Documentação da Biblioteca `JoystickPi`

Documentação da Biblioteca `JoystickPi`

Introdução

A biblioteca **JoystickPi** foi desenvolvida para simplificar a integração e o gerenciamento de um módulo de joystick com o Raspberry Pi Pico. Ela fornece funções para inicializar o joystick, ler seus eixos analógicos (X e Y) e detectar o estado do seu botão. A biblioteca também inclui funções utilitárias para mapear valores analógicos para intervalos personalizados, facilitando a interpretação dos movimentos do joystick em várias aplicações.

Essa biblioteca é especialmente útil para projetos que envolvem entrada do usuário, como controles de jogos, robótica ou qualquer aplicação que necessite de controle direcional e entrada por botão.

Propósito da Biblioteca

A biblioteca **JoystickPi** foi criada para:

1. **Simplificar a Integração do Joystick:**
   - Fornecer uma interface fácil de usar para ler valores analógicos dos eixos X e Y do joystick.
   - Lidar com a entrada digital do botão do joystick.
2. **Conversão Analógico-Digital:**
   - Utilizar o ADC (Conversor Analógico-Digital) do Raspberry Pi Pico para ler os valores analógicos do joystick.
3. **Detecção do Estado do Botão:**
   - Detectar o estado do botão do joystick, com suporte a resistores de pull-up para garantir leituras precisas.
4. **Mapeamento de Valores:**
   - Fornecer uma função utilitária para mapear valores analógicos brutos para intervalos personalizados, o que é útil para interpretar os movimentos do joystick em diferentes contextos.

Funcionalidades Principais

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

1. **Inicialização do Joystick:**
   - Configura o ADC e os pinos GPIO para leitura dos eixos analógicos e do botão do joystick.

2. **Leitura do Estado do Joystick:**
    - Lê o estado atual do joystick, incluindo os eixos X e Y e o botão.
3. **Leitura Individual dos Eixos e Botão:**
    - Fornece funções separadas para ler o eixo X, o eixo Y e o estado do botão individualmente.
4. **Mapeamento de Valores:**
    - Mapeia valores analógicos brutos para um intervalo especificado, permitindo uma interpretação fácil dos movimentos do joystick.

Documentação das Funções

Aqui está uma explicação detalhada de todas as funções da biblioteca, com foco em seu propósito e comportamento.

Função: `void joystickPi_init()`

Propósito:
Inicializa o joystick, configurando o ADC e os pinos GPIO para leitura dos eixos analógicos e do botão.

Comportamento:

1. Inicializa o ADC usando `adc_init()`.
2. Configura os pinos dos eixos X e Y (`JOYSTICK_X_PIN` e `JOYSTICK_Y_PIN`) como entradas analógicas usando `adc_gpio_init()`.
3. Configura o pino do botão (`JOYSTICK_BUTTON_PIN`) como uma entrada digital usando `gpio_init()` e `gpio_set_dir()`.
4. Habilita o resistor de pull-up interno para o pino do botão usando `gpio_pull_up()`.

Exemplo de Uso:

```
joystickPi_init(); // Inicializa o joystick
```

Função: `joystick_state_t joystickPi_read()`

Propósito:
Lê o estado atual do joystick, incluindo os eixos X e Y e o botão.

Valor de Retorno:

- Uma estrutura `joystick_state_t` contendo os seguintes campos:
    - `x`: O valor analógico bruto do eixo X.
    - `y`: O valor analógico bruto do eixo Y.

- `button`: O estado do botão (`true` se pressionado, `false` se não pressionado).

Comportamento:

1. Lê o valor do eixo X selecionando o canal ADC correspondente (`adc_select_input(1)`) e chamando `adc_read()`.
2. Lê o valor do eixo Y selecionando o canal ADC correspondente (`adc_select_input(0)`) e chamando `adc_read()`.
3. Lê o estado do botão usando `gpio_get()` e inverte o valor (já que o botão está conectado com um resistor de pull-up).

Exemplo de Uso:

```
joystick_state_t state = joystickPi_read();
printf("X: %d, Y: %d, Botão: %d\n", state.x, state.y, state.button);
```

Função: `uint16_t joystickPi_read_x()`

Propósito:
Lê o valor analógico bruto do eixo X do joystick.

Valor de Retorno:

- O valor analógico bruto do eixo X (0 a 4095 para um ADC de 12 bits).

Comportamento:

1. Seleciona o canal ADC para o eixo X (`adc_select_input(0)`).
2. Lê o valor analógico usando `adc_read()`.

Exemplo de Uso:

```
uint16_t x_value = joystickPi_read_x();
printf("Valor do eixo X: %d\n", x_value);
```

Função: `uint16_t joystickPi_read_y()`

Propósito:
Lê o valor analógico bruto do eixo Y do joystick.

Valor de Retorno:

- O valor analógico bruto do eixo Y (0 a 4095 para um ADC de 12 bits).

Comportamento:

1. Seleciona o canal ADC para o eixo Y (`adc_select_input(1)`).

2. Lê o valor analógico usando `adc_read()`.

Exemplo de Uso:

```
uint16_t y_value = joystickPi_read_y();
printf("Valor do eixo Y: %d\n", y_value);
```

Função: `bool joystickPi_read_button()`

Propósito:
Lê o estado do botão do joystick.

Valor de Retorno:

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

Comportamento:

1. Lê o estado do botão usando `gpio_get()`.
2. Inverte o valor (já que o botão está conectado com um resistor de pull-up).

Exemplo de Uso:

```
bool button_state = joystickPi_read_button();
if (button_state) {
    printf("Botão pressionado!\n");
}
```

Função: `int16_t joystickPi_map_value(uint16_t value, uint16_t min_input, uint16_t max_input, int16_t min_output, int16_t max_output)`

Propósito:
Mapeia um valor analógico bruto de um intervalo para outro.

Parâmetros:

- `value`: O valor analógico bruto a ser mapeado.
- `min_input`: O valor mínimo do intervalo de entrada.
- `max_input`: O valor máximo do intervalo de entrada.
- `min_output`: O valor mínimo do intervalo de saída.
- `max_output`: O valor máximo do intervalo de saída.

Valor de Retorno:

- O valor mapeado no intervalo de saída.

Comportamento:

1. Mapeia o valor de entrada do intervalo de entrada (`min_input` a `max_input`) para o intervalo de saída (`min_output` a `max_output`).

Exemplo de Uso:

```
uint16_t raw_value = joystickPi_read_x();
int16_t mapped_value = joystickPi_map_value(raw_value, 0, 4095, -100, 100);
printf("Valor X mapeado: %d\n", mapped_value);
```

Conclusão

A biblioteca **JoystickPi** fornece uma maneira simples e eficiente de integrar um joystick ao Raspberry Pi Pico, permitindo leitura dos eixos analógicos e do botão com facilidade. Com essa documentação detalhada, você está pronto para utilizar a biblioteca em seus projetos e aproveitar ao máximo suas funcionalidades. Para mais detalhes, consulte o arquivo `JoystickPi.c` e `JoystickPi.h`.