Inatel Instituto Nacional de Telecomunicações	**RELATÓRIO 2**	**Data: / /**
	Disciplina: E209	
	Profs: João Magalhães e Yvo Chiaradia **Monitores:** **Thalita Domingos, Diego Coutinho, Pedro Fraga, Thayana Lucero e Ewel Fernandes**	
Conteúdo: Revisão de Linguagem C		
Tema: Revisão Linguagem C e Máquina de Estados		
Nome:	**Matrícula:**	**Curso:**

OBJETIVOS:

- Desenvolver e aplicar os conceitos de máquinas de estados finitos;
- Extrair um diagrama de estados de um programa em C;
- Desenvolver um programa em C a partir de um diagrama de estados.

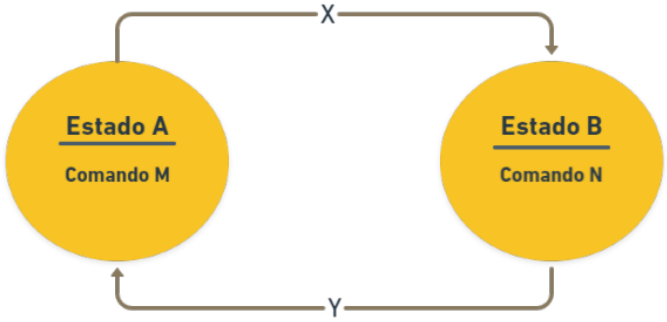
Parte Teórica

Definição de Máquinas de Estado:

Uma máquina de estados finitos é um modelo matemático usado para representar programas de computadores ou circuitos lógicos. O conceito é concebido como uma máquina abstrata que deve estar em um de seus finitos estados. **A máquina está em apenas um estado por vez** e este estado é chamado de **estado atual**. Uma **transição** indica uma mudança de estado e ocorre quando uma condição for satisfeita.

Diagramas de Estado

A partir dos diagramas de estado podemos desenvolver as lógicas de controle a serem implementadas. Os diagramas de estado assemelham-se ao seguinte modelo:



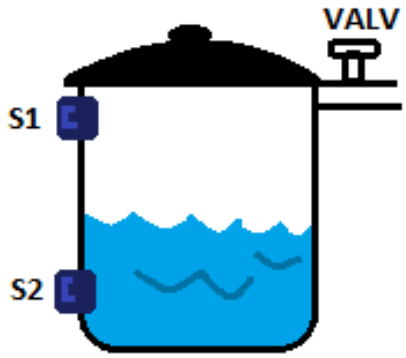
Onde temos:

- **X** e **Y** = transições -> condições necessárias para que ocorra transição de um estado para o outro;
- No **estado A** é executado o **Comando M** e no **estado B** é executado o **Comando N;**
- Se a máquina estiver no **estado A** ***E*** a **condição X for verdadeira**, o sistema irá mudar do **estado A para o estado B** (transição de A para B);
- Se a máquina estiver no **estado B** ***E*** a **condição Y for verdadeira**, o sistema irá mudar do **estado B para o estado A** (transição de B para A).

Exemplo de Máquina de Estado aplicada:

No exemplo a seguir é ilustrado o funcionamento de um sistema de controle de nível de um reservatório de água. O diagrama de estados descreve a lógica de abertura da válvula a partir de dois sensores de nível.

Deseja-se desenvolver um circuito que faça o controle de nível de um reservatório. Existem 2 sensores, MAX (S1) e MIN (S2), e uma válvula VALV. Quando o nível estiver abaixo do mínimo, a válvula é ligada. Ela continua acionada até que o nível ultrapasse o valor máximo.



Assim, para resolver o problema, o primeiro passo é identificar os estados do sistema: **vazio** e **cheio**.

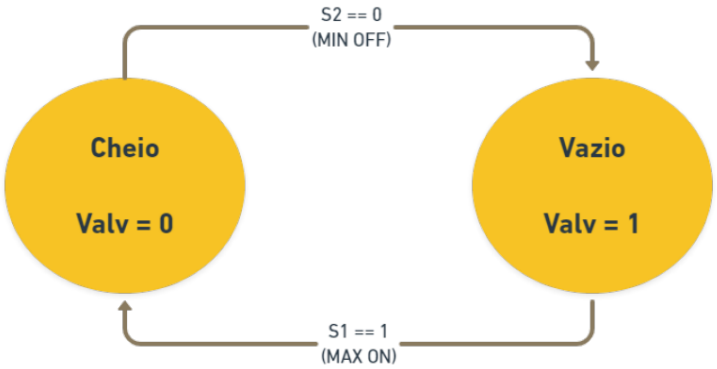
O próximo passo é **identificar a(s) saída(s)**: válvula (VALV), assim é possível verificar o que ocorre com o(s) valor(es) da(s) saída(s) para cada estado: no **vazio** a **VALV** fica ligada (1) e no **cheio** a **VALV** fica desligada (0).

O passo seguinte é **identificar a(s) entrada(s):** sensor **MAX(S1)** e sensor **MIN(S2)**.

O final é analisar o problema e **determinar quais condições das entradas fazem com que ocorra a transição de estados**. Ambos sensores são ALTO-ATIVO, ou seja, fornecem nível lógico alto (1) quando ativos.

Por exemplo: sensor **MAX(S1)** em nível lógico alto (S1==1) indica o reservatório **cheio** e sensor **MIN(S2)** em nível lógico baixo(S2==0) indica o reservatório **vazio**.

A figura a seguir demonstra o diagrama de estados para o problema.



Implementação da máquina de estados em linguagem C:

```
#define CHEIO 5
#define VAZIO 0

unsigned char estado_atual = VAZIO, sensor_min = 0, sensor_max = 0, valvula = 1;

int main(void)
{
   for (;;)
   {
      switch (estado_atual)
      {
        case CHEIO:
                        valvula = 0;
                        if (sensor_min == 0)
                                estado_atual = VAZIO;
                        break;
        case VAZIO:
                        valvula = 1;
                        if (sensor_max == 1)
                                estado_atual = CHEIO;
                        break;
        default:
                        break;
      }
   }
}
```

OBS.: Note que a saída não é alterada diretamente pelo valor de entrada, mas sim pelo valor do **estado_atual** da máquina de estados, uma variável do tipo char não sinalizada (positiva).

Parte Prática

1) Para representar o funcionamento de um semáforo de trânsito, crie uma máquina de estados, sabendo que**:**
O tempo de cada estado é: Verde – 12s / Amarelo – 3s / Vermelho – 15s

a) Elabore o diagrama de estados que satisfaça a operação do semáforo. As transições e saídas podem ser representadas de maneira simples

b) Desenvolva um programa em C que obedeça ao diagrama de estados do item 2) a). Utilize variáveis com nomes objetivos para ilustrar os estados. Compile o projeto e execute para conferir a lógica elaborada na máquina de estados.
Para isso, use os pinos 2, 3 e 4 do Arduino Uno e o programa disponibilizado pelo monitor.

Dicas:
Para fazer com que o programa espere um tempo **T**, utilize a função **_delay_ms(T em ms)** -> ex: **_delay_ms(500)** para **500ms**.

2) Analise o **programa anexo** e extraia o diagrama de estados para as condições previstas. Represente cada transição com o valor de entrada e em cada estado o valor da saída.

3) Exercício Proposto:
Elabore um diagrama de estados de um dispositivo ou processo à sua escolha.
-> ex: Funcionamento de uma lâmpada, de uma porta, de uma máquina de café, etc.

ANEXO) PROGRAMA POR MÁQUINA DE ESTADOS

```
#define BUTTON_ON !(PIND & 0b00010000)

#define LED_RED_ON PORTD = PORTD | 0b10000000
#define LED_RED_OFF PORTD = PORTD & ~(0b10000000)
#define LED_GREEN_ON PORTD = PORTD | 0b00100000
#define LED_GREEN_OFF PORTD = PORTD & ~(0b00100000)

#define DELAY _delay_ms(500)
char estado = 0;

int main(void)
{
   DDRD = DDRD | 0b10100000;   // Configurando pino 5 e 7 como saída
   PORTD = PORTD | 0b00010000; // Habilita resistor de PULL-UP
   for (;;)
   {
      switch (estado)
      {
      case 0:
         LED_RED_OFF;   //desliga led do pino 7
         LED_GREEN_OFF; //desliga led do pino 5
         if (BUTTON_ON)
         {
            estado = estado + 1; //incrementa o estado
            DELAY;               // delay para evitar o Bouncing
         }
         break;

      case 1:
         LED_RED_ON;    //liga led do pino 7
         LED_GREEN_OFF; //desliga led do pino 5
         if (BUTTON_ON)
         {
            estado = estado + 1; //incrementa o estado
            DELAY;               // delay para evitar o Bouncing
         }
         break;

      case 2:
         LED_RED_OFF;  //desliga led do pino 7
         LED_GREEN_ON; //liga led do pino 5
         if (BUTTON_ON)
         {
            estado = estado + 1; //incrementa o estado
            DELAY;               // delay para evitar o Bouncing
         }
         break;


      case 3:
         LED_RED_ON;   //liga led do pino 7
         LED_GREEN_ON; //liga led do pino 5
         if (BUTTON_ON)
```

```
        {
           estado = estado + 1; //incrementa o estado
           DELAY;             // delay para evitar o Bouncing
        }
        break;

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