Documentação da Biblioteca ButtonPi

Documentação da Biblioteca ButtonPi

A biblioteca `ButtonPi` foi desenvolvida para facilitar o gerenciamento de botões conectados ao Raspberry Pi Pico. Ela permite inicializar botões, ler seu estado e registrar funções de callback que serão chamadas quando o botão for pressionado. A biblioteca utiliza interrupções para detectar bordas de descida, garantindo que os callbacks sejam chamados de forma eficiente e sem a necessidade de polling.

Estrutura da Biblioteca

A biblioteca é composta por dois arquivos principais:

- **ButtonPi.h**: Contém as declarações das funções e estruturas utilizadas pela biblioteca.
- **ButtonPi.c**: Contém a implementação das funções declaradas no arquivo `.h`.

Estrutura `ButtonPi`

A estrutura `ButtonPi` armazena as informações de um botão, incluindo o pino GPIO ao qual ele está conectado e o último estado lido.

```
typedef struct {
    uint pin;                   // Pino GPIO ao qual o botão está conectado
    bool last_state;            // Último estado lido do botão (para
detecção de borda)
} ButtonPi;
```

Funções da Biblioteca

`ButtonPi_init`

Inicializa um botão em um pino GPIO específico. Configura o pino como entrada, habilita o resistor de pull-up interno e inicializa o estado do botão.

```
void ButtonPi_init(ButtonPi *btn, uint pin);
```

Parâmetros:

- `btn`: Ponteiro para a estrutura `ButtonPi` que representa o botão.
- `pin`: Pino GPIO ao qual o botão está conectado.

Exemplo de uso:

```
ButtonPi myButton;
ButtonPi_init(&myButton, 15);  // Inicializa o botão no pino GPIO 15
```

`ButtonPi_read`

Lê o estado atual do botão. Retorna `true` se o botão estiver pressionado e `false` caso contrário.

```
bool ButtonPi_read(ButtonPi *btn);
```

Parâmetros:

- `btn`: Ponteiro para a estrutura `ButtonPi` que representa o botão.

Retorno:

- `true` se o botão estiver pressionado.
- `false` se o botão não estiver pressionado.

Exemplo de uso:

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

`ButtonPi_attach_callback`

Registra uma função de callback para ser chamada quando o botão for pressionado. Configura uma interrupção na borda de descida (quando o botão é pressionado) para chamar a função de callback fornecida.

```
void ButtonPi_attach_callback(ButtonPi *btn, void (*callback)(void));
```

Parâmetros:

- `btn`: Ponteiro para a estrutura `ButtonPi` que representa o botão.
- `callback`: Função de callback que será chamada quando o botão for pressionado.

Exemplo de uso:

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

ButtonPi myButton;
ButtonPi_init(&myButton, 15);
ButtonPi_attach_callback(&myButton, button_pressed);
```

Exemplo Completo

Aqui está um exemplo completo que inicializa um botão no pino GPIO 15, lê seu estado e registra uma função de callback para ser chamada quando o botão for pressionado.

```c
/**
 * @file main.c
 * @brief Exemplo de uso da biblioteca ButtonPi para gerenciar um botão no Raspberry Pi Pico.
 *
 * Este exemplo demonstra como usar a biblioteca ButtonPi para inicializar um botão, registrar uma função de callback
 * para ser chamada quando o botão é pressionado (usando interrupções) e verificar o estado do botão continuamente
 * (usando polling).
 */

#include "ButtonPi.h"
#include <stdio.h>

/**
 * @brief Função de callback chamada quando o botão é pressionado.
 *
 * Esta função é executada quando uma interrupção é disparada devido ao pressionamento do botão.
 * Ela imprime uma mensagem no console indicando que o botão foi pressionado.
 */
void button_pressed() {
    printf("Botão pressionado! (Interrupção Disparada)\n");
}

/**
 * @brief Função principal do programa.
 *
 * Inicializa um botão no pino GPIO 15, registra a função de callback `button_pressed` para ser chamada
 * quando o botão é pressionado e entra em um loop infinito onde verifica o estado do botão continuamente
 * (polling) e imprime o estado atual.
 *
 * @return int Retorna 0 ao finalizar (embora o programa entre em um loop infinito e não retorne).
 */
```

```c
int main() {
    // Estrutura que representa o botão
    ButtonPi myButton;

    // Inicializa o botão no pino GPIO 15
    ButtonPi_init(&myButton, 15);

    // Registra a função de callback para ser chamada quando o botão é pressionado
    ButtonPi_attach_callback(&myButton, button_pressed);

    // Loop principal que verifica o estado do botão continuamente (polling)
    while (true) {
        if (ButtonPi_read(&myButton)) {
            printf("Botão está pressionado (Polling).\n");
        } else {
            printf("Botão não está pressionado.\n");
        }
        sleep_ms(100); // Aguarda 100ms antes de verificar novamente
    }

    return 0;
}
```

Considerações Finais

A biblioteca `ButtonPi` é uma ferramenta simples e eficaz para gerenciar botões no Raspberry Pi Pico. Ela abstrai a complexidade de configurar interrupções e gerenciar estados, permitindo que você se concentre na lógica do seu projeto. Com a capacidade de registrar callbacks, você pode facilmente adicionar comportamentos complexos ao pressionamento de botões sem a necessidade de polling contínuo.

Para mais informações sobre o Raspberry Pi Pico e suas funcionalidades GPIO, consulte a documentação oficial.