediatamente quando o botão é pressionado, graças ao uso de interrupções. Isso é ideal para aplicações que exigem respostas rápidas a eventos, como jogos ou sistemas de segurança.
3. **Simplificação do Código:**
    - A biblioteca encapsula a lógica de inicialização e leitura de botões, reduzindo a quantidade de código repetitivo e propenso a erros. Isso torna o código mais limpo e fácil de manter.
4. **Uso em Sistemas Embarcados:**
    - O Raspberry Pi Pico é amplamente utilizado em sistemas embarcados, e a biblioteca **ButtonPi** facilita a integração de botões nesses sistemas, seja para controle de dispositivos, automação residencial ou prototipagem rápida.
5. **Educação e Prototipagem:**
    - Para fins educacionais ou de prototipagem, a biblioteca é uma ferramenta excelente para ensinar conceitos de GPIO, interrupções e gerenciamento de hardware de forma prática e acessível.

Funcionalidades Principais

A biblioteca **ButtonPi** oferece as seguintes funcionalidades:

1. **Inicialização de Botões:**
   - Configura automaticamente o pino GPIO como entrada e habilita o resistor de pull-up interno.
2. **Leitura do Estado do Botão:**
   - Fornece uma função simples para ler o estado atual do botão (pressionado ou liberado).
3. **Callbacks por Interrupção:**
   - Permite registrar funções de callback que são executadas quando o botão é pressionado, utilizando interrupções para detecção de borda de descida.
4. **Simplicidade e Eficiência:**
   - A biblioteca é leve e eficiente, ideal para sistemas com recursos limitados, como o Raspberry Pi Pico.

Documentação Detalhada das Funções

Função: `void ButtonPi_init(ButtonPi *btn, uint pin)`

Propósito:
Inicializa um botão, configurando o pino GPIO ao qual ele está conectado e preparando-o para leitura.

Parâmetros:

- `ButtonPi *btn`: Um ponteiro para a estrutura `ButtonPi` que armazenará as informações do botão.
- `uint pin`: O número do pino GPIO ao qual o botão está conectado.

Comportamento:

1. Armazena o número do pino GPIO na estrutura `ButtonPi`.
2. Inicializa o pino GPIO usando `gpio_init(pin)`.
3. Configura o pino como entrada usando `gpio_set_dir(pin, GPIO_IN)`.
4. Habilita o resistor de pull-up interno com `gpio_pull_up(pin)`.
5. Armazena o estado inicial do botão em `btn->last_state`.

Exemplo de Uso:

```
ButtonPi myButton;
ButtonPi_init(&myButton, 5); // Inicializa o botão no pino 5
```

Função: `bool ButtonPi_read(ButtonPi *btn)`

Propósito:
Lê o estado atual do botão (pressionado ou liberado).

Parâmetros:

- `ButtonPi *btn`: Um ponteiro para a estrutura `ButtonPi` que representa o botão.

Retorno:

- `true`: Se o botão estiver pressionado.
- `false`: Se o botão estiver liberado.

Comportamento:

1. Lê o estado do pino GPIO usando `gpio_get(btn->pin)`.
2. Inverte o estado lógico (já que o botão está conectado ao GND) e retorna o resultado.

Exemplo de Uso:

```
if (ButtonPi_read(&myButton))
{
    printf("Botão pressionado!\n");
}
else
{
    printf("Botão liberado!\n");
}
```

Função: `void ButtonPi_attach_callback(ButtonPi *btn, void (*callback)(void))`

Propósito:
Registra uma função de callback que será chamada quando o botão for pressionado (detecção de borda de descida).

Parâmetros:

- `ButtonPi *btn`: Um ponteiro para a estrutura `ButtonPi` que representa o botão.
- `void (*callback)(void)`: Um ponteiro para a função de callback que será executada quando o botão for pressionado.

Comportamento:

1. Inicializa o gerenciador de interrupções com `gpio_irq_manager_init()`.
2. Verifica se o pino GPIO é válido (0 a 29).

3. Registra a função de callback usando `register_gpio_callback(btn->pin, callback, GPIO_IRQ_EDGE_FALL)`.

Exemplo de Uso:

```
void on_button_press() {
    printf("Botão pressionado!\n");
}

int main()
{
    ButtonPi myButton;
    // Inicializa o botão no pino 5
    ButtonPi_init(&myButton, 5);
    // Registra o callback
    ButtonPi_attach_callback(&myButton, on_button_press);

    while (1)
    {
        // Loop principal (não é necessário fazer nada aqui)
        tight_loop_contents();
    }

    return 0;
}
```

Exemplo Completo de Uso

Aqui está um exemplo completo que demonstra como usar todas as funções da biblioteca **ButtonPi**:

```
#include "ButtonPi.h"
#include "pico/stdlib.h"
#include <stdio.h>

#define LED_PIN 25

// Callback para quando o botão A é pressionado
void button_a_pressed()
{
    printf("Botão A pressionado! Ligando o LED.\n");
    gpio_put(LED_PIN, 1); // Liga o LED
}

// Callback para quando o botão B é pressionado
```

```c
void button_b_pressed()
{
    printf("Botão B pressionado! Desligando o LED.\n");
    gpio_put(LED_PIN, 0); // Desliga o LED
}

int main()
{
    ButtonPi buttonA, buttonB;

    // Inicializa os botões nos pinos padrão
    ButtonPi_init(&buttonA, BUTTON_A_PIN);
    ButtonPi_init(&buttonB, BUTTON_B_PIN);

    // Configura o pino do LED
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);

    // Registra as funções de callback
    ButtonPi_attach_callback(&buttonA, button_a_pressed);
    ButtonPi_attach_callback(&buttonB, button_b_pressed);

    while (1)
    {
        // Loop principal (não é necessário fazer nada aqui)
        tight_loop_contents();
    }

    return 0;
}
```

Explicação do Exemplo:

1. **Botão A:**
   - Quando pressionado, liga o LED conectado ao pino 25 e imprime uma mensagem no console.
2. **Botão B:**
   - Quando pressionado, desliga o LED e imprime uma mensagem no console.
3. **Loop Principal:**
   - O loop principal está vazio porque a lógica é toda tratada por interrupções (callbacks).

Com essa documentação detalhada e exemplos de uso, você está pronto para começar a usar a biblioteca em seus projetos! Para mais detalhes, consulte os arquivos `ButtonPi.h` e `ButtonPi.c`.