TÓPICOS ESPECIAIS – SISTEMAS EMBARCADOS
PROGRAMAÇÃO PARALELA E TEMPO REAL

PROF. JOSENALDE OLIVEIRA

josenalde.oliveira@ufrn.br

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

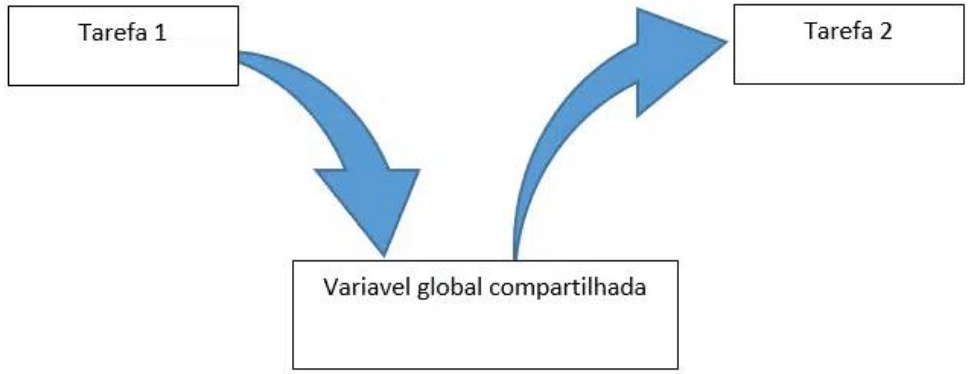
FreeRTOS – trabalhando com FILAS (queue)

- 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. Por sinal, em sistemas embarcados, ao usar variáveis globais é um padrão de projeto usar o qualificador **volatile** em C/C++. Por exemplo:

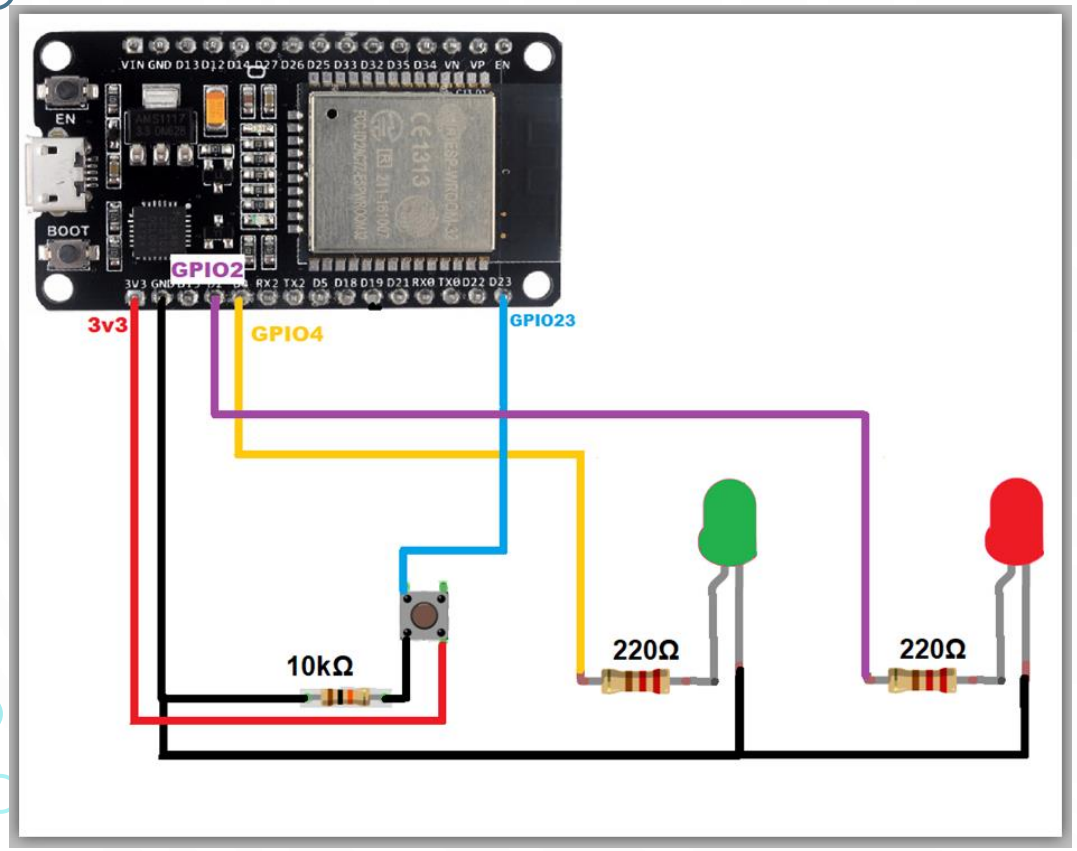
volatile int foo;
int volatile foo;

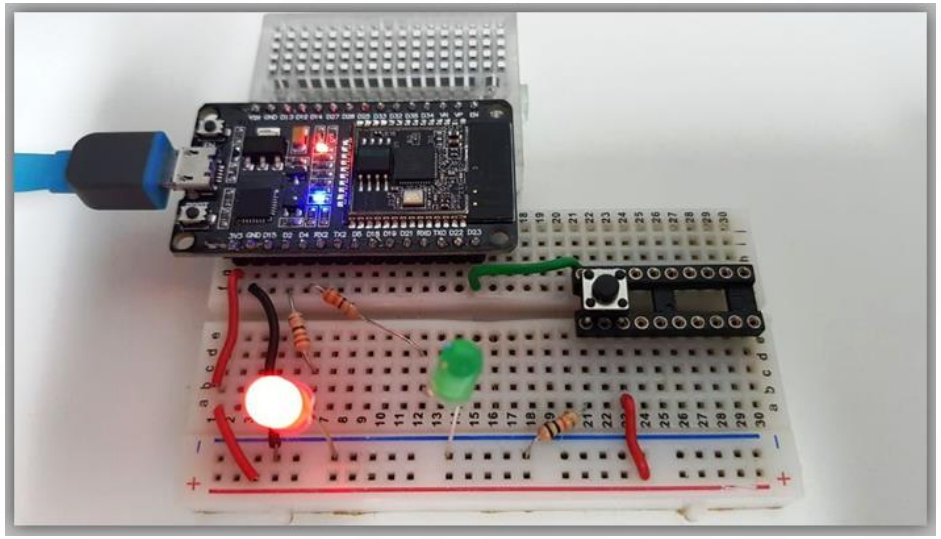
- Indica que o valor pode ser alterado a qualquer momento/inesperadamente sem ser influenciado por processos de otimização do compilador. Devem ser usadas em situações de variáveis globais alteradas por ISR e em aplicações multithread

Mecanismos de comunicação entre tarefas que não são suscetíveis a este tipo de problema são 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 – Exemplo de volatile





https://www.fernandok.com/2018/06/os-profissionais-sabem-disso-interrupt.html

FreeRTOS – Exemplo de volatile

```
#define pinBUTTON    23 //pino de interrupção (botão)
#define DEBOUNCETIME 10 //tempo máximo de debounce para o botão (ms)

#define pinREDled    2 //pino do led VERMELHO
#define pinGREENled  4 //pino do led VERDE

//É DECLARADA VOLÁTIL PORQUE SERÁ COMPARTILHADA PELO ISR E PELO CÓDIGO PRINCIPAL
volatile int numberOfButtonInterrupts = 0; //número de vezes que a interrupção foi executada
volatile bool lastState; //guarda o último estado do botão quando ocorreu a interrupção
volatile uint32_t debounceTimeout = 0; //guarda o tempo de debounce

//variáveis para controle dentro do loop
uint32_t saveDebounceTimeout;
bool saveLastState;
int save;

// For setting up critical sections (enableinterrupts and disableinterrupts not available)
// used to disable and interrupt interrupts
// Para configurar seções críticas (interrupções de ativação e interrupções de desativação não c
// usado para desabilitar e interromper interrupções
portMUX_TYPE mux = portMUX_INITIALIZER_UNLOCKED;
```

https://www.fernandok.com/2018/06/os-profissionais-sabem-disso-interrupt.html

FreeRTOS – Exemplo de volatile

```
void setup()
{
  Serial.begin(115200);
  String taskMessage = "Debounced ButtonRead Task running on core ";
  taskMessage = taskMessage + xPortGetCoreID();
  Serial.println(taskMessage); //mostra o core que o botão está executando

  // set up button Pin
  pinMode (pinREDled, OUTPUT);
  pinMode (pinGREENled, OUTPUT);

  pinMode(pinBUTTON, INPUT_PULLUP);  // Pull up to 3.3V on input - some buttons already have thi

  attachInterrupt(digitalPinToInterrupt(pinBUTTON), handleButtonInterrupt, CHANGE);   //configu

  digitalWrite(pinREDled, HIGH); //inicializa o LED VERMELHO como aceso
}
```

https://www.fernandok.com/2018/06/os-profissionais-sabem-disso-interrupt.html

```
void loop()
{
    portENTER_CRITICAL_ISR(&mux); // início da seção crítica
      save  = numberOfButtonInterrupts;
      saveDebounceTimeout = debounceTimeout;
      saveLastState  = lastState;
    portEXIT_CRITICAL_ISR(&mux); // fim da seção crítica

    bool currentState = digitalRead(pinBUTTON); //recupera o estado atual do botão

    //se o estado do botão mudou, atualiza o tempo de debounce
    if(currentState != saveLastState)
    {
      saveDebounceTimeout = millis();
    }

    //se o tempo passado foi maior que o configurado para o debounce e o número de interrupções ocorridas é maior que Z
    if( (millis() - saveDebounceTimeout) > DEBOUNCETIME && (save != 0) )
    {
           //se o botão está pressionado
           //liga o led verde e apaga o vermelho
           //caso contrário
           //liga o led vermelho e apaga o verde
           if(currentState) {
              digitalWrite(pinGREENled, HIGH);
              digitalWrite(pinREDled, LOW);
            }
            else{
              digitalWrite(pinGREENled, LOW);
              digitalWrite(pinREDled, HIGH);
            }
            Serial.printf("Button Interrupt Triggered %d times, current State=%u, time since last trigger %dms\n", save
            portENTER_CRITICAL_ISR(&mux);  //início da seção crítica
              numberOfButtonInterrupts = 0; // reconhece que o botão foi pressionado e reseta o contador de interrupçã
            portEXIT_CRITICAL_ISR(&mux); //fim da seção crítica
    }
}
```

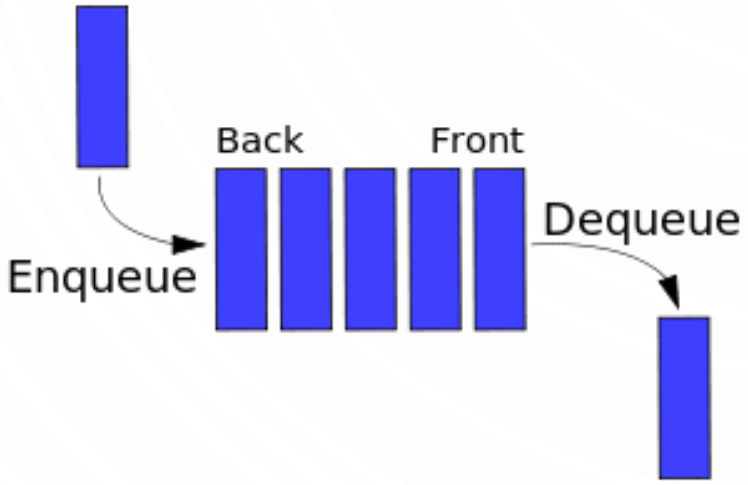
FreeRTOS – Exemplo de volatile

```
// Interrupt Service Routine - Keep it short!
//Interrupção
//Interrompe a rotina de serviço
//IRAM_ATTR --> é utilizado para indicar que esse trecho de código ficará na seção do barramento de instruções da RAM
//portENTER_CRITICAL_ISR /´portEXIT_CRITICAL_ISR --> Isso é necessário porque a variável que vamos usar também é alter
                                                    //como visto anteriormente, e precisamos evitar problemas de acess
void IRAM_ATTR handleButtonInterrupt() {
    portENTER_CRITICAL_ISR(&mux);
      numberOfButtonInterrupts++;
      lastState = digitalRead(pinBUTTON);
      debounceTimeout = xTaskGetTickCount(); //versão do millis () que funciona a partir da interrupção //version of m
    portEXIT_CRITICAL_ISR(&mux);
}
```

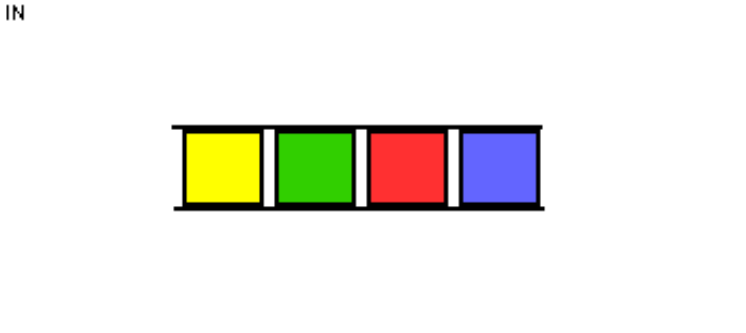
https://www.fernandok.com/2018/06/os-profissionais-sabem-disso-interrupt.html

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)

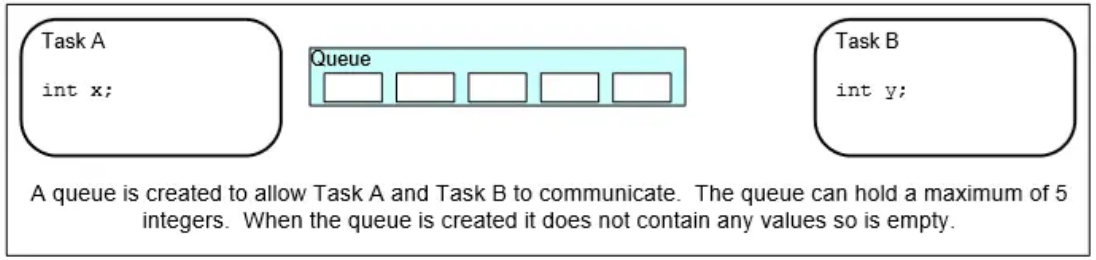


FIFO

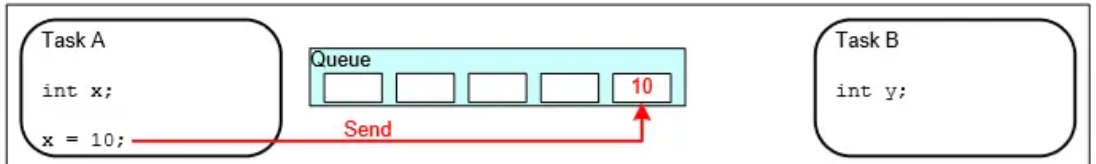


https://embarcados.com.br/rtos-queue-sincronizacao-e-comunicacao/

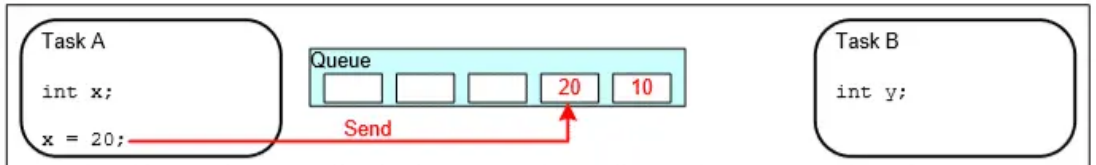
FreeRTOS – trabalhando com FILAS (queue)



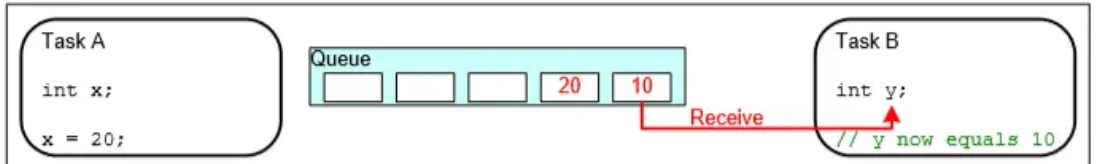
A queue is created to allow Task A and Task B to communicate. The queue can hold a maximum of 5 integers. When the queue is created it does not contain any values so is empty.



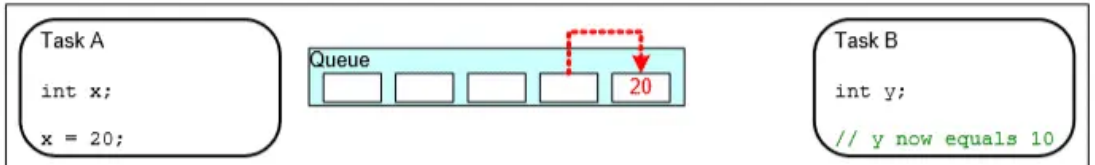
Task A writes (sends) the value of a local variable to the back of the queue. As the queue was previously empty the value written is now the only item in the queue, and is therefore both the value at the back of the queue and the value at the front of the queue.



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 queue. This is the value Task B would receive next if it read from the queue again. The queue now has four empty spaces remaining.

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

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

FreeRTOS – trabalhando com FILAS (queue)

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.

FreeRTOS – trabalhando com FILAS (queue)

- Filas de tamanho fixo e tamanho definido por slot (bytes)
- Exemplo, passagem de variáveis entre tasks

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

```
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
```

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

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.

https://github.com/josenalde/parallel_programming_rtos/blob/main/src/queue1/queue1.ino

FreeRTOS – trabalhando com FILAS (queue)

- Exemplo básico

```
QueueHandle_t queue;

void setup() {

  Serial.begin(115200);

  queue = xQueueCreate( 10, sizeof( int ) );

  if(queue == NULL){
    Serial.println("Error creating the queue");
  }

}
```

```
void loop() {

  if(queue == NULL) return;

  for(int i = 0; i<10; i++){
    xQueueSend(queue, &i, portMAX_DELAY);
  }

  int element;

  for(int i = 0; i < 10; i++){
    xQueueReceive(queue, &element, portMAX_DELAY);
    Serial.print(element);
    Serial.print("|");
  }

  Serial.println();
  delay(1000);
}
```

https://github.com/josenalde/parallel_programming_rtos/blob/main/src/queue2/queue2.ino

FreeRTOS – produtor consumidor com filas

https://github.com/josenalde/parallel_programming_rtos/blob/main/src/producer_consumer/producer_consumer.ino

FreeRTOS – filas para comunicação task - ISR

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

Exemplo de aplicação multicore/task em automação

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

Exemplo de aplicação multicore/task IoT com MQTT

https://www.makerhero.com/blog/faca-seu-rastreador-veicular-com-esp32-gps-e-freertos/