

Sistemas Embarcados

FreeRTOS - Tarefas -

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/)

Passagem de parâmetros para uma tarefa

Passagem de parâmetros para uma tarefa

- Exemplo 3: Com base no exercício 1 (aula anterior), utilize o parâmetro LED1 (pino 7) na criação da tarefa.

```
void vTarefa1(void *pvParameters);

void setup() {
    Serial.begin(9600);
    xTaskCreate(vTarefa1, "Tarefa1",128, (void*)LED1, 1, &Tarefa1Handle);
    xTaskCreate(vTarefa2, "Tarefa2",128, NULL, 2, &Tarefa2Handle);
    xTaskCreate(vTarefa3, "Tarefa3",128, NULL, 3, &Tarefa3Handle);
}

void vTarefa1(void *pvParameters)
{
    int LEDx = (int)pvParameters;
    pinMode(LEDx, OUTPUT);
    while(1)
    {
        digitalWrite(LEDx, !digitalRead(LEDx));
        vTaskDelay(pdMS_TO_TICKS(500));
    }
}
```

- Fazendo dessa forma, é possível alterar o LED sem ter que modificar dentro da função da tarefa

Múltiplas instâncias

- Exemplo 4: Instancie a função vTarefa1 duas vezes (tire a vTarefa2)

```
#define LED1 7
#define LED2 8

/* Variáveis para armazenamento do handle das tasks
TaskHandle_t Tarefa1Handle = NULL;
TaskHandle_t Tarefa2Handle = NULL;
TaskHandle_t Tarefa3Handle = NULL;

/*protótipos das Tasks*/
void vTarefa1(void *pvParameters);
/*void vTarefa2(void *pvParameters);*/
void vTarefa3(void *pvParameters);

void setup() {
    Serial.begin(9600);
    xTaskCreate(vTarefa1, "Tarefa1",128, (void*)LED1, 1, &Tarefa1Handle);
    xTaskCreate(vTarefa1, "Tarefa2",128, (void*)LED2, 2, &Tarefa2Handle);
    xTaskCreate(vTarefa3, "Tarefa3",128, NULL, 3, &Tarefa3Handle);
}
```

```
void vTarefa1(void *pvParameters)
{
    int LEDx = (int)pvParameters;
    pinMode(LEDx, OUTPUT);
    while(1)
    {
        digitalWrite(LEDx,!digitalRead(LEDx));
        vTaskDelay(pdMS_TO_TICKS(500));
    }
}
```

vTarefa2 não será utilizada

Duas tarefas utilizando a mesma função

- A função permanece a mesma, os dois LED irão piscar juntos.

Múltiplas instâncias

- Exemplo 5: Utilizando duas instâncias e passando dois parâmetros para cada instância. (Instância 1: LED1 em 1Hz, instância 2: LED2 em 0,5Hz). Faça testes reduzindo o delay dos LEDs e mantendo todas as tarefas com a mesma prioridade.

```
#define LED1 7
#define LED2 8

typedef struct pisca_led
{
    uint16_t Tempo_ms;
    uint8_t pino;
} led_t;

led_t led1 = {500, LED1}; //LED1: delay 500ms (1Hz), pino 7
led_t led2 = {1000, LED2}; //LED2: delay 1000ms (0,5Hz), pino 8

/* Variáveis para armazenamento do handle das tasks*/
TaskHandle_t Tarefa1Handle = NULL;
TaskHandle_t Tarefa2Handle = NULL;

/*protótipos das Tasks*/
void vTarefa1(void *pvParameters);

void setup()
{
    Serial.begin(9600);
    xTaskCreate(vTarefa1, "Tarefa1", 128, (void *)&led1, 1, &Tarefa1Handle);
    xTaskCreate(vTarefa1, "Tarefa2", 128, (void *)&led2, 1, &Tarefa2Handle);
}
```

Múltiplas instâncias

```
void loop()
{
}

void vTarefa1(void *pvParameters) // Mesma função chamada com 2 tarefas diferentes
{
    led_t *led = (led_t *)pvParameters; //faz a leitura da struct passada
    pinMode(led->pino, OUTPUT);          //configura pino como saída
    while (1)
    {
        Serial.println("Piscando o LED no pino: " + String(led->pino));
        digitalWrite(led->pino, !digitalRead(led->pino));
        vTaskDelay(pdMS_TO_TICKS(led->Tempo_ms));
    }
}
```

Exemplo no MOODLE

```
Piscando o LED no pino: 8
Piscando o LED no pino: 7
Piscando o LED no pino: 7
Piscando o LED no pino: 8
Piscando o LED no pino: 7
Piscando o LED no pino: 7
```

Múltiplas instâncias

- Exercício 5: Adicione uma terceira instância no exemplo anterior controlando o LED3 no pino 9. (Instância 1: LED1 em 1Hz, instância 2: LED2 em 0,5Hz, instância 3: LED3 em 5Hz).

```
#define LED1 7
#define LED2 8
#define LED3 9

typedef struct pisca_led
{
    uint16_t Tempo_ms;
    uint8_t pino;
} led_t;

led_t led1 = {500, LED1}; //LED1: delay 500ms (1Hz), pino 7
led_t led2 = {1000, LED2}; //LED2: delay 1000ms (0,5Hz), pino 8
led_t led3 = {100, LED3}; //LED3: delay 100ms (5Hz), pino 9

/* Variáveis para armazenamento do handle das tasks*/
TaskHandle_t Tarefa1Handle = NULL;
TaskHandle_t Tarefa2Handle = NULL;
TaskHandle_t Tarefa3Handle = NULL;

/*protótipos das Tasks*/
void vTarefa1(void *pvParameters);

void setup()
{
    Serial.begin(9600);
    xTaskCreate(vTarefa1, "Tarefa1", 128, (void *)&led1, 1, &Tarefa1Handle);
    xTaskCreate(vTarefa1, "Tarefa2", 128, (void *)&led2, 1, &Tarefa2Handle);
    xTaskCreate(vTarefa1, "Tarefa3", 128, (void *)&led3, 1, &Tarefa3Handle);
}
```


Debounce

- Podemos fazer um debounce simples utilizando a `vTaskDelay` dentro de uma tarefa

```
void vTarefa3(void *pvParameters)
{
    int debouncingContagem = 0;

    while(1)
    {
        if(digitalRead(BOTAO) == LOW)
        {
            debouncingContagem++;
            if(debouncingContagem >= 10) {
                debouncingContagem = 0;
                //
                // Faça algo
                //
            }
        }
        else
        {
            debouncingContagem = 0;
        }
        vTaskDelay(pdMS_TO_TICKS(20));
    }
}
```

Não utilize valores muito baixos, vai ser limitado pelo timer tick do sistema



- Exercício 6: Usando como base o exercício 5, modifique o código da tarefa 3 de forma que quando um botão no pino 2 for pressionado, ele suspenda a tarefa 1 e, quando pressionado novamente, reative a tarefa 1. Use um Debounce com tempo em torno de 200 ms.

Debounce

```
void vTarefa3(void *pvParameters)
{
    int debouncingContagem = 0;
    int x = 0;
    int estado_anterior = HIGH;
    while(1)
    {
        if(digitalRead(BOTAO) == LOW)
        {
            debouncingContagem++;
            if(debouncingContagem >= 10 && x == 0 && estado_anterior == HIGH) {
                debouncingContagem = 0;
                Serial.println("Suspendendo tarefa 1");
                vTaskSuspend(Tarefa1Handle);
                digitalWrite(LED1,LOW);
                x=1;
                estado_anterior = LOW;
            }
            else if (debouncingContagem >= 10 && x == 1 && estado_anterior == HIGH){
                debouncingContagem = 0;
                Serial.println("Reativando tarefa 1");
                vTaskResume(Tarefa1Handle);
                digitalWrite(LED1,LOW);
                x=0;
                estado_anterior = LOW;
            }
        }
        else
        {
            debouncingContagem = 0;
            estado_anterior = HIGH;
        }
        vTaskDelay(pdMS_TO_TICKS(20));
    }
}
```

- Exercício 7: Crie uma tarefa 4 que controle frequência do LED1 de 1Hz até 10Hz de acordo com uma entrada analógica. Utilize a função MAP.

```
void vTarefa4(void *pvParameters)
{
while (1)
{
    valor_pot = map(analogRead(POT), 0, 1023, 50, 500);


    Serial.println(valor_pot);
    led1.Tempo_ms = valor_pot;
    vTaskDelay(pdMS_TO_TICKS(500));
}
}
```

Escolhendo o núcleo de processamento

- Caso o microcontrolador tiver mais de um núcleo de processamento, é possível escolher em qual núcleo a tarefa irá ser executada. Para o caso do ESP32, o Port do FreeRTOS possui a seguinte API:

```
xTaskCreatePinnedToCore(vTarefa1, "Tarefa1", 128, NULL, 1, NULL, APP_CPU_NUM);
```

Definições:



```
57  
58 #define PRO_CPU_NUM (0)  
59 #define APP_CPU_NUM (1)  
60
```

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html>

- FreeRTOS – Semáforos/MUTEX

Referências

- <https://www.freertos.org/>
- https://www.freertos.org/fr-content-src/uploads/2018/07/FreeRTOS_Reference_Manual_V10.0.0.pdf
- https://www.freertos.org/fr-content-src/uploads/2018/07/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf
- <https://www.embarcados.com.br/>