Documentação da Biblioteca `JoystickPi`

Documentação da Biblioteca JoystickPi

A biblioteca `JoystickPi` foi desenvolvida para facilitar a leitura de um joystick analógico conectado ao Raspberry Pi Pico. O joystick possui dois eixos (X e Y) e um botão. Os eixos são lidos através de conversores analógico-digitais (ADC), e o botão é lido como uma entrada digital com resistor de pull-up.

Estrutura da Biblioteca

A biblioteca é composta por dois arquivos principais:

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

Estrutura `joystick_state_t`

A estrutura `joystick_state_t` armazena os valores dos eixos X e Y e o estado do botão.

```
typedef struct {
    uint16_t x;      // Valor do eixo X (0-4095)
    uint16_t y;      // Valor do eixo Y (0-4095)
    bool button;     // Estado do botão (true se pressionado, false caso contrário)
} joystick_state_t;
```

Funções da Biblioteca

`joystickPi_init`

Inicializa o joystick, configurando os pinos ADC para leitura dos eixos X e Y e o pino GPIO para leitura do botão. Habilita o resistor de pull-up no pino do botão.

```
void joystickPi_init();
```

Exemplo de uso:

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

`joystickPi_read`

Lê o estado atual do joystick, incluindo os valores dos eixos X e Y e o estado do botão.

```
joystick_state_t joystickPi_read();
```

Retorno:

- Estrutura `joystick_state_t` contendo os valores dos eixos X e Y e o estado do botão.

Exemplo de uso:

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

`joystickPi_read_x`

Lê apenas o valor do eixo X do joystick.

```
uint16_t joystickPi_read_x();
```

Retorno:

- Valor do eixo X (0-4095).

Exemplo de uso:

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

`joystickPi_read_y`

Lê apenas o valor do eixo Y do joystick.

```
uint16_t joystickPi_read_y();
```

Retorno:

- Valor do eixo Y (0-4095).

Exemplo de uso:

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

`joystickPi_read_button`

Lê apenas o estado do botão do joystick.

```
bool joystickPi_read_button();
```

Retorno:

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

Exemplo de uso:

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

`joystickPi_map_value`

Mapeia um valor de uma faixa de entrada para uma faixa de saída. Útil para normalizar os valores do ADC para uma faixa desejada (ex: -100 a 100).

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

Parâmetros:

- `value`: Valor a ser mapeado.
- `min_input`: Valor mínimo da faixa de entrada.
- `max_input`: Valor máximo da faixa de entrada.
- `min_output`: Valor mínimo da faixa de saída.
- `max_output`: Valor máximo da faixa de saída.

Retorno:

- Valor mapeado para a faixa de saída.

Exemplo de uso:

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

Exemplo Completo

Aqui está um exemplo completo que inicializa o joystick, lê os valores dos eixos X e Y, verifica o estado do botão e mapeia os valores do ADC para uma faixa personalizada.

```
#include "JoystickPi.h"
#include <stdio.h>
```

```c
int main() {
    // Inicializa o joystick
    joystickPi_init();

    while (true) {
        // Lê o estado do joystick
        joystick_state_t state = joystickPi_read();

        // Mapeia os valores do ADC para uma faixa de -100 a 100
        int16_t x_mapped = joystickPi_map_value(state.x, 0, 4095, -100,
100);
        int16_t y_mapped = joystickPi_map_value(state.y, 0, 4095, -100,
100);

        // Exibe os valores lidos
        printf("Eixo X: %d (Mapeado: %d), Eixo Y: %d (Mapeado: %d), Botão:
%d\n",
               state.x, x_mapped, state.y, y_mapped, state.button);

        // Aguarda 100ms antes de ler novamente
        sleep_ms(100);
    }

    return 0;
}
```

Saída Esperada

Se o joystick estiver sendo movido e o botão pressionado, a saída no console pode ser algo como:

```
Eixo X: 2048 (Mapeado: 0), Eixo Y: 1024 (Mapeado: -50), Botão: 0
Eixo X: 4095 (Mapeado: 100), Eixo Y: 0 (Mapeado: -100), Botão: 1
Eixo X: 0 (Mapeado: -100), Eixo Y: 4095 (Mapeado: 100), Botão: 0
```

- **Eixo X e Y**: Valores brutos do ADC e seus valores mapeados.
- **Botão**: `1` se pressionado, `0` se não pressionado.

Considerações Finais

A biblioteca `JoystickPi` simplifica a leitura de um joystick analógico no Raspberry Pi Pico, fornecendo funções para inicialização, leitura dos eixos e botão, e mapeamento de valores. Ela é útil em aplicações que requerem controle de movimento ou entrada de usuário, como em jogos, robótica ou interfaces de controle.

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