Documentação da Biblioteca `BuzzerPi`

Documentação da Biblioteca BuzzerPi

A biblioteca `BuzzerPi` foi desenvolvida para controlar um buzzer no Raspberry Pi Pico usando PWM (Modulação por Largura de Pulso). Ela permite tocar tons, melodias e beeps com controle de frequência, duração e divisão de clock.

Funcionalidades Principais

1. **Inicialização do PWM**: Configura o pino GPIO para funcionar como saída PWM.
2. **Cálculo de Frequência**: Calcula o valor de "wrap" necessário para gerar frequências específicas.
3. **Reprodução de Tons**: Toca tons únicos com controle de frequência e duração.
4. **Reprodução de Melodias**: Toca sequências de tons (melodias) a partir de arrays de frequências e durações.
5. **Reprodução de Beeps**: Reproduz beeps repetidos com controle de frequência, duração e número de repetições.

Estrutura da Biblioteca

A biblioteca é composta por dois arquivos principais:

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

Funções da Biblioteca

`initialize_pwm`

Inicializa o PWM no pino especificado, configurando-o como saída PWM.

```
void initialize_pwm(uint pin);
```

Parâmetros:

- `pin`: Pino GPIO onde o buzzer está conectado.

Exemplo de uso:

```
initialize_pwm(15);  // Inicializa o PWM no pino GPIO 15
```

`calculate_wrap`

Calcula o valor de "wrap" necessário para gerar uma frequência específica.

```
uint16_t calculate_wrap(uint32_t target_frequency, float clkdiv);
```

Parâmetros:

- `target_frequency`: Frequência desejada em Hz.
- `clkdiv`: Divisor de clock usado para o PWM.

Retorno:

- Valor de "wrap" calculado. Se o valor exceder 65535, retorna 65535.

Exemplo de uso:

```
uint16_t wrap_value = calculate_wrap(440, 1.0f);  // Calcula o valor de wrap para 440 Hz
```

`play_tone`

Toca um tom no buzzer com a frequência e duração especificadas.

```
void play_tone(uint pin, uint32_t freq, uint duration_ms);
```

Parâmetros:

- `pin`: Pino GPIO onde o buzzer está conectado.
- `freq`: Frequência do tom em Hz.
- `duration_ms`: Duração do tom em milissegundos.

Exemplo de uso:

```
play_tone(15, 440, 1000);  // Toca um tom de 440 Hz por 1 segundo no pino GPIO 15
```

`play_tone_clkdiv`

Toca um tom no buzzer com a frequência, duração e divisor de clock especificados.

```
void play_tone_clkdiv(uint pin, int freq, int duration_ms, float clkdiv);
```

Parâmetros:

- `pin`: Pino GPIO onde o buzzer está conectado.
- `freq`: Frequência do tom em Hz.
- `duration_ms`: Duração do tom em milissegundos.
- `clkdiv`: Divisor de clock usado para o PWM.

Exemplo de uso:

```
play_tone_clkdiv(15, 440, 1000, 2.0f);  // Toca um tom de 440 Hz por 1 segundo com divisor de clock 2.0
```

`play_melody`

Toca uma melodia a partir de arrays de frequências e durações.

```
void play_melody(uint pin, int *melody, int *durations, float clkdiv, int length);
```

Parâmetros:

- `pin`: Pino GPIO onde o buzzer está conectado.
- `melody`: Array de frequências que compõem a melodia.
- `durations`: Array de durações correspondentes a cada frequência.
- `clkdiv`: Divisor de clock usado para o PWM.
- `length`: Número de notas na melodia.

Exemplo de uso:

```
int melody[] = {440, 494, 523, 587, 659, 698, 784, 880};
int durations[] = {500, 500, 500, 500, 500, 500, 500, 500};
play_melody(15, melody, durations, 1.0f, 8);  // Toca uma melodia no pino GPIO 15
```

`beep`

Reproduz um beep repetido no buzzer.

```
void beep(uint pin, int freq, int duration, int repetition);
```

Parâmetros:

- `pin`: Pino GPIO onde o buzzer está conectado.
- `freq`: Frequência do beep em Hz.
- `duration`: Duração de cada beep em milissegundos.
- `repetition`: Número de vezes que o beep será repetido.

Exemplo de uso:

```
beep(15, 440, 200, 3);  // Reproduz um beep de 440 Hz por 200 ms, 3 vezes no pino GPIO 15
```

Exemplo Completo

Aqui está um exemplo completo que inicializa o buzzer, toca um tom, uma melodia e um beep repetido:

```c
#include "BuzzerPi.h"

int main() {
    // Inicializa o buzzer no pino GPIO 15
    initialize_pwm(15);

    // Toca um tom de 440 Hz por 1 segundo
    play_tone(15, 440, 1000);

    // Toca uma melodia
    int melody[] = {440, 494, 523, 587, 659, 698, 784, 880};
    int durations[] = {500, 500, 500, 500, 500, 500, 500, 500};
    play_melody(15, melody, durations, 1.0f, 8);

    // Reproduz um beep repetido
    beep(15, 440, 200, 3);

    return 0;
}
```

Considerações Finais

A biblioteca `BuzzerPi` é uma ferramenta poderosa para controlar um buzzer no Raspberry Pi Pico. Ela permite tocar tons, melodias e beeps de forma simples e eficiente, com controle total sobre a frequência, duração e divisor de clock. Isso a torna ideal para aplicações que requerem feedback sonoro, como alarmes, notificações ou até mesmo música.

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