Ministério da Educação
**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**
*Campus Curitiba*
Departamento Acadêmico de Eletrônica (Daeln-CT)
**ELF52 – Sistemas Microcontrolados**

Prof.: Guilherme de S. Peron
Prof.: Marcos E. P. Monteiro
Prof.: Paulo D. G. da Luz

# Atividade Prática 2 - GPIO

**Contexto:**

**GPIOs** (*General Purpose Input Output*) são pinos genéricos em microcontroladores para trocar informações com mundo externo. Podem ser utilizados para leitura de níveis lógicos, por exemplo leitura de chaves, ou escrita binária, por exemplo controlar LEDs.

**Objetivo:**

Utilizando instruções Assembly para Cortex-M4 e o kit de desenvolvimento EK-TM4C1294XL, desenvolver um programa que controle o acendimento dos LEDs de duas formas diferentes:

- Passeio do cavaleiro: Os 4 LEDs (D1 a D4) devem realizar o algoritmo do passeio do cavaleiro, ou seja, no tempo *t* acende-se D1, no tempo *t*+1 apaga-se o D1 e acende-se o D2, até acender o D4. Após isso, fazer o caminho contrário apagando-se D4 e acendendo-se D3, e assim, até acender-se o D1. O algoritmo apresenta no total 6 estados que são representados na Figura 1.

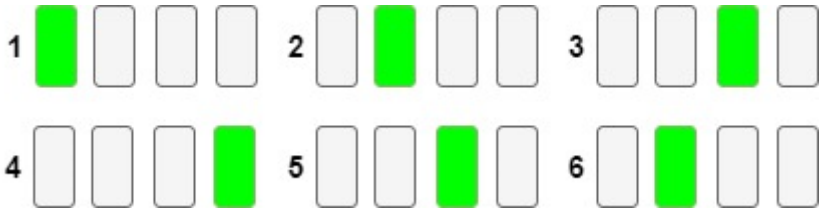

Figura 1: Estados do Passeio do Cavaleiro

- Contador binário de 0 a 15: Os LEDs devem realizar a contagem binária de 0 a 15, possuindo 16 estágios, representados na Figura 2.

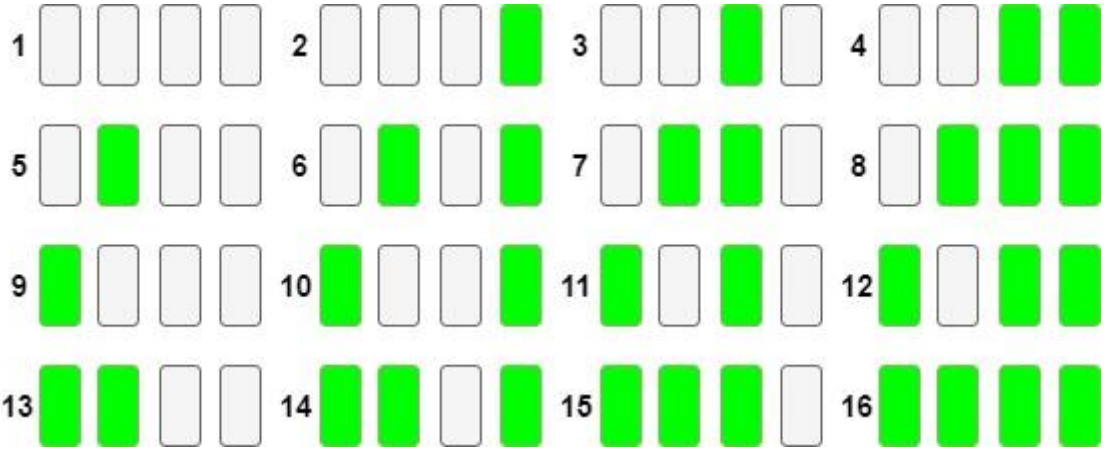

Figura 2: Estados do Contador de 0 a 15

**Tarefas:**

O programa deverá fazer o controle por meio das chaves USR_SW1 e USR_SW2 do modo de operação e da velocidade de acendimento dos LEDs.

1) A chave USR_SW1 deverá fazer o controle do modo de operação:
   a) Passeio do Cavaleiro;
   b) Contador Binário de 0 a 15;
2) A chave USR_SW2 deverá fazer o controle do acendimento de cada LED:
   a) 1000ms;
   b) 500ms;
   c) 200ms.

Ao inicializar o programa, os LEDs deverão estar acendendo conforme o algoritmo do passeio do cavaleiro. Ao se pressionar a chave USR_SW1 os LEDs deverão alterar o modo de acendimento para o contador binário. Ao se pressionar novamente a chave USR_SW1 novamente, o modo de operação deverá voltar para o passeio do cavaleiro e assim sucessivamente sempre alternando o modo de operação a cada vez que a chave USR_SW1 é pressionada.

De maneira similar, quando o programa é inicializado, em relação à velocidade de acendimento dos LEDs, cada LED deverá ficar aceso por 1000ms. Ao se pressionar a USR_SW2, cada LED deverá ficar aceso por 500ms. Ao se pressionar novamente a USR_SW2, cada LED deverá acender por 200ms. Um novo pressionamento da USR_SW2 deve retornar a velocidade de acendimento para 1000ms e assim sucessivamente.

**Entregar a pasta do projeto do Keil com todos os arquivos zipado e o fluxograma da ideia proposta (preferencialmente em algum site ou aplicativo, e.g. http://draw.io). Nomear o arquivo com o nome e o último sobrenome dos dois alunos da dupla. Ex.: fulanodetal1_fulanodetal2_ap2.zip**

**Dicas:**

A implementação pode variar de equipe para equipe. Entretanto pode-se seguir as seguintes dicas:

1) A partir de agora todos os programas devem ser executados em um laço infinito. Normalmente a partir do label "Start" há a inicialização das variáveis (Reigstradores, memória RAM) e na sequência um outro **label** (por exemplo "mainloop"). Sempre que chegar ao final da execução voltar ao "mainloop";

2) Lembrar de fazer a inicialização de todos os periféricos, chamando as funções PLL_Init, Systick_Init e GPIO_Init. As duas primeiras, são as mesmas do arquivo GPIO2.zip, basta incluir o arquivo utils.s. Para a GPIO_Init, pode-se usar como base os exercícios do tópico sobre GPIO;

3) Utilizar 4 registradores como variáveis globais, para armazenar os seguintes estados:
   a) Modo de operação;
   b) Velocidade;
   c) Estado do algoritmo do passeio (para deixar armazenado se for trocado para o modo contador)

d) Estado do contador (para deixar armazenado caso for trocado para o modo passeio)

Durante o código inteiro não utilizar esses registradores para nenhuma outra coisa, a menos que eles sejam empilhados durante o uso e desempilhados na sequência.
**NÃO** utilizar os registradores R0 a R3 para isso, porque estes são utilizados convencionalmente para passagem de parâmetros e operações locais.

3) O loop principal pode ter apenas 3 chamadas de funções:
   a) Testar as chaves;
   b) Acender os Leds;
   c) Esperar o tempo.

4) A função acender LEDs, pode ser escrita como duas grandes tabelas, utilizando as instruções de CMP e B{cond}. Primeiro verifica-se o modo de operação dos LEDs, para saber para qual tabela irá ser varrida. Depois pula-se para respectiva tabela daquele modo (passeio do cavaleiro ou contador)

5) A partir daí verificar em qual estado do acendimento está e fazer a chamada das funções de acender os LEDs PortN_Output e PortF_Output. Depois incrementa-se o estado do contador do estado respectivo.

6) A função para esperar o tempo apenas testa qual o tempo está configurado e chama a função SysTick_Wait1ms.

7) A função de verificar as chaves chama a função PortJ_Input para ler o estado das chaves. Depois verifica-se qual das chaves está pressionada. Após isso insere-se um pequeno delay de 100ms para garantir o debouncing *(evitar o efeito repique)* das chaves.

8) De acordo com o pressionamento da chave 1 ou 2, chamar a função para alterar o modo ou a velocidade;

9) Como ainda não se estudou as interrupções e evitar que fique mudando o modo ou a velocidade indefinidamente enquanto as chaves estão pressionadas, ao final da função adicionar um teste para verificar se as chaves voltaram para a posição original, realizando um outro PortJ_Input e, verificando se ambas as chaves estão em 2_11.

10) Lembrar de fazer sempre o empilhamento do LR quando for chamar funções em cascata.

*Obs.: Essas são apenas dicas de desenvolvimento. Outras formas de implementação poderão ser adotadas.*