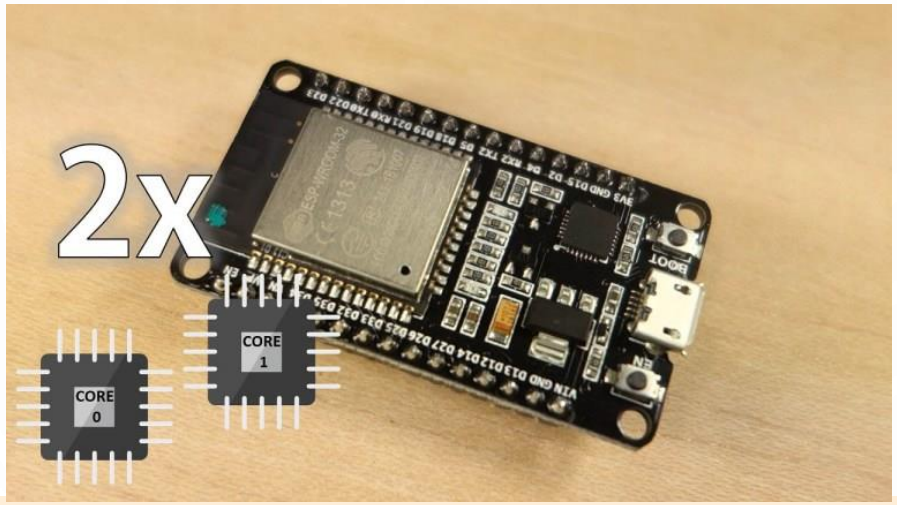
SISTEMAS EMBARCADOS

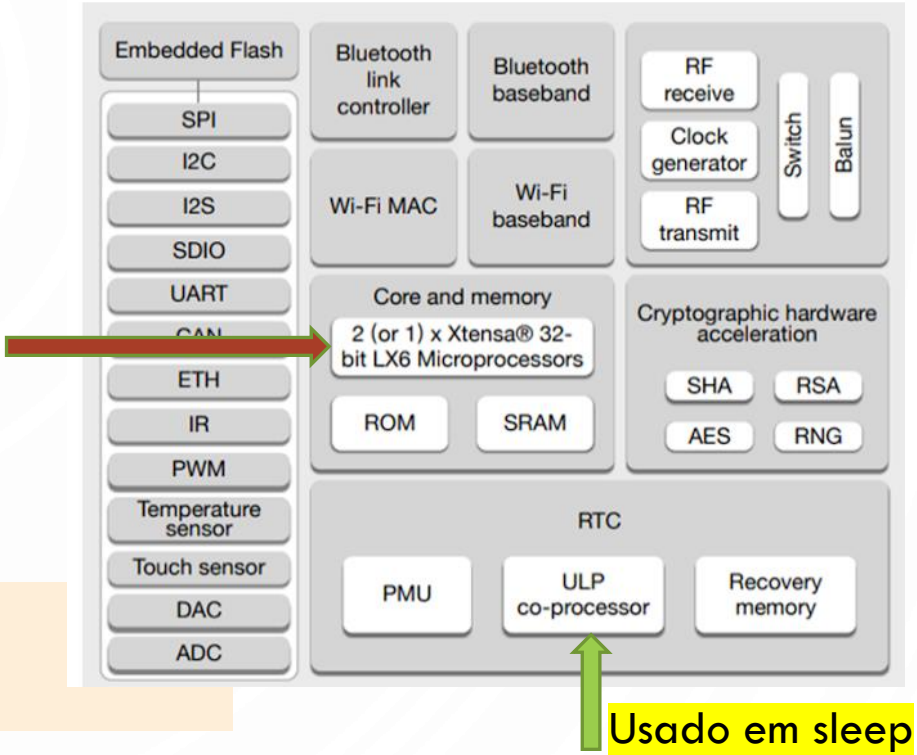
PROF. JOSENALDE OLIVEIRA

josenalde.oliveira@ufrn.br

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

Multicore e RTOS (real time OS) no ESP32





- Core 1 (APP_CPU) – default
- Core 0 (PROTOCOL_CPU) - livre

```
// setup() and loop() runs at APP_CPU by default
void setup() {
  Serial.begin(115200);
  Serial.println(xPortGetCoreID()); // ID 1
}

void loop() {
    //Default loop runs on APP_CPU
    Serial.print("This loop runs on APP_CPU which id is:");
    Serial.println(xPortGetCoreID()); // ID 1 (APP_CPU)
}
```

FreeRTOS (port para ESP32-nativo (Xtensa)): https://freertos.org/
https://freertos.org/RTOS_ports.html
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html.

Outras opções: ZEPHYR, AWS FreeRTOS, Mongoose OS etc.
https://docs.zephyrproject.org/latest/boards/xtensa/esp32/doc/index.htm

Multicore e RTOS (real time OS) no ESP32

https://www.embarcados.com.br/rtos-sistema-operacional-de-tempo-real/

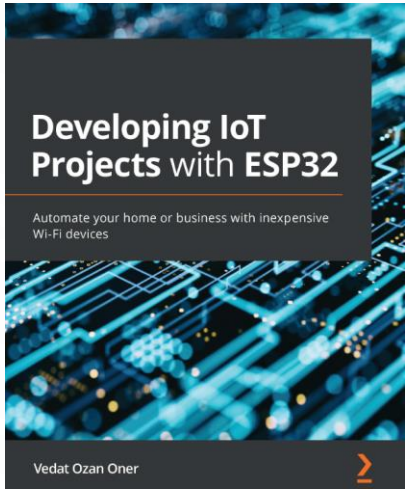
Quais são as diferenças entre GPOS e RTOS?

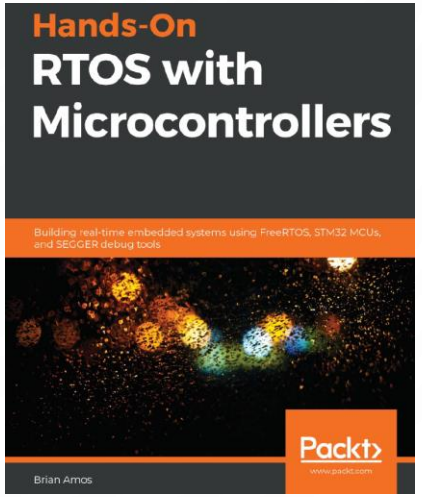
GPOS: É amplamente utilizado em computadores e similares por ter um alto throughput (vazão) de dados, é focado na execução de muitas tarefas simultaneamente, onde atraso de uma não é crítico, irá apenas atrasá-la. As tarefas do GPOS não tem um tempo limite para serem executadas, então é comumente chamado de “Not Time Critical”.

RTOS: É um sistema operacional mais especializado onde o tempo de resposta é mais importante do que executar centenas de tarefas simultaneamente. O tempo de resposta não precisa necessariamente ser o mais rápido possível, como o nome pode sugerir, mas deve ser previsível, logo, “Real-Time” pode ser uma resposta de vários minutos ou nanos segundos, dependendo da forma que seu sistema funciona. As tarefas do RTOS contam com tempo limite para serem executadas, então é comumente chamado de “Time Critical”.

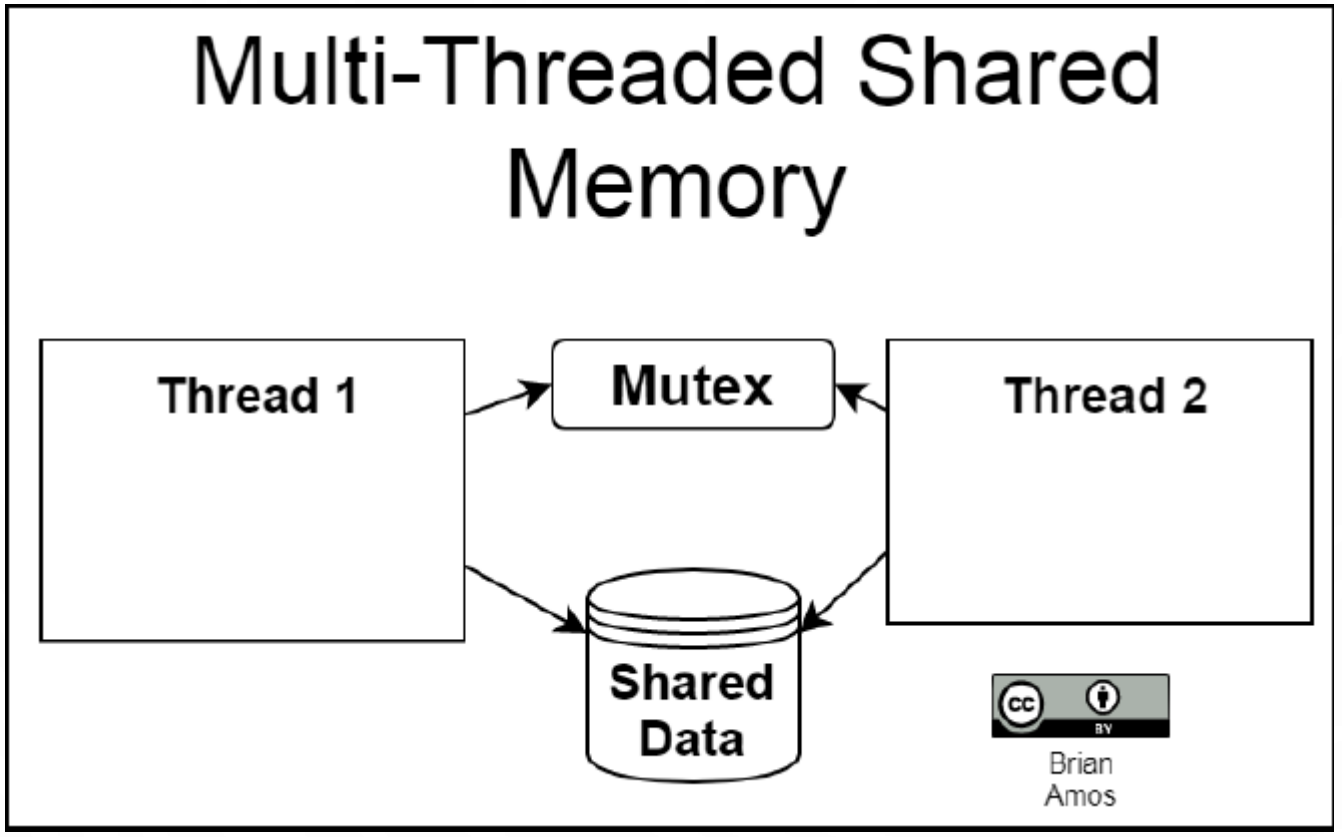
Resgata conceitos de SO: scheduler (agendador) e tasks (tarefas)

No RTOS – scheduler determinístico – garantir tarefa executada no tempo especificado





Multicore e RTOS (real time OS) no ESP32



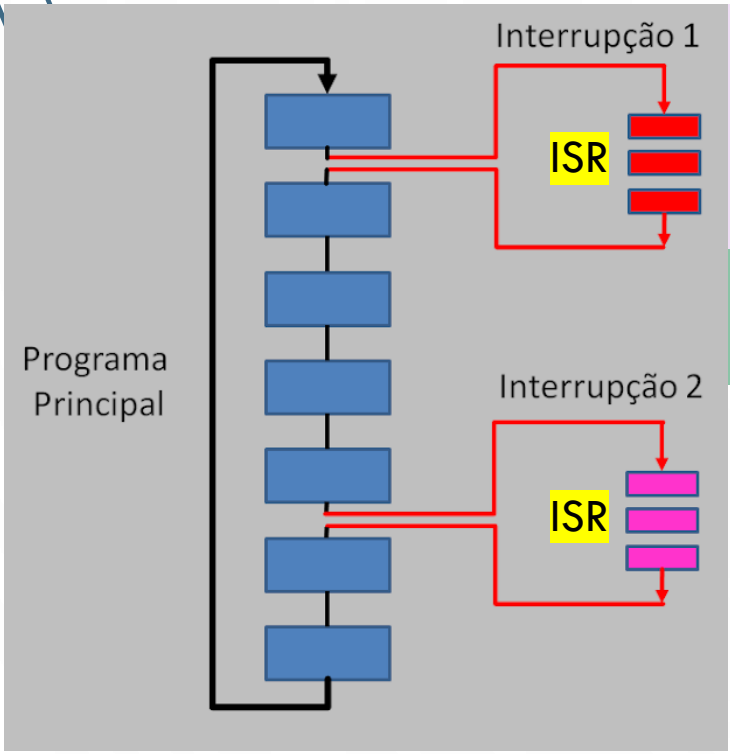

```
/* Task 1 */
mutexWait(mutex_mens_room);
// Safely use shared resource
mutexRelease(mutex_mens_room);
/* Task 2 */
mutexWait(mutex_mens_room);
// Safely use shared resource
mutexRelease(mutex_mens_room);
```

```
/* Task 1 - Producer */
semPost(sem_power_btn);
// Send the signal
/* Task 2 - Consumer */
semPend(sem_power_btn);
// Wait for signal
```

Semárofo
Não bloqueante
Usado dentro de ISR
Para sinalização

MUTEX: bloqueio de recurso compartilhado, liberando o recurso após o uso

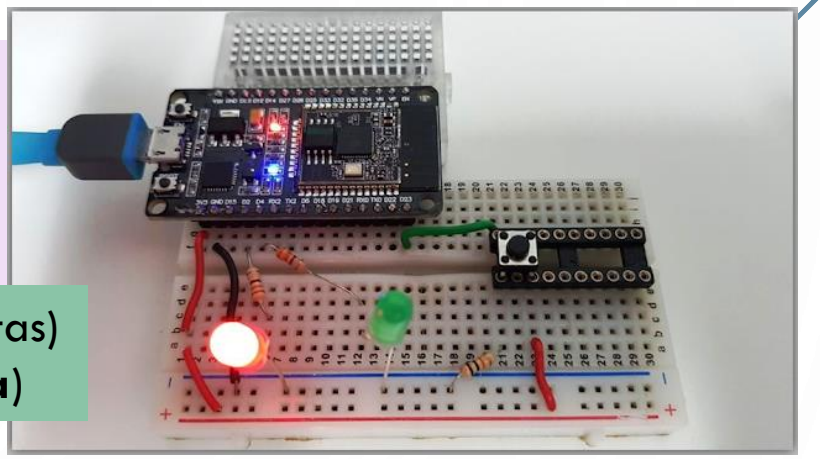
Interrupções – INTERRUPT SERVICE ROUTINE (ISR)



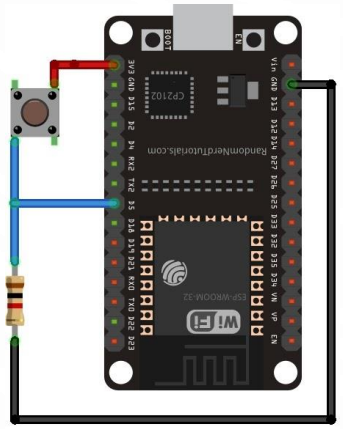
No caso do ESP32 deve ser sinalizado que a ISR executa na RAM e não na FLASH ROM

```
void IRAM_ATTR <func_name>
```

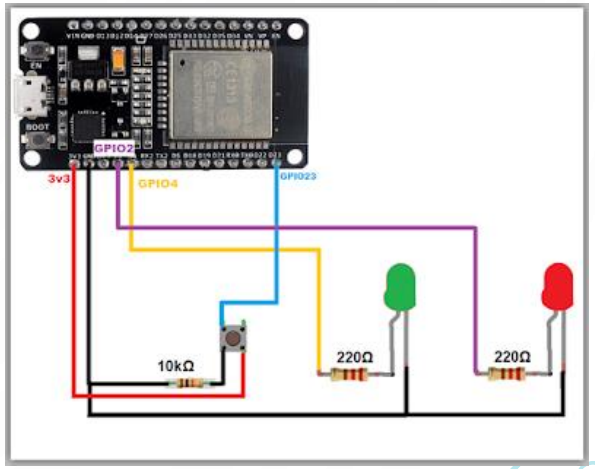
Leituras de GPIO: polling (varredura nas portas) **por interrupção (demanda)**



MODOS DE DISPARO DE INTERRUPÇÃO



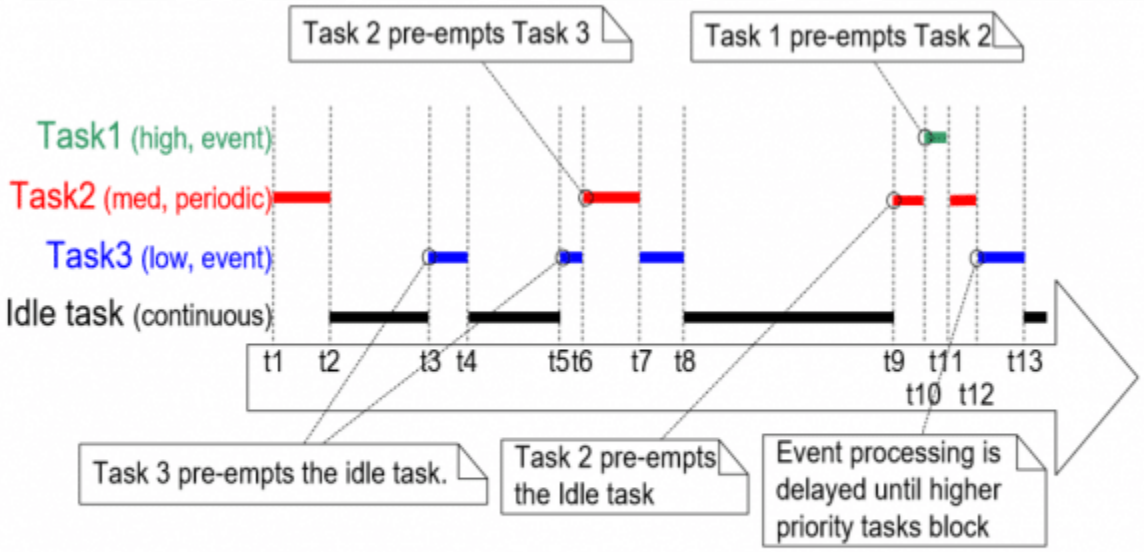
- **FALLING:** um GPIO vai do nível alto (3V3) para nível baixo (0V).
- **RISING:** um GPIO vai do nível baixo (0V) para nível alto (3V3).
- **LOW:** gera uma interrupção quando o GPIO está em nível baixo;
- **HIGH:** gera interrupção quando o GPIO está em nível alto;
- **CHANGE:** quando há qualquer transição de nível no GPIO. Ou seja, tanto de nível baixo para alto quanto de nível alto para baixo.



https://github.com/josenalde/embeddedsystems/blob/master/src/mt8.ino

Escalonadores

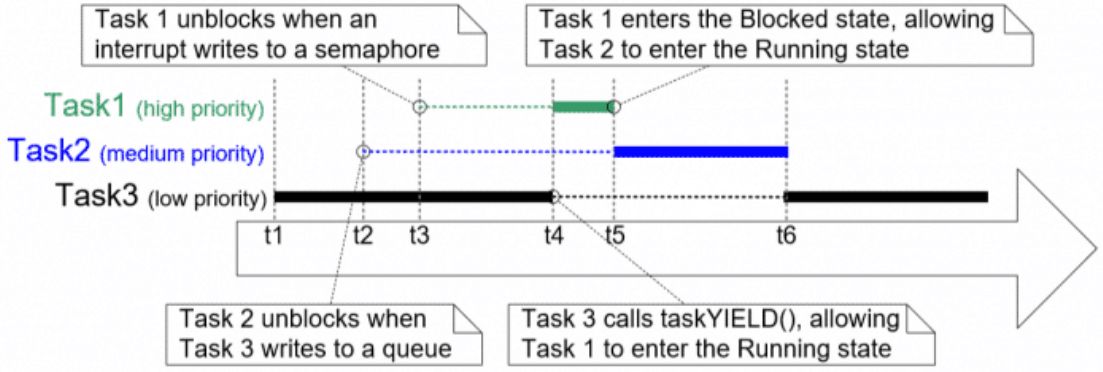
PREEMPTIVO - Padrão do FreeRTOS ESP32



Execution pattern highlighting task prioritization and pre-emption in a hypothetical application in which each task has been assigned a unique priority

Tarefas de maior prioridade podem interromper de menor prioridade.
Idle task é sempre 0

COOPERATIVO



Execution pattern demonstrating the behavior of the co-operative scheduler

Tarefa executa até o fim

Time slicing ESP32 (CPU divide tempo entre tarefas)
à 240 MHz, <= 10us, porém padrão é 1000 Hz (1 ms)
Para a escolha das tarefas métodos RR (padrão), SJF, SRT etc.: Se mesma prioridade RR tenta ser justo!

Estado das tarefas

COOPERATIVO

BLOQUEADA (blocked): aguardando sua vez; retorna por TEMPO (timeout) ou SINCRONISMO (semáforo, filas etc.)
Um delay em RTOS não trava a CPU, apenas deixa bloqueada a tarefa T segundos, as outras rodam normal... `(vTaskDelay)`

SUSPENSA (suspended): vTaskSuspend() e vTaskResume() – pouco usado (desligar tarefa por tempo)

PRONTA (ready): após um delay, por exemplo. Não entra em execução imediata mas espera ser escolhida pelo scheduler

EXECUÇÃO (RUNNING): alocada no momento na CPU. No caso do ESP32, pode ter 2 simultâneas; as outras estão em algum outro estado (blocked, suspended ou ready)

Vamos estudar alguns códigos:

https://github.com/josenalde/embeddedsystems/blob/master/src/mt3.ino

https://github.com/josenalde/embeddedsystems/blob/master/src/mt4.ino