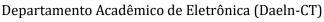


#### Ministério da Educação

#### UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ









Prof.: Paulo D. G. da Luz

# ATIVIDADE PRÁTICA 5 - MÚLTIPLOS PERIFÉRICOS (12C, PWM, UART, GPIO)

#### Contexto:

O domínio da utilização de múltiplos periféricos é essencial em sistemas embarcados. É muito comum um controlador gerenciar múltiplos periféricos em projetos reais. Portanto é muito importante aprimorar as técnicas de "Super Loop" para pequenos projetos e soluções. Visando a continuação do aprendizado do SDK da biblioteca "*TivaWare*", será introduzido dois novos periféricos I2C e PWM.

1 Alterar o programa da atividade prática 4 para que funcione somente com 03 x Relés, sendo que um deles agora irá se tornar um LED com controle de brilho por **PWM**. Utilizar 03 x Leds na placa do Kit TIVA EK-TM4C1294XL como "Relés". Simular os Relés nos 03 leds da placa: LED1 (PN1), LED2 (PNO), LED3 (PF4) e LED4 (PF0). O LED ligado significa Relé ligado e LED desligado significa Relé desligado. Um dos quatro LEDs deve ter agora seu brilho controlado por **PWM** e não irá representar mais o estado de um Relé. A escolha do LED deve ser feita conforme o datasheet do TM4C1294NCPDT, procurando qual GPIO dos LEDs pode ter a função especial de **PWM**(Cap. 10 Signal Description – Tabela 10-2: GPIO Pins and Alternate Functions – pg. 743 / Cap. 23 - Pulse Width Modulator – pg. 1669). O programa deve ser no estilo "Super Loop", devendo ficar em loop infinito sempre que receber um pacote válido na serial, deve atuar no *hardware*. Ao mesmo tempo foi desenvolvido pelo professor um shield para ser acoplado na placa EK-TM4C1294XL contendo 02 x displays gráficos monocromáticos e um teclado matricial (4x4). Utilizaremos um display em modo gráfico para indicar o estado dos 03 x Relés. O outro display será utilizado em modo texto e deve mostrar o % de 00 a 99 do brilho do LED. O ajuste do brilho será feito via teclado sendo possível ser editado e alterado a qualquer instante. Lembrando que durante uma alteração de brilho, não se pode perder dados da serial e não podemos atrasar a atualização do display gráfico.

#### Dicas:

O protocolo desenvolvido para esta atividade está descrito no arquivo: **Protocolo\_Lab04.pdf** <a href="http://www.elf74.daeln.com.br/Labs/Protocolo\_Lab04.pdf">http://www.elf74.daeln.com.br/Labs/Protocolo\_Lab04.pdf</a>. Também está disponível para *download* o programa **sscom3.2.exe**, que é um utilitário para comunicação serial e que não precisa ser instalado no Windows. Também está disponível para *download* o programa **Lab4\_Serial.exe**, que é um

programa para comunicação serial que implementa o protocolo descrito no **PDF** e que não precisa ser instalado no Windows.

Para auxiliar neste processo segue o **PDF** do esquemático do *Shield* e um exemplo de inicialização e demonstração dos *displays* também foi disponibilizado.

http://www.elf74.daeln.com.br/Placas/Tlcd\_I2C/Tiva\_EKTM4C1294XL\_TL\_CD\_I2C.pdf

http://www.elf74.daeln.com.br/Placas/Tlcd I2C/OLED EKTM4C1294XL ShieldI2C.zip

Na pasta do exemplo pode ser encontrado o programa: **LCDAssistant.exe** que converte imagens em vetores. O *datasheet* do *display* **SSD1306.pdf**. O arquivo **LSSD1306.rar** que contém uma biblioteca gráfica para o SSD1306 que pode ser facilmente portada para o kit. No exemplo foi feito uma portabilidade parcial desta biblioteca pelo professor, caso necessite de outras funções adicionais, estas deverão ser portadas no seu código ...

### Links úteis:

Dentro do arquivo "readme.txt" do exemplo podem ser encontrados os principais links adicionais para o entendimento do uso do *display*. Seguem aqui:

 $\frac{https://oledtutorials.blogspot.com/2020/06/tutorial-1-oled-basic-programming.html}{}$ 

https://www.youtube.com/watch?v=nZbA2qKRk30

https://www.youtube.com/watch?v=97 Vyph9EzM

https://mischianti.org/images-to-byte-array-online-converter-cpp-arduino/

https://javl.github.io/image2cpp/

https://github.com/rickkas7/DisplayGenerator

https://rickkas7.github.io/DisplayGenerator/index.html

https://xbm.jazzychad.net/

https://github.com/afiskon/stm32-ssd1306

https://github.com/lexus2k/ssd1306

https://controllerstech.com/oled-display-using-i2c-stm32/

-> oled.zip = \Docs\LSSD1606.rar

Para o exemplo do **PWM** utilizar como base o exemplo em: <a href="https://nuclearrambo.com/wordpress/generating-pwm-on-tiva-c-connected-launchpad-tm4c1294/">https://nuclearrambo.com/wordpress/generating-pwm-on-tiva-c-connected-launchpad-tm4c1294/</a>

Adaptar este código para o GPIO do LED que deverá sofrer o controle do brilho.

## Conteúdo ao arquivo (readme.txt):

Disciplina: ELF74
Laboratório: 4

Equipe: Nome do aluno 1, RA
Nome do aluno 2, RA

Data: XX/YY/ZZZZ

### Configuração de Diretórios:

Manter os projetos de todos os laboratórios no mesmo nível de diretório que o diretório da "*TivaWare*", também manter o nome do diretório original: "**TivaWare\_C\_Series-2.2.0.295**". Isto serve para que consiga compilar os exemplos do professor e vice-versa, quando entregar os laboratórios para avaliação.

Para entregar o Laboratório enviar um *email* para o professor com o diretório do projeto compactado em um arquivo .zip.

\*\*\* Este laboratório deverá ser mostrado em sala de aula em funcionamento junto ao professor para ser validada a nota. \*\*\*