MICROPROCESSADORES II

Professor: Patric Janner Marques

Aula : Conceito de máquina de estados

Referências

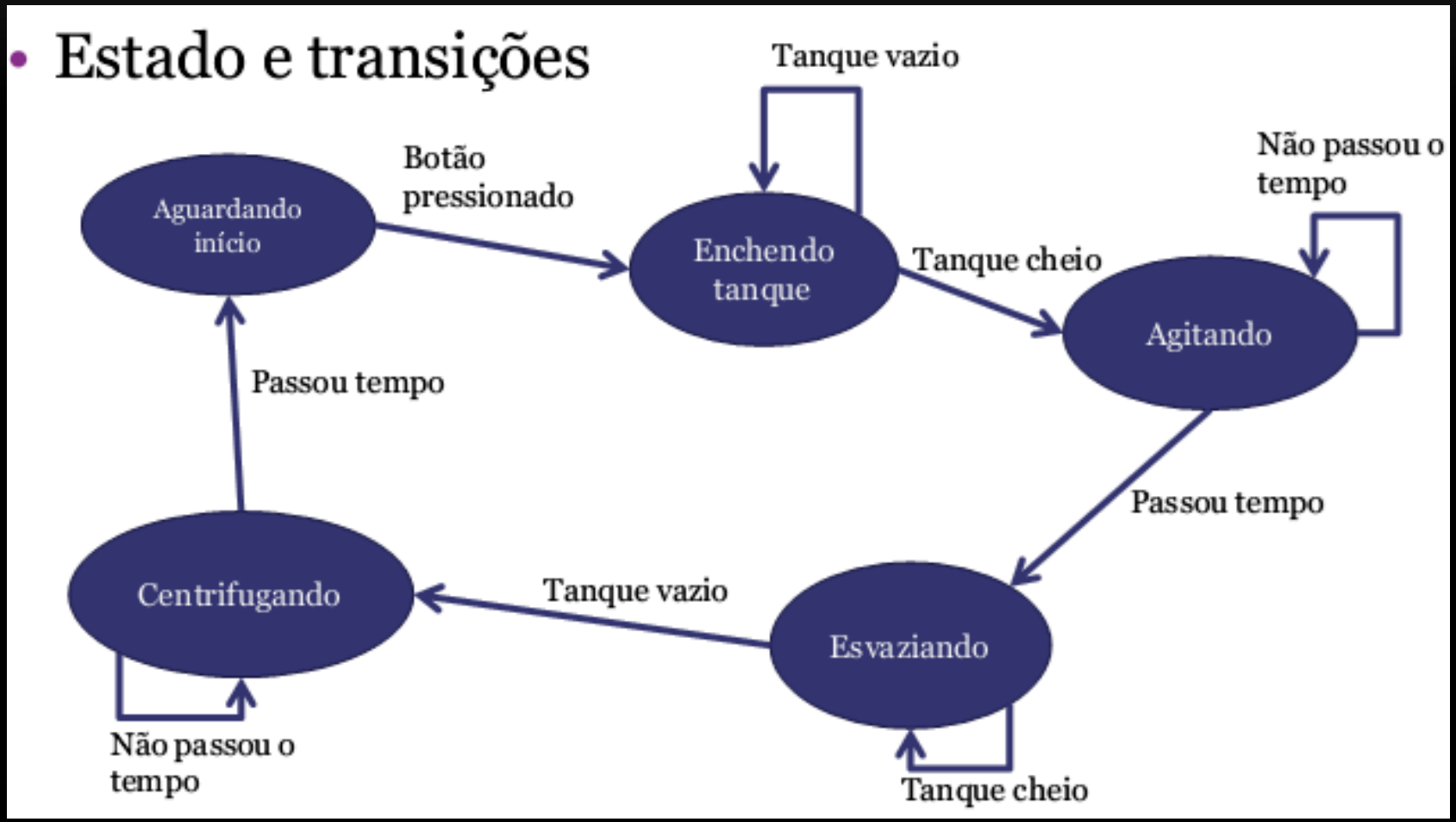
- https://sergioprado.org/maquina-de-estados-em-c/
- https://www.embarcados.com.br/maquina-de-estado/
- https://www.embarcados.com.br/arquitetura-de-desenvolvimento-de-software-parte-iii/

Conceito geral

- *Design patterns*, ou padrões de projeto, são soluções para problemas normalmente encontrados em projetos de *software*. São independentes de linguagem, e oferecem uma descrição ou *template* de como resolver determinado problema.
- No padrão de projeto conhecida como máquina de estados, cada estado representa uma situação onde o sistema realiza uma determinada tarefa, ou um conjunto de tarefas. A mudança do estado é dada quando alguma condição for satisfeita.
- É possível avançar, recuar ou permanecer em um estado, e o mecanismo deste fluxo da máquina de estado é definido pelo desenvolvedor e sua complexidade é tão grande quanto a complexidade do sistema desenvolvido.

Máquina de estados

- Estado e transições



Máquina de estados - Enum e switch-case

- Implementar uma máquina de estados utilizando Enumeração (enum) e switch-case. Exemplo:

```
typedef enum {
    ST_aguardando, ST_enchendo, ST_agitando, ST_esvaziando, ST_centrifugando
} Estados;
void main ()
{
    Estados estado= ST_aguardando;
    while(1)
    {
        switch (estado) {
          case ST_aguardando:
             …
           break;
……
```

Máquina de estados - Funções

- Implementar os estados com funções, onde:
  - ST_aguardando: deve aguardar a chave Iniciar ser pressionada;
  - ST_enchendo: simula o enchimento do tanque, acendendo o primeiro led de um conjunto de 8 leds, depois o segundo led até atingir o último, que corresponde ao tanque cheio;
  - ST_agitando: simula o agitamento do tanque, fazendo o primeiro led acender, de um outro conjunto de 8 leds, e, depois, apagar e acender o segundo led, assim sucessivamente, indo da direita para a esquerda e vice-versa, por um tempo determinado;

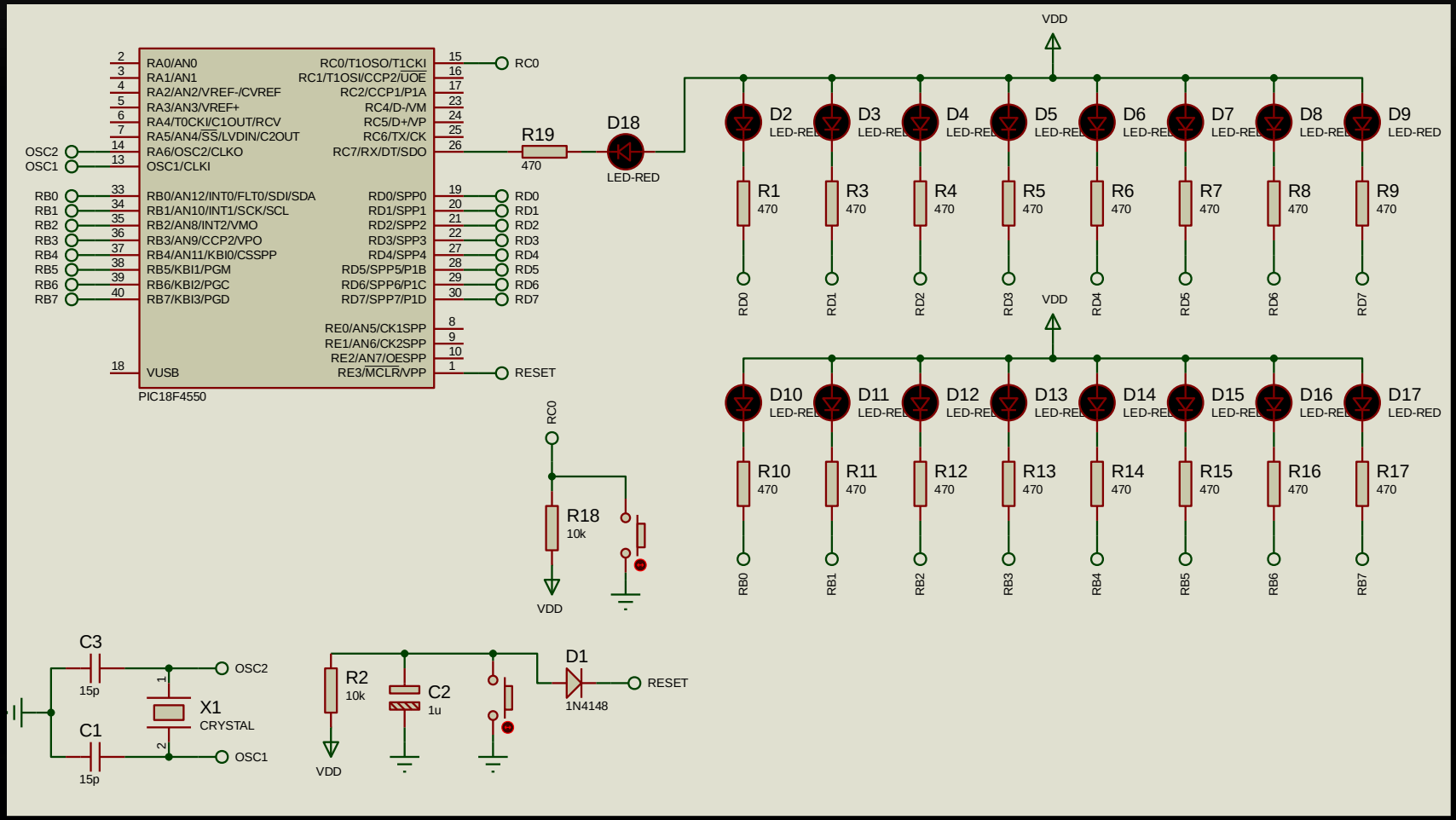
Máquina de estados - Funções

- Implementar os estados com funções, onde:
  - ST_esvaziando: simula o esvaziamento do tanque, apagando o último led do conjunto de 8 leds utilizado no estado ST_enchendo, depois o penúltimo led até atingir o primeiro, que corresponde ao tanque vazio;
  - ST_centrifugando: simula o processo de centrifugação, fazendo 4 leds menos significativos acenderem e 4 leds mais significativos apagarem, do mesmo conjunto de 8 leds utilizado no ST_agitando, e fique alternando-os entre ligado e desligado, diminuindo o intervalo de tempo entre as piscadas, por um tempo determinado.

Máquina de estados - Biblioteca

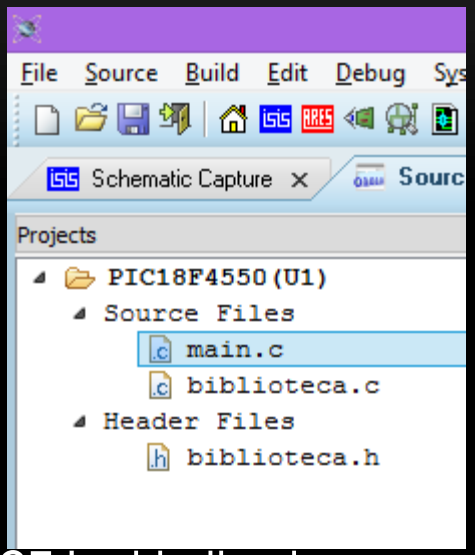
- Implementar os estados com funções, crie uma biblioteca, insira estas funções na biblioteca e adicione-a em seu projeto.
- Lembre-se de utilizar, como forma de exercício, diferente tipos de dados (de forma otimizada), qualificadores, como o *const*, classes de armazenamento, como *static* e *extern*, estruturas de dados e uniões.

Solução para Máquina de estados



Cuidar com as bibliotecas no Proteus

Main.c




```
#include "biblioteca.h"
extern unsigned char tempo_agito; // Variável global para
exemplificar a classe extern
void main(void)
{
   TRISB= 0;   // Setando toda a porta B como saída
   TRISD= 0;   // Setando toda a porta D como saída
   TRISC = 0b00000001; // Setando RC0 como entrada // RC7 Led indicador
   LED_START = 1;
   LATB=255;         // Inicializando os leds apagados
   LATD=255;         // Inicializando os leds apagados
   Estados estado_atual= ST_aguardando;
   unsigned char valor=1;
   const unsigned char tempo_max=30;
```

Main.c

```
while(1) {
    switch (estado_atual) {
            case ST_aguardando:
              if (CHAVE_INICIAR==0){
                estado_atual=ST_enchendo;
                LED_START = 0;          }
              break;
             case ST_enchendo:
               valor=Enchendo(valor);
               if (valor==9){
                 estado_atual=ST_agitando;
                 valor--;      }
               break;
             case ST_agitando:
               Agitando();
               if (tempo_agito==tempo_max){
                 estado_atual=ST_esvaziando;
                 LATD=0xff;
                 tempo_agito=0;          }
               break;
             case ST_esvaziando:
               valor=Esvaziando(valor);
               if (valor==0){
                 estado_atual=ST_centrifugando;
                 LATB=0xff;                  }
               break;
             case ST_centrifugando:
               Centrifugando();
               if (tempo_agito==tempo_max){
                 estado_atual=ST_aguardando;
                 LATD=0xff;
                 tempo_agito=0;
                 LED_START = 1;          }
               break;
    }   }   }
```

Biblioteca.h

```
#include <xc.h>
#include <math.h>
#define _XTAL_FREQ   20000000   // Usado como base para função __delay_ms()

// CONFIG1L
#pragma config PLLDIV = 1       // PLL Prescaler Selection bits (No prescale (4 MHz oscillator input drives PLL directly))
#pragma config CPUDIV = OSC1_PLL2// System Clock Postscaler Selection bits ([Primary Oscillator Src: /1][96 MHz PLL Src: /2])
#pragma config USBDIV = 1       // USB Clock Selection bit (used in Full-Speed USB mode only; UCFG:FSEN = 1) (USB clock source comes directly from the primary oscillator block with no postscale)

// CONFIG1H
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator (HS))
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)
#pragma config IESO = OFF       // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)

/////// PARA RESETAR O SISTE
#pragma config MCLRE = ON       // MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled)MA
```

Biblioteca.h

```
// Dados especiais e definicoes
typedef enum {ST_aguardando, ST_enchendo, ST_agitando, ST_esvaziando,
ST_centrifugando} Estados;

#define CHAVE_INICIAR RC0
#define LED_START RC7

// Prototipos das funções
void Mydelay(int tempo);
unsigned char Enchendo(unsigned char valor);
void Agitando(void);
unsigned char Esvaziando(unsigned char valor);
void Centrifugando(void);
```

Biblioteca.c

```
#include "biblioteca.h"

unsigned char tempo_agito=0; // Variável global para exemplificar a classe extern

void Mydelay(int tempo)
{
   int i;
   for (i=0;i<tempo;i++)
   {
      __delay_ms(1);
   }
}
unsigned char Enchendo(unsigned char valor)
{
   unsigned char aux_char;
   float aux_float = (float)valor;
   aux_float = pow(2,aux_float)-1;
   aux_char = (unsigned char)aux_float;
   LATB=~aux_char;
   Mydelay(700);   // espera 700 ms
   return ++valor;
}
```

Biblioteca.c

```
void Agitando(void) {
   static bit controle=0;
   static char Porta_D=0b01111111;
   if (controle==0)    {
      Mydelay(400);   // espera 400 ms
      Porta_D = (Porta_D << 1) | (Porta_D >> 7);
      LATD=Porta_D;    }
   else    {
      Mydelay(400);   // espera 400 ms
      Porta_D = (Porta_D >> 1) | (Porta_D << 7);
      LATD=Porta_D;    }
   if (RD0==0)      controle=0;
   if (RD7==0)      controle=1;
   tempo_agito++;
}
unsigned char Esvaziando(unsigned char valor) {
   unsigned char aux_char;
   float aux_float = (float)valor;
   aux_float = pow(2,aux_float)-1;
   aux_char = (unsigned char)aux_float;
   LATB=~aux_char;
   Mydelay(700);   // espera 700 ms
   return --valor;
}
```

```
void Centrifugando(void)
{
   static char Porta_D=0b11110000;
   LATD=Porta_D;
   Porta_D =~Porta_D;
   Mydelay(500-(tempo_agito*15));   // espera variavel
   tempo_agito++;
}
```