Centro Federal de Educação Tecnológica de Minas Gerais
Departamento de Engenharia Elétrica
Curso: Engenharia Elétrica -
Disciplina: Sistemas Embarcados
Professores: Túlio Carvalho

Prática 04
LED PWM Controller

Objetivo

LED PWM Controller

Procedimento

1) Adicione o exemplo ledc_basic seguindo o procedimento adotado nas práticas anteriores. Faça o *build* do projeto. Este exemplo mostra como gerar um sinal PWM utilizando o módulo LEDC.

2) Monitore o funcionamento do programa e plote o terminal de saída. Para isso, altere os LEDC_OUTPUT_IO para GPIO16/17/26/33/32. Altere o LEDC_MODE para LEDC_HIGH_SPEED_MODE. Qual o duty cycle ajustado? Altere os valores de frequência e duty cycle e observe os resultados no osciloscópio.

Centro Federal de Educação Tecnológica de Minas Gerais
Departamento de Engenharia Elétrica
Curso: Engenharia Elétrica -
Disciplina: Sistemas Embarcados
Professores: Túlio Carvalho

Projeto
Data de avaliação 29/10/2025
Valor 9,0 pts

1) Seguindo o procedimento anterior, utilize o mesmo programa das práticas anteriores. Neste exercício você irá criar um 2 PWMs que funcionarão sincronizados. O primeiro será ligado a um LED e o segundo ao oscilosópio. Os PWMs controlarão o brilho do LED de forma automática por uma forma de onda do tipo dente de serra ou manualmente por meio de incremento de um botão. Utilize o botão 1 para escolher automático, botão 2 manual e botão 3 para incrementar o duty cycle.

2) Crie uma Task para o PWM.

3) Inicialize (dentro da task) 2 PWMs com as seguintes carcaterísticas:
    a. Configure o timer do PWM para trabalhar com frequência de 5 KHz e resolução de 13 bits
    b. Associe o canal 0 para um dos LEDs conectados aos GPIOs 16/17/26 e o canal 1 aos GPIO 33/32

4) Crie um semaphoro binário para sincronizar task (GPIO) com task do timer. Este semaphoro será responsável por sincronizar o incremento a cada 100 milisegundo.

Dica:

```
static SemaphoreHandle_t semaphore_pwm = NULL; // variável global
semaphore_pwm = xSemaphoreCreateBinary(); // criação do semaphore binario
xSemaphoreGive(semaphore_pwm); //Função na TASK Timer para sincronizar com a task PWM
xSemaphoreTake( semaphore_pwm, portMAX_DELAY )
```

5) Sincronize o aperto do botão com o ajuste do PWM da seguinte forma:
    a. Botão 21 coloca o PWM em automatico.
    b. Botão 22 colocar o PWM em manual.
    c. Botão 23 incrementa o duty cycle quando estiver no modo manual.
    d. Utilize uma fila para passar as informações de task-to-task (GPIO-PWM).
    e. Crie também um tipo PWM_elements_t para passar um um bool (automático ou não) e um inteiro (int16_t) pela fila.

6) Crie uma nova TAG para o PWM e informe através do ESP_LOGI O duty cycle atual a cada aperto de botão (qualquer um dos 3).

7) Critérios de avaliação:
    a. Projeto salvo e armazenado corretamente no github e compartilhado com o professor
    b. Programa devidamente comentado
    c. Criação da tarefa (task)
    d. Uso da fila (Queue) e semaphoro.
    e. Configurações do PWM conforme solicitado
    f. Utilização da biblioteca ESP_LOG para auxiliar na depuração do programa.