12º Laboratório de Programação para Sistemas Embarcados

Prof. Otávio Gomes e Prof. Rodrigo Almeida

Leia com atenção - Informações iniciais:

1. No início de cada tópico/assunto é apresentado um **exercício de revisão** em que basta copiar o código na ferramenta, realizar a compilação e a execução e, então, interpretar o resultado. Este tipo de exercício tem como objetivo auxiliar o aluno a relembrar alguns conceitos e a validar as ferramentas que estão sendo utilizadas. Este código sempre estará correto e funcionando.
2. Os exercícios estão apresentados em **ordem crescente de dificuldade**.
3. Para o **registro de frequência**, o aluno deverá enviar o código relativo ao exercício mais difícil desta lista que conseguir resolver. Por exemplo:
   a. Se em uma lista contendo 6 exercícios o aluno A conseguiu resolver até o exercício 4, é este que ele deve enviar para registro de frequência.
   b. Se o aluno B conseguiu resolver toda a lista de exercícios, deve enviar o último exercício da lista.
4. Os exercícios abordam todos os conceitos relacionados ao conteúdo da aula em questão. Deste modo, caso o aluno não consiga resolver alguns dos exercícios, recomenda-se que o mesmo participe dos **plantões de dúvidas** e que busque aprender os conceitos envolvidos na atividade.
5. A **próxima atividade** de laboratório admitirá que os conceitos aqui apresentados já foram plenamente compreendidos.
6. A **entrega** desta atividade para o controle de frequência será realizada pelo SIGAA.

1) Crie um programa *main.*c com o código a seguir. O programa realiza a leitura do sinal do Potenciômetro e apresenta o resultado no LCD do PICSimLab.

```
#include "config.h"
#include "lcd.h"
#include "adc.h"

void main(void) {
    lcdInit();
    adcInit();
    int v;
    for (;;) {
        v = adcRead(0);
        lcdCommand(0x80);
        lcdString("Valor: ");
        lcdChar((v / 1000) % 10 + 48);
        lcdChar((v / 100) % 10 + 48);
        lcdChar((v / 10) % 10 + 48);
        lcdChar(v % 10 + 48);
    }
}
```

2) Crie um programa *main.*c com o código a seguir. O programa realiza o controle do *RGB* através do Potenciômetro e o controle do *Buzzer* através do teclado.

```
#include "config.h"
#include "lcd.h"
#include "adc.h"
#include "rgb.h"
#include "keypad.h"
#include "pwm.h"

void main(void) {
    lcdInit();
    adcInit();
    rgbInit();
    kpInit();
    pwmInit();
    int v, i=0;
    char slot=0;

    for (;;) {
      switch (slot) {
      case 0:
        v = adcRead(0);
        lcdCommand(0x80);
        lcdString("Valor: ");
        lcdChar((v / 1000) % 10 + 48);
        lcdChar((v / 100) % 10 + 48);
        lcdChar((v / 10) % 10 + 48);
        lcdChar(v % 10 + 48);

        rgbColor(v);
        slot = 1;
       break;

     case 1:
        kpDebounce();
        if (kpReadKey() != 0) {
            lcdChar(kpReadKey());
            if (kpReadKey() == 'S') {
              pwmSet(100);
            } else {
              pwmSet(0);
            }
        }
        slot = 0;
        break;

     default:
        slot = 0;
        break;
         }
    }
}
```

3) Crie uma ***biblioteca*** de controle de temperatura. Esta biblioteca utilizará as saídas PWM para controlar o *cooler* (não está presente no simulador). O valor do PWM deverá ser apresentado no display LCD.
A biblioteca deve possuir três funções: duas irão configurar os limites superior e inferior da temperatura (que deverão ser salvos dentro da biblioteca) e a terceira irá verificar o valor atual da temperatura e ligar o *cooler* de acordo com a necessidade, mantendo a temperatura atual dentro dos limites configurados.
Faça com que o acionamento do PWM seja gradual aumentando, aos poucos, a velocidade de rotação do *cooler*.
Os valores de configuração dos limites e o valor atual da temperatura deverão ser enviados através de comunicação serial.

```
void ConfiguraLimiteSuperior (char temp);
void ConfiguraLimiteInferior (char temp);
void AtualizarSistema (void);
```

Este programa deve exibir no LCD:

- A temperatura atual (**T**), lida periodicamente através da serial;
- Os limites superior (**H**) e inferior (**L**) de temperatura, lidos no início da execução, durante a fase de configuração;
- A porcentagem configurada na saída PWM do cooler (**C**) para o resfriamento do dispositivo.

Um exemplo de exibição é apresentado a seguir: