EGM0017 (60H)

# FLUXO E METODOLOGIAS DE PROJETO DE SISTEMAS EMBARCADOS

PROF. JOSENALDE OLIVEIRA

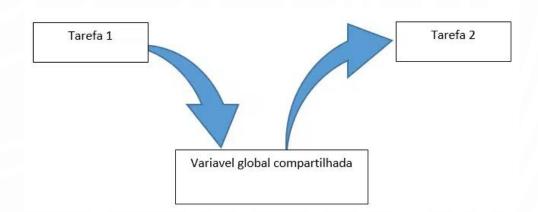
josenalde.oliveira@ufrn.br

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

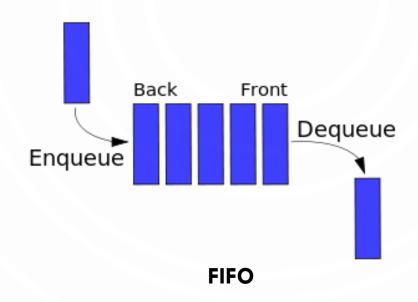
#### FreeRTOS - trabalhando com FILAS (queue)

- Sincronização e comunicação entre tarefas e ISRs (em geral, envio de variáveis entre tasks)
- Queue é um BUFFER, uma fila de dados no formato **FIFO** (first in first out)
- Reduz necessidade de variáveis globais.

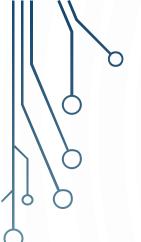
Mecanismos de comunicação entre tarefas denominados de *thread safe*. As tarefas envolvidas em comunicação *thread safe* poderão ser interrompidas ou executadas a qualquer momento sem que isso gere um estado inconsistente nos dados que são transferidos entre estas tarefas. (semáforos, mutexes, filas)



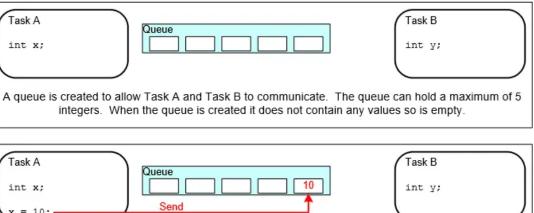
## FreeRTOS – trabalhando com FILAS (queue)

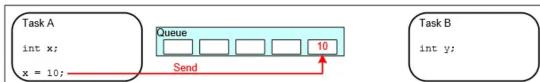


https://embarcados.com.br/rtos-queue-sincronizacao-e-comunicacao/

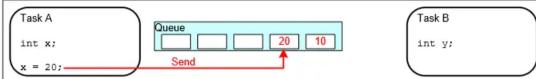


# FreeRTOS - trabalhando com FILAS (queue)

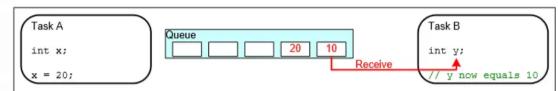




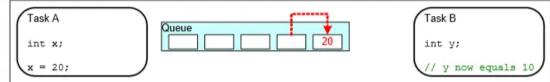
Task A writes (sends) the value of a local variable to the back of the queue. As the queue was previously lempty the value written is now the only item in the queue, and is therefore both the value at the back of the queue and the value at the front of the queue.



Task A changes the value of its local variable before writing it to the queue again. The queue now contains copies of both values written to the queue. The first value written remains at the front of the queue, the new value is inserted at the end of the queue. The queue has three empty spaces remaining.



Task B reads (receives) from the queue into a different variable. The value received by Task B is the value from the head of the queue, which is the first value Task A wrote to the queue (10 in this illustration).



Task B has removed one item, leaving only the second value written by Task A remaining in the gueue. This is the value Task B would receive next if it read from the queue again. The queue now has four empty spaces remaining

PROF. JOSENALDE OLIVEIRA – FLUXO E METODOLOGIAS DE PROJETO DE SISTEMAS EMBARCADOS

An example sequence of writes to, and reads from a queue

# FreeRTOS – trabalhando com FILAS (<u>queue</u>)

Nome da função	Finalidade	Observação
xQueueCreate	Cria uma fila nova	Antes de realizar qualquer operação com filas, esta função deve ser executada para criar a fila.
vQueueDelete	Apaga uma fila e libera toda a memória alocada para a fila	Em sistemas com pouca memória sempre é importante liberar memória não utilizada
xQueueSend	Enfileira um elemento no final da fila.	Esta função NÃO deve ser usada dentro de uma rotina de tratamento de interrupção.
xQueueSendFromISR	Enfileira um elemento no final da fila.	Esta função pode ser usada dentro de uma rotina de tratamento de interrupção.
xQueueReceive	Remove um elemento do início da uma fila.	Esta função NÃO deve ser usada dentro de uma rotina de tratamento de interrupção.
xQueueReceiveFromISR	Remove um elemento do início de uma fila.	Esta função pode ser usada dentro de uma rotina de tratamento de interrupção.

#### <sup>©</sup>FreeRTOS — trabalhando com FILAS (<u>queue</u>)

- Filas de tamanho fixo e tamanho definido por slot (bytes). Em geral usa-se sizeof(...)
- Exemplo, passagem de variáveis entre tasks

QueueHandle\_t xQueueCreate( UBaseType\_t uxQueueLength, UBaseType\_t uxItemSize ); Se a fila for criada, a função retornará o identificador da fila. Se a fila não for criada, retornará 0.

BaseType\_t xQueueSend(QueueHandle\_t xQueue, const void \* pvItemToQueue, TickType\_t xTicksToWait); // tempo de bloqueio até fila estar com slot disponível - se setado em 0, retorno imediato se a fila está cheia

Equivalente à xQueueSendToBack(). Não pode ser usado dentro de ISR

Se enviou retorna pdTRUE, senão errQUEUE FULL

BaseType\_t xQueueReceive( QueueHandle\_t xQueue, void \*pvBuffer, TickType\_t
xTicksToWait);

Se enviou retorna pdTRUE, senão pdFALSE. Recebe e remove da fila. Se quiser manter usar xQueuePeek

### <sup>©</sup>FreeRTOS — trabalhando com FILAS (<u>queue</u>)

void loop() { Exemplo básico – framework Arduino QueueHandle t queue; for(int i = 0; i<10; i++){ xQueueSend(queue, &i, portMAX\_DELAY); void setup() { Serial.begin(115200); int element; vTaskDelay(pd\_MS\_TO\_TICKS(1000)); for(int i = 0; i < 10; i++){ queue = xQueueCreate(10, sizeof(int)); xQueueReceive(queue, &element, portMAX\_DELAY); Serial.print(element); if(queue == NULL){ Serial.print(" "); Serial.println("Error creating the queue"); vTaskDelete(NULL); //não executará loop() Serial.println(); vTaskDelay(pd\_MS\_TO\_TICKS(1000));



#### FreeRTOS - Exemplos(queue)

FreeRTOS – produtor consumidor com filas

https://github.com/josenalde/flux-embedded-design/blob/main/src/producer consumer.ino

FreeRTOS – filas para comunicação task - ISR

https://blog.eletrogate.com/freertos-filas-trocando-informacao-entre-tarefas/

Interrupção e filas – esp-idf

https://github.com/josenalde/intr-queue-espidf

Cálculo de média móvel

https://github.com/josenalde/flux-embedded-design/blob/main/src/average temperature queue.c

Exemplo de aplicação multicore/task em automação

https://blog.eletrogate.com/rtos-com-esp32-como-programar-multitarefas/

Exemplo de aplicação multicore/task loT com MQTT <a href="https://www.makerhero.com/blog/faca-seu-rastreador-veicular-com-esp32-gps-e-freertos/">https://www.makerhero.com/blog/faca-seu-rastreador-veicular-com-esp32-gps-e-freertos/</a>