

Universidade Tecnológica Federal do Paraná – Toledo
Engenharia da Computação – COENC

Sistemas Embarcados

Padrões de projeto Máquinas de estados

Tiago Piovesan Vendruscolo



Esta licença permite que outros remixem, adaptem e criem a partir do trabalho para fins não comerciais, desde que atribuam o devido crédito aos autores originais. [4.0 international](https://creativecommons.org/licenses/by-nc-nd/4.0/)

- Existem duas formas tradicionais para implementar máquinas de estados
 - Switch / Case: Mais simples e normalmente mais usado, porém, conforme a máquina de estados vai ganhando complexidade, fica cada vez mais difícil de manter a organização e realizar mudanças.
 - Ponteiros de funções: É uma forma de dividir a máquina de estados em módulos, o que torna a manutenção mais fácil mesmo em máquina complexas. Também mantém o código mais limpo.

Máquinas de estados - Ponteiros de funções

Estados

```
//Ponteiro de função da máquina de estados.  
//Ele aponta sempre para a função da máquina de estados que deve ser executada  
void (*PonteiroDeFuncao) ();  
  
//prototipos das funções de cada estado  
void ESTADO_1(void);  
void ESTADO_2(void);  
void ESTADO_3(void);  
void ESTADO_4(void);  
  
void ESTADO_1(void)  
{  
    digitalWrite(LED1, HIGH);  
    digitalWrite(LED2, LOW);  
    digitalWrite(LED3, LOW);  
    digitalWrite(LED4, LOW);  
    if(digitalRead(Botao)){  
        PonteiroDeFuncao = ESTADO_2;  
    }  
    else  
    {PonteiroDeFuncao = ESTADO_4;}  
}  
  
void ESTADO_2(void)  
{  
}  
  
void ESTADO_3(void)  
{  
}  
  
void ESTADO_4(void)  
{  
}  
  
void loop()  
{  
    //aponta para o estado inicial. Nunca esquecer de informar um estado inicial  
    //(senão, nesse caso em específico, pode haver um erro fatal / hard fault).  
    PonteiroDeFuncao = ESTADO_1;  
  
    while(1)  
    {  
        //chama a função apontada pelo ponteiro de função (logo, chama o estado corrente)  
        (*PonteiroDeFuncao) ();  
    }  
}
```

Leitura do botão dentro de
cada função de estado

Máquinas de estados - Ponteiros de funções

- Exercício 1: Faça o controle de um motor de passo utilizando passo completo 1, utilizando os LED1 a 4 (pinos 4 a 7). Quando a entrada (pino 2 - PULL_UP) estiver em nível lógico “1” (não é interrupção), o motor deve girar no sentido LED1 – LED4, caso contrário, no sentido inverso. (Lembre de por o Delay)

Código no moodle

- Passo completo 1 (Full Step)**

Nº do passo	B3	B2	B1	B0	Decimal
1	1	0	0	0	8
2	0	1	0	0	4
3	0	0	1	0	2
4	0	0	0	1	1

Máquinas de estados - Ponteiros de funções

```
void ESTADO_4(void)
{
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    digitalWrite(LED4, HIGH);
    if(digitalRead(Botao)){
        PonteiroDeFuncao = ESTADO_1;
    }
    else
        {PonteiroDeFuncao = ESTADO_3;}
}

void loop()
{
    PonteiroDeFuncao = ESTADO_1;
    while(1)
    {
        (*PonteiroDeFuncao)();
        delay(300);
    }
}
```

Máquinas de estados - Ponteiros de funções

- Exercício 2: Faça o controle de um motor de passo utilizando passo completo 1, utilizando os LED1 a 4 (pinos 4 a 7). Quando a entrada (pino 2 - PULL_UP) estiver em nível lógico “1”, o motor deve girar no sentido LED1 – LED4, caso contrário, no sentido inverso. Utilize a serial para alterar o Delay entre os estados.

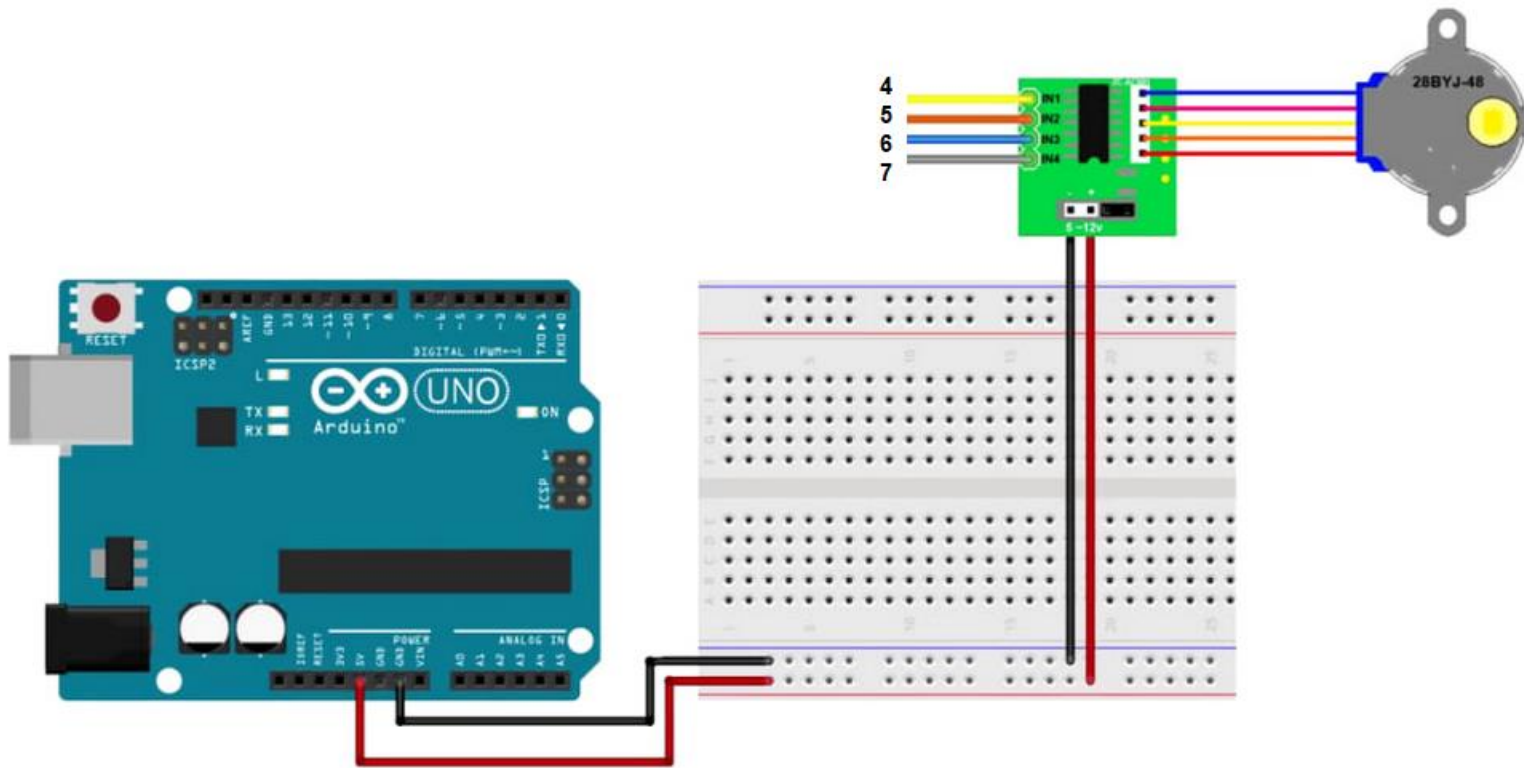
```
void loop()
{
    PonteiroDeFuncao = ESTADO_1; //aponta para o estado in

    while(1)
    {
        if (Serial.available() > 0) {
            x = Serial.parseInt();
            Serial.println(x);

            (*PonteiroDeFuncao)();    //chama a função apontada
            delay(x);
        }
    }
}
```

Máquinas de estados - Ponteiros de funções

- Exercício 3: Teste o exercício 2 utilizando o motor de passo no lugar dos LEDs. (Utilize 2 ms como velocidade máxima).



Máquinas de estados - Ponteiros de funções

- Exercício 4: Continuando o exercício 2, acrescente um botão de emergência (pino 3 – interrupção) que quando pressionado, pare o motor imediatamente (todos os LEDs desligados), e quando solto, continue de onde tinha parado.

```
void (*PonteiroDeFuncao)(); //ponteiro de função da máquina de estados.  
void (*temp)(); // posição da máquina de estados antes do stop
```

```
//prototypes:
```

```
void ESTADO_1(void); //função que representa o estado inicial da máq  
void ESTADO_2(void);  
void ESTADO_3(void);  
void ESTADO_4(void);  
void ESTADO_5(void);
```

```
void Botao_ISR();
```

```
void setup() {  
    pinMode(LED1, OUTPUT);  
    pinMode(LED2, OUTPUT);  
    pinMode(LED3, OUTPUT);  
    pinMode(LED4, OUTPUT);  
    pinMode(Botao, INPUT_PULLUP);  
    pinMode(Botao_Stop, INPUT_PULLUP);  
    attachInterrupt(digitalPinToInterrupt(Botao_Stop), Botao_ISR, CHANGE);  
    Serial.begin(9600);  
}
```

```
void Botao_ISR() {  
    if (!digitalRead(Botao_Stop)) {  
        temp = PonteiroDeFuncao;  
        PonteiroDeFuncao = ESTADO_5; }  
    else {  
        PonteiroDeFuncao = temp; }  
}
```


Inserindo na interrupção, é preciso fazer o debounce do botão, caso contrário, a máquina poderá ficar presa no estado 5 devido aos ruídos

Máquinas de estados - Ponteiros de funções

- Exercício 4: Continuando o exercício 2, acrescente um botão de emergência (pino 3 – interrupção) que quando pressionado, pare o motor imediatamente (todos os LEDs desligados), e quando solto, continue de onde tinha parado.

Inserindo nos estados 1 a 4, é garantido que a máquina que não fique presa.

```
void ESTADO_1(void)
{
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    digitalWrite(LED4, LOW);
    temp = PonteiroDeFuncao;
    if(digitalRead(Botao)) {
        PonteiroDeFuncao = ESTADO_2;
    }
    else
    {PonteiroDeFuncao = ESTADO_4;}
}
```



Máquinas de estados - Ponteiros de funções

- Exercício 5: Continuando o exercício 2, acrescente um botão de emergência (pino 3 – interrupção) que quando pressionado, pare o motor (todos os LEDs desligados) quando chegar no estado 4, e quando solto, comece no estado 1.

```
void Botao_ISR() {  
    if (!digitalRead(Botao_Stop)) {  
        x=0;}  
    else{  
        PonteiroDeFuncao = ESTADO_1;  
        x=1;}  
}  
  
void ESTADO_4(void)  
{  
    digitalWrite(LED1, LOW);  
    digitalWrite(LED2, LOW);  
    digitalWrite(LED3, LOW);  
    digitalWrite(LED4, HIGH);  
    if(digitalRead(Botao)) {  
        PonteiroDeFuncao = ESTADO_1;}  
    else  
        {PonteiroDeFuncao = ESTADO_3;}  
    if(x==0){  
        PonteiroDeFuncao = ESTADO_5;  
    }  
}
```

- Máquinas de estados são bastante úteis para a organização de código em sistemas embarcados.
 - *Switch/Case – Máquinas de estados simples, com poucos estados*
 - *Ponteiro de funções – Máquinas de estados maiores e mais complexas*
- Sempre que seja necessário fazer a leitura de entradas externas, optar por interrupções para não afetar o desempenho do código.
 - *Não faça grandes processamentos dentro da função de interrupção*
 - *Quando utilizar várias interrupções, classificar por prioridade*

Referências

- <https://sergioprado.org/maquina-de-estados-em-c/>
- <https://www.embarcados.com.br/maquina-de-estado/>