

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

## **Sistemas Embarcados**

# **FREERTOS**

## **- Filas -**

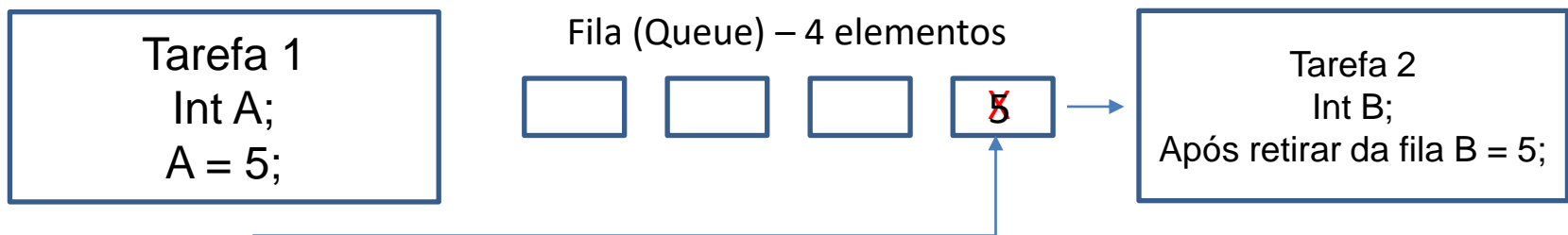
**Tiago Piovesan Vendruscolo**



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/)

# Fila de mensagens (Queue)

- Definição:
  - *Estrutura de dados dinâmicas onde é possível inserir ou remover elementos. Sendo que o elemento a ser removido da fila é o primeiro elemento que entrou na fila (FIFO: First In First Out).*
- Características
  - *Troca de mensagens entre tarefas ou entre uma interrupção e uma tarefa*
  - *Pode ser acessada por diversas tarefas e interrupções*
  - *O tamanho é definido no processo de criação da fila.*



# Fila de mensagens (Queue)

- *A fila de mensagens permite guardar (por meio de uma fila) mensagens até que o destinatário esteja pronto para recebe-las, operando com um buffer.*
- *Exemplo: tratamento de um teclado. Em alguns casos, a interface gráfica do SO congela enquanto o usuário está digitando. Quando a interface gráfica volta ao estado normal, aparecem todas as teclas pressionadas durante o congelamento.*
  - *Nota: para isso, o tratamento das teclas precisa ter uma prioridade mais alta.*

# Utilizando filas (Queue)

- Incluir queue.h `#include "queue.h"`

- Criando uma fila

```
QueueHandle_t xQueueCreate(UBaseType_t uxQueueLength, UBaseType_t uxItemSize);
```

Número de elementos da fila

Tamanho de cada elemento

- Testar se foi criada:

```
if( xQueue1 == NULL )  
{  
    /* Queue was not created and must not be used. */  
}
```

<https://www.freertos.org/a00116.html>

- Deletando uma fila – libera a memória

```
void vQueueDelete( QueueHandle_t xQueue );
```

Delete a queue - freeing all the memory allocated for storing of items placed on the queue.

## Parameters:

*xQueue* A handle to the queue to be deleted.

# Utilizando filas (Queue)

- Resetando uma fila (limpando a fila)

```
BaseType_t xQueueReset( QueueHandle_t xQueue );
```

Resets a queue to its original empty state.

<https://www.freertos.org/a00018.html>

- Escrevendo um dado em uma fila

```
BaseType_t xQueueSend( QueueHandle_t xQueue, const void * pvItemToQueue, TickType_t xTicksToWait );
```

Handle da fila

Dado a ser escrito

Tempo de espera para a fila estar disponível (timeout). pdMS\_TO\_TICKS(50). portMAX\_DELAY para aguardar indefinidamente.

- Quando você precisa enviar um dado direto para a saída

```
BaseType_t xQueueSendToFront( QueueHandle_t xQueue, const void * pvItemToQueue, TickType_t xTicksToWait );
```

# Utilizando filas (Queue)

- Recebendo um dado em uma fila

```
BaseType_t xQueueReceive(QueueHandle_t xQueue, void *pvBuffer, TickType_t xTicksToWait);
```

Handle da fila

Variável que irá armazenar o dado lido

Tempo de espera para a fila estar disponível (timeout). pdMS\_TO\_TICKS(50). portMAX\_DELAY para aguardar indefinidamente.

# Utilizando filas (Queue)

- Exemplo 1: Envie uma informação de uma tarefa para outra. O código terá 3 tarefas:
  - *Tarefa 1: LED piscando em 1Hz*
  - *Tarefa 2: Enviar o valor de um contador para a tarefa 3 através de uma fila com 5 elementos. Imprimir o valor atual do contador*
  - *Tarefa 3: Imprimir o valor recebido da tarefa 2*
  - *Arquivo no moodle*

# Utilizando filas (Queue)

- Exemplo 1: Envie uma informação de uma tarefa para outra. O código terá 3 tarefas:
  - *Tarefa 1: LED piscando em 1Hz*
  - *Tarefa 2: Enviar o valor de um contador de segundos para a tarefa 3 através de uma fila com 5 elementos. Imprimir o valor atual do contador*
  - *Tarefa 3: Imprimir o valor recebido da tarefa 2*

```
#include "Arduino_FreeRTOS.h"
```

```
#include "task.h"
```

```
→ #include "queue.h"
```

```
#define LED1 7
```

```
/* Variáveis para armazenamento do handle das tasks */
```

```
TaskHandle_t xTarefa1Handle = NULL;
```

```
TaskHandle_t xTarefa2Handle = NULL;
```

```
TaskHandle_t xTarefa3Handle = NULL;
```

```
/* Variáveis para armazenamento do handle das filas */
```

```
→ QueueHandle_t xFila;
```




# Utilizando filas (Queue)

## ■ Exemplo 1:

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

void setup() {
    Serial.begin(9600);
    xFila = xQueueCreate(5, sizeof(int));
    xTaskCreate(vTarefa1, "Tarefa1",128, NULL, 3, &xTarefa1Handle);
    xTaskCreate(vTarefa2, "Tarefa2",256, NULL, 2, &xTarefa2Handle);
    xTaskCreate(vTarefa3, "Tarefa3",128, NULL, 1, &xTarefa3Handle);
}
```


Número de elementos



# Utilizando filas (Queue)


## Exemplo 1:

```
void vTarefa2(void *pvParameters){  
    int contador = 0;  
    while(1)  
    {  
        xQueueSend(xFila, &contador, pdMS_TO_TICKS(1000));  
        Serial.println("Valor atual da tarefa 2: " + String(contador));  
        contador++;  
        vTaskDelay(pdMS_TO_TICKS(1000));  
    }  
}
```



Envio dos elementos

```
void vTarefa3(void *pvParameters){  
    int valorcontador = 0;  
    while(1)  
    {  
        if(xQueueReceive(xFila, &valorcontador, pdMS_TO_TICKS(1000)) == pdTRUE){  
            Serial.println("Tarefa 3 imprimindo contador da tarefa 2: " + String(valorcontador));  
        }  
        else{  
            Serial.println("Tarefa 3 TIMEOUT");  
        }  
        vTaskDelay(pdMS_TO_TICKS(1000));  
    }  
}
```



Recebimento dos elementos

# Utilizando filas (Queue)

## ■ Exemplo 1:

### Resultado:

```
-> Valor atual da tarefa 2: 0  
-> Tarefa 3 imprimindo contador da tarefa 2: 0  
-> Valor atual da tarefa 2: 1  
-> Tarefa 3 imprimindo contador da tarefa 2: 1  
-> Valor atual da tarefa 2: 2  
-> Tarefa 3 imprimindo contador da tarefa 2: 2
```

Coloque a prioridade da tarefa 3 em 3 e da tarefa 1 em 1:

```
-> Tarefa 3 imprimindo contador da tarefa 2: 0  
-> Valor atual da tarefa 2: 0  
-> Tarefa 3 imprimindo contador da tarefa 2: 1  
-> Valor atual da tarefa 2: 1  
-> Tarefa 3 imprimindo contador da tarefa 2: 2  
-> Valor atual da tarefa 2: 2  
-> Tarefa 3 imprimindo contador da tarefa 2: 3  
-> Valor atual da tarefa 2: 3
```

# Utilizando filas (Queue)

- Exercício 1: Faça as seguintes modificações para testar o código (faça de forma separada com base no programa anterior):
  1. *Aumente o delay da tarefa 2 para 3000ms e verifique o momento em que ocorre o timeout da tarefa 3.*
  2. *Reduza o delay e o tempo de espera para adicionar um novo elemento na tarefa 2 para 250ms e verifique os problemas de sincronia que irão ocorrer.*

# Utilizando filas (Queue)

```
void vTarefa2(void *pvParameters){
    int contador = 0;
    while(1)
    {
        xQueueSend(xFila, &contador, pdMS_TO_TICKS(1000)); //exercicio 1_1
        //xQueueSend(xFila, &contador, pdMS_TO_TICKS(250)); //exercicio 1_2
        Serial.println("Valor atual da tarefa 2: " + String(contador));
        contador++;
        vTaskDelay(pdMS_TO_TICKS(3000)); //exercicio 1_1
        //vTaskDelay(pdMS_TO_TICKS(250)); //exercicio 1_2
    }
}

void vTarefa3(void *pvParameters){
    int valorcontador = 0;
    while(1)
    {
        if(xQueueReceive(xFila, &valorcontador, pdMS_TO_TICKS(1000)) == pdTRUE){
            Serial.println("Tarefa 3 imprimindo contador da tarefa 2: " + String(valorcontador));
        }
        else{
            Serial.println("Tarefa 3 TIMEOUT");
        }
        //vTaskDelay(pdMS_TO_TICKS(1000)); //exercicio 1_2, comentar para o 1_1
    }
}
```

# Utilizando filas (Queue)

- Exercício 1.1 – Como o delay está em 3s, ele recebe 1 elemento e dps ocorre 2 timeout até receber o próximo.

```
09:34:17.902 -> Valor atual da tarefa 2: 0
09:34:19.573 -> Tarefa 3 imprimindo contador da tarefa 2: 0
09:34:20.689 -> Tarefa 3 TIMEOUT
09:34:21.809 -> Tarefa 3 TIMEOUT
09:34:22.928 -> Valor atual da tarefa 2: 1
09:34:22.928 -> Tarefa 3 imprimindo contador da tarefa 2: 1
09:34:24.003 -> Tarefa 3 TIMEOUT
09:34:25.125 -> Tarefa 3 TIMEOUT
```

# Utilizando filas (Queue)

- Exercício 1.2 – Após encher a fila com 5 elementos, ele não consegue enviar todos e alguns elementos começam ser perdidos

```
09:37:39.502 -> Valor atual da tarefa 2: 0
09:37:41.193 -> Tarefa 3 imprimindo contador da tarefa 2: 0
09:37:41.427 -> Valor atual da tarefa 2: 1
09:37:41.709 -> Valor atual da tarefa 2: 2
09:37:41.990 -> Valor atual da tarefa 2: 3
09:37:42.224 -> Valor atual da tarefa 2: 4
09:37:42.272 -> Tarefa 3 imprimindo contador da tarefa 2: 1
09:37:42.505 -> Valor atual da tarefa 2: 5
09:37:42.788 -> Valor atual da tarefa 2: 6
09:37:43.306 -> Valor atual da tarefa 2: 7
09:37:43.399 -> Tarefa 3 imprimindo contador da tarefa 2: 2
09:37:43.589 -> Valor atual da tarefa 2: 8
09:37:44.146 -> Valor atual da tarefa 2: 9
09:37:44.521 -> Valor atual da tarefa 2: 10
09:37:44.521 -> Tarefa 3 imprimindo contador da tarefa 2: 3
09:37:45.037 -> Valor atual da tarefa 2: 11
09:37:45.599 -> Valor atual da tarefa 2: 12
09:37:45.599 -> Tarefa 3 imprimindo contador da tarefa 2: 4
09:37:45.832 -> Valor atual da tarefa 2: 13
09:37:46.399 -> Valor atual da tarefa 2: 14
09:37:46.724 -> Valor atual da tarefa 2: 15
09:37:46.771 -> Tarefa 3 imprimindo contador da tarefa 2: 5
09:37:47.286 -> Valor atual da tarefa 2: 16
09:37:47.802 -> Valor atual da tarefa 2: 17
09:37:47.849 -> Tarefa 3 imprimindo contador da tarefa 2: 6
09:37:48.085 -> Valor atual da tarefa 2: 18
09:37:48.599 -> Valor atual da tarefa 2: 19
09:37:48.975 -> Valor atual da tarefa 2: 20
09:37:48.975 -> Tarefa 3 imprimindo contador da tarefa 2: 8
09:37:49.493 -> Valor atual da tarefa 2: 21
09:37:50.057 -> Valor atual da tarefa 2: 22
09:37:50.057 -> Tarefa 3 imprimindo contador da tarefa 2: 10
09:37:50.288 -> Valor atual da tarefa 2: 23
09:37:50.850 -> Valor atual da tarefa 2: 24
09:37:51.179 -> Valor atual da tarefa 2: 25
09:37:51.224 -> Tarefa 3 imprimindo contador da tarefa 2: 13
09:37:51.742 -> Valor atual da tarefa 2: 26
09:37:52.257 -> Valor atual da tarefa 2: 27
09:37:52.304 -> Tarefa 3 imprimindo contador da tarefa 2: 15
09:37:52.536 -> Valor atual da tarefa 2: 28
09:37:53.052 -> Valor atual da tarefa 2: 29
09:37:53.427 -> Valor atual da tarefa 2: 30
09:37:53.427 -> Tarefa 3 imprimindo contador da tarefa 2: 18
09:37:53.943 -> Valor atual da tarefa 2: 31
09:37:54.505 -> Valor atual da tarefa 2: 32
```

# Utilizando filas (Queue) com interrupções

- Exercício 2: Monitore um botão através de uma interrupção, de forma que ele envie para a tarefa 2, utilizando a estrutura de fila, o valor “1” quando estiver em VCC, e “0” quando estiver em GND. A tarefa 1 é um LED em 1Hz.

```
BaseType_t xQueueSendFromISR  
(  
    QueueHandle_t xQueue,  
    const void *pvItemToQueue,  
    BaseType_t *pxHigherPriorityTaskWoken  
);
```

Setando `pdTRUE` (caso contrário, coloque `NULL`) a tarefa ativada pela interrupção receberá uma prioridade mais alta que a tarefa atual em execução, fazendo com que a mesma seja executada imediatamente. Para garantir a troca de contexto rápida, pode utilizar `portYIELD_FROM_ISR()` ;

## Parameters:

<i>xQueue</i>	The handle to the queue on which the item is to be posted.
<i>pvItemToQueue</i>	A pointer to the item that is to be placed on the queue. The size of the items the queue will hold was defined when the queue was created, so this many bytes will be copied from <i>pvItemToQueue</i> into the queue storage area.
<i>pxHigherPriorityTaskWoken</i>	<code>xQueueSendFromISR()</code> will set <i>*pxHigherPriorityTaskWoken</i> to <code>pdTRUE</code> if sending to the queue caused a task to unblock, and the unblocked task has a priority higher than the currently running task. If <code>xQueueSendFromISR()</code> sets this value to <code>pdTRUE</code> then a context switch should be requested before the interrupt is exited.



# Utilizando filas (Queue) com interrupções

```
#define LED1 7
#define BOTAO 2

/* Variáveis para armazenamento do handle das tasks */
TaskHandle_t xTarefa1Handle = NULL;
TaskHandle_t xTarefa2Handle = NULL;

/* Variáveis para armazenamento do handle das filas */
QueueHandle_t xFila;

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

void funcaoISR() {
    int valor;
    if (digitalRead(BOTAO)) {
        valor=1;
    }
    else{
        valor=0;
    }
    xQueueSendFromISR(xFila, &valor, NULL);
}

void setup() {
    Serial.begin(9600);
    pinMode(BOTAO, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(BOTAO), funcaoISR, CHANGE);
    xFila = xQueueCreate(5, sizeof(int));
    xTaskCreate(vTarefa1, "Tarefa1",128, NULL, 2, &xTarefa1Handle);
    xTaskCreate(vTarefa2, "Tarefa2",128, NULL, 1, &xTarefa2Handle);
}
```

Vai imprimir várias vezes,  
está sem debounce

```
void loop() {
    //As funções são executadas nas tarefas
}
```

```
void vTarefa1(void *pvParameters){
    pinMode(LED1, OUTPUT);
    while(1)
    {
        digitalWrite(LED1, !digitalRead(LED1));
        vTaskDelay(pdMS_TO_TICKS(500));
    }
}
```

Vai aguardar indefinidamente

```
void vTarefa2(void *pvParameters){
    int valorrecebidofila;
    while(1)
    {
        xQueueReceive(xFila, &valorrecebidofila, portMAX_DELAY);
        Serial.println("Interrupcao: " + String(valorrecebidofila));
    }
}
```

Setando pdTRUE a tarefa ativada pela  
interrupção receberá uma prioridade mais alta  
que a tarefa atual em execução, fazendo com  
que a mesma seja executada imediatamente.

# Utilizando filas (Queue) com interrupções

- Exercício 3: Com base no exercício 1.2, faça um código em que, quando o botão (interrupção) for pressionado, ele resete a fila.

```
attachInterrupt(digitalPinToInterrupt(BOTAO), funcaoISR, FALLING);

void funcaoISR(){
    xQueueReset(xFila);
    Serial.println("Fila resetada");
}
```

```
-> Tarefa 3 imprimindo contador da tarefa 2: 10
-> Valor atual da tarefa 2: 23
-> Valor atual da tarefa 2: 24
-> Valor atual da tarefa 2: 25
-> Tarefa 3 imprimindo contador da tarefa 2: 13
-> Valor atual da tarefa 2: 26
-> Valor atual da tarefa 2: 27
-> Fila resetada
-> Valor atual da tarefa 2: 28
-> Tarefa 3 imprimindo contador da tarefa 2: 27
-> Valor atual da tarefa 2: 29
-> Valor atual da tarefa 2: 30
-> Valor atual da tarefa 2: 31
-> Valor atual da tarefa 2: 32
```

# Utilizando filas (Queue) com interrupções

- Exercício 4: Com base no exercício 1.2, faça um código em que, quando o botão for pressionado, o elemento atual seja enviado para o início da fila. Deixe a tarefa 2 com prioridade mais alta que as outras e altere o delay da tarefa e o tempo de espera da fila para 500 ms. Imprima qual valor foi enviado para o início da fila.

# Utilizando filas (Queue) com interrupções

```
#include "Arduino_FreeRTOS.h"
#include "task.h"
#include "queue.h"

#define LED1 7
#define BOTAO 2

int x=0;

/* Variáveis para armazenamento do handle das tasks */
TaskHandle_t xTarefa1Handle = NULL;
TaskHandle_t xTarefa2Handle = NULL;
TaskHandle_t xTarefa3Handle = NULL;

/* Variáveis para armazenamento do handle das filas */
QueueHandle_t xFila;

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

void funcaoISR() {
    x = 1;
}

void setup() {
    Serial.begin(9600);
    xFila = xQueueCreate(5, sizeof(int));
    xTaskCreate(vTarefa1, "Tarefa1",128, NULL, 2, &xTarefa1Handle);
    xTaskCreate(vTarefa2, "Tarefa2",256, NULL, 3, &xTarefa2Handle);
    xTaskCreate(vTarefa3, "Tarefa3",128, NULL, 1, &xTarefa3Handle);
    pinMode(BOTAO, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(BOTAO), funcaoISR, FALLING);
}

void loop() {
    //As funções são executadas nas tarefas
}

void vTarefa1(void *pvParameters){
    pinMode(LED1, OUTPUT);
    while(1)
    {
        digitalWrite(LED1,!digitalRead(LED1));
        vTaskDelay(pdMS_TO_TICKS(500));
    }
}
```

# Utilizando filas (Queue) com interrupções

```
void vTarefa2(void *pvParameters){
    int contador = 0;
    while(1)
    {
        if (x == 0){
            xQueueSend(xFila, &contador, pdMS_TO_TICKS(500));
        }
        if (x == 1)
        {
            Serial.println("Valor: " + String(contador) + " enviado para o início da fila");
            xQueueSendToFront(xFila, &contador, pdMS_TO_TICKS(500));
            x=0;
        }
        Serial.println("Valor atual da tarefa 2: " + String(contador));
        contador++;
        vTaskDelay(pdMS_TO_TICKS(500));
    }
}

void vTarefa3(void *pvParameters){
    int valorcontador = 0;
    while(1)
    {
        if(xQueueReceive(xFila,&valorcontador, pdMS_TO_TICKS(1000))==pdTRUE){
            Serial.println("Tarefa 3 imprimindo contador da tarefa 2: " + String(valorcontador));
        }
        else{
            Serial.println("Tarefa 3 TIMEOUT");
        }
        vTaskDelay(pdMS_TO_TICKS(1000));
    }
}
```



# Utilizando filas (Queue) com interrupções

- Exercício 5: Com base no exercício 1.2, aumente o delay da tarefa 2 para 500ms, aumente o tamanho da fila para 10 e imprima o número de espaços livres e de espaços ocupados na fila. Consulte o manual:
- <https://www.freertos.org/a00018.html>

```
void vTarefa2(void *pvParameters){
    int contador = 0;
    while(1)
    {
        xQueueSend(xFila, &contador, pdMS_TO_TICKS(500));
        Serial.println("Valor atual da tarefa 2: " + String(contador));
        Serial.println("Espaços livre na fila: " + String(uxQueueSpacesAvailable(xFila)));
        Serial.println("Espaços ocupados na fila: " + String(uxQueueMessagesWaiting(xFila)));
        contador++;
        vTaskDelay(pdMS_TO_TICKS(500));
    }
}
```

16:29:16.343 -> Valor atual da tarefa 2: 0  
16:29:18.098 -> Espaços livre na fila: 9  
16:29:18.146 -> Espaços ocupados na fila: 1  
16:29:18.192 -> Tarefa 3 imprimindo contador da tarefa 2: 0  
16:29:18.708 -> Valor atual da tarefa 2: 1  
16:29:18.755 -> Espaços livre na fila: 9  
16:29:18.755 -> Espaços ocupados na fila: 1  
16:29:19.316 -> Valor atual da tarefa 2: 2  
16:29:19.363 -> Espaços livre na fila: 8  
16:29:19.363 -> Espaços ocupados na fila: 2  
16:29:19.410 -> Tarefa 3 imprimindo contador da tarefa 2: 1  
16:29:19.930 -> Valor atual da tarefa 2: 3  
16:29:19.977 -> Espaços livre na fila: 8  
16:29:20.025 -> Espaços ocupados na fila: 2

- <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/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.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/>