# Relatório do Projeto hid\_composite\_consumer\_control

## Introdução

Este relatório documenta o projeto hid\_composite\_consumer\_control, uma implementação de dispositivo USB HID (Human Interface Device) customizado, baseado no projeto hid\_composite original da biblioteca TinyUSB para Raspberry Pi Pico (RP2040). O projeto foi modificado para focar exclusivamente na funcionalidade de Consumer Control.

## Estrutura de Diretórios

hid\_composite\_consumer\_control/  
├── CMakeLists.txt  
├── hid\_composite.c  
├── usb\_descriptors.c  
├── usb\_descriptors.h  
├── tusb\_config.h  
└── build/ (diretório gerado após compilação)

## Descrição Geral dos Arquivos

• CMakeLists.txt: Arquivo de configuração do CMake que define como o projeto será compilado, incluindo as dependências, flags de compilação e bibliotecas utilizadas.

• hid\_composite.c: Arquivo principal de código-fonte contendo a lógica do dispositivo HID, incluindo inicialização de GPIOs, callbacks do TinyUSB, tarefa de HID e envio de relatórios de controle de mídia.

• usb\_descriptors.c: Define os descritores USB necessários para descrever ao host o tipo de dispositivo HID implementado, incluindo a definição do Report Descriptor para Consumer Control.

• usb\_descriptors.h: Header file contendo as declarações associadas aos descritores USB.

• tusb\_config.h: Arquivo de configuração da biblioteca TinyUSB, definindo parâmetros como o número de interfaces habilitadas e o tamanho do endpoint.

• build/: Diretório gerado automaticamente pelo CMake e Ninja no processo de compilação, contendo os arquivos objeto, binários e intermediários da build.

## Funcionamento do Projeto

O projeto hid\_composite\_consumer\_control funciona como um dispositivo USB HID configurado exclusivamente para enviar comandos da categoria Consumer Control (controle multimídia, como volume up/down) ao sistema operacional.  
Do projeto original hid\_composite (que suportava múltiplas interfaces HID como teclado, mouse e gamepad), este projeto removeu os demais relatórios e focou apenas no Report Descriptor e na lógica de Consumer Control.  
O envio de relatórios é realizado via a função tud\_hid\_report(), acionada por eventos de leitura nos GPIOs associados aos botões de volume.

Na placa um botão no GPIO 5, quando pressionado aumenta o volume, outro botão no GPIO 6, quando pressiona diminue o volume.

## Personalização e Reutilização

Este projeto serve como um exemplo de HID customizado baseado no template hid\_composite. Para criar outros dispositivos HID especializados, pode-se utilizar esta estrutura como base, modificando o Report Descriptor (usb\_descriptors.c) e a lógica de controle (hid\_composite.c) para suportar as funcionalidades desejadas. Essa abordagem facilita a criação de dispositivos HID com diferentes perfis, eliminando o que não é necessário do projeto original.

## Conclusão

O projeto hid\_composite\_consumer\_control demonstra como especializar um firmware HID, mantendo apenas os elementos necessários e simplificando o código base. Ele é uma base sólida para aplicações embarcadas onde apenas uma funcionalidade específica de controle de mídia (Consumer Control) é desejada, e pode ser expandido ou adaptado para outras finalidades HID.