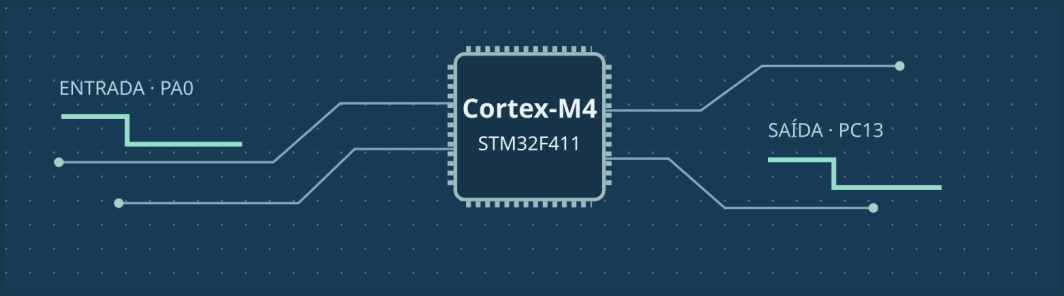
Manual de uso da plataforma

Investigue a resposta do firmware com o código, os pinos e o escalonador. Cinco estudos de caso reconstroem o que aconteceu na BlackPill simulada.



Investigações de threads

01	Travessia de pedestres →	Por que Traffic não aparece?	09
02	Sirene e prazo →	Qual tarefa atrasou o alarme?	11
03	Fila de pulsos →	A fila copiou os três itens?	14
04	Deadlock →	Quem espera por qual mutex?	17
05	Starvation →	Por que a tarefa pronta não executa?	19

Bancada e instrumentos

Primeiro LED →
Editor e depuração →
Circuito e display →
Analisador lógico →
Controles do Thread Viewer →
Histórico e exportação →
Arquitetura e NVIC →
Relatório de ensaio →
Validação na placa →

Abrir o Cortex Lab →

Universidade Tecnológica Federal do Paraná · Curitiba

01 / BANCADA • PRIMEIROS PASSOS

Acender o primeiro LED

Comece com um sinal cujo comportamento você possa prever. Na BlackPill, tanto o botão KEY em PA0 quanto o LED em PC13 são ativos em nível baixo.

Uma execução de referência

01 Abra a base da aula 6. Em Laboratórios da UTFPR, abra o exemplo resolvido da aula sobre nível e evento. Selecione Abrir no editor. O projeto informa STM32F411CE e CMSIS.

02 Compile e leia o retorno. Pressione F7. Aguarde a confirmação de compilação e verifique se o nome da placa e a entrada main.c correspondem ao roteiro. Se houver erro, abra Build / comandos e examine o primeiro diagnóstico com arquivo e linha.

03 Aplique um estímulo. Pressione F5 e abra Ver circuito. Pressione o botão de PA0, mantenha-o pressionado e solte. Volte à análise de sinais para conferir a mudança de PC13.

Ação	PA0	PC13	LED físico
Botão solto	1	1	Apagado
Botão pressionado	0	0	Aceso
Botão solto outra vez	1	1	Apagado

> Critério de sucesso
> A saída acompanha a entrada nas duas transições. Se o LED continuar aceso, pause a CPU, confira a última leitura do botão e verifique se o projeto em execução é o que você compilou.

Abrir o exemplo da aula 6 →

02 / EDITOR • DEPURAÇÃO

Encontre a linha que decide

Um projeto guarda a placa, os arquivos, a configuração e os módulos associados. Abra sua própria cópia antes de alterar um exemplo para comparar o original e o ensaio.

Do arquivo à instrução

01 Localize a entrada. Em Meus projetos, abra o projeto. Na árvore, selecione main.c ou o arquivo de entrada indicado. A busca do editor localiza ocorrências e Ctrl + clique sobre um símbolo, ou F12, abre a definição disponível em um arquivo do projeto.

02 Formule uma previsão. No exemplo do botão, encontre button_pressed e a escrita em GPIOC->BSRR. Preveja o nível de PC13 em cada ramo. Em Fluxograma, confira quais decisões levam à saída.

03 Pare em um ponto observável. Compile e marque um breakpoint em uma linha mapeada no binário. Em Código de máquina, localize o endereço correspondente. Um trecho em C pode gerar várias instruções.

Atalho	Ação na bancada	Uso no ensaio
F7	Compilar	Gerar o binário após uma edição
F5	Executar / pausar	Parar a CPU para examinar estado
F11	Passo	Avançar uma instrução
F10	Passar	Acompanhar o efeito de uma chamada
Ctrl + F11	Retornar	Sair da função atual
Ctrl + F10	Até cursor	Avançar até a linha selecionada
Shift + F5	Reiniciar	Recomeçar CPU e captura

> Guarde a versão medida
> Ao testar outro valor, anote o arquivo e a constante modificada. Exporte a cópia do projeto usada na captura para que outra pessoa possa repetir a execução.

03 / EDITOR • MÁQUINAS DE ESTADOS

Leia a máquina desenhada pelo código

A aba Estados desenha transições reconhecidas nos arquivos C do projeto. Use o mapa para prever uma mudança, localizar a atribuição responsável e decidir onde medir.

Da seta à instrução

01 Abra a lanterna. Na aula 11 da UTFPR, abra o primeiro exemplo no editor. Entre em Estados. Se o projeto tiver várias máquinas, use o seletor Máquina para escolher a variável e o arquivo.

02 Localize uma transição. Siga a seta de FIXA para PISCA. Selecione a transição para abrir a linha em que o programa atribui o destino. No código, confira a condição de entrada e a origem do evento PRESSIONOU.

03 Faça uma alteração verificável. Troque o destino PISCA por APAGADA nessa atribuição. Consulte Estados novamente. Para testar o comportamento, compile, execute e pressione KEY duas vezes. Observe PC13 e a variável estado em Watches.

Recurso	O que conferir
Estados	Origem, destino e condição extraídos do código atual. Um diagrama parcial pede leitura dos ramos que o analisador não reconhece.
Fluxograma	Caminhos de execução dentro da função. O teste do botão também aparece em funções que somente tratam a entrada.
Analisador lógico e Watches	A borda realmente recebida, o estado em memória e o nível de saída ao executar o estímulo.

Na aula 15, acompanhe separadamente cada máquina cooperativa. Localize quem produz o evento de pedido e quem consome a confirmação. Anote a ordem das duas chamadas no laço principal antes de prever o resultado.

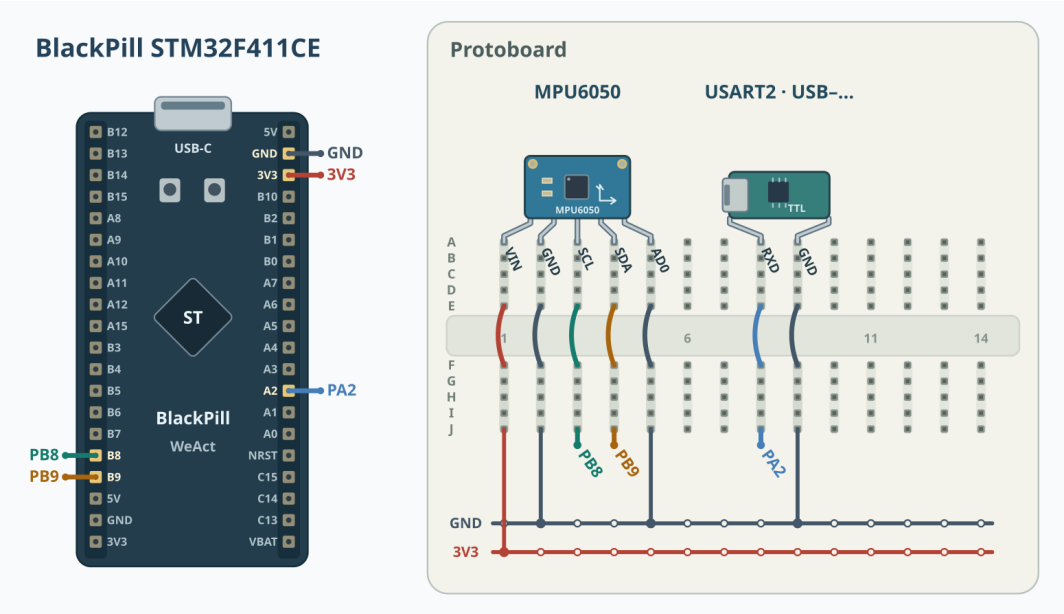
Abrir a aula da lanterna →

Abrir a aula de máquinas cooperativas →

04 / CIRCUITO • ESTÍMULOS

Do pino ao periférico

Ver circuito abre Chip e conexões com o projeto da prática. Ali o botão, o acelerômetro e o display respondem ao programa em execução. Montagem no protoboard apresenta a fiação para reproduzir o circuito físico.



Montagem do inclinômetro da bancada. O MPU6050 usa PB8 para SCL e PB9 para SDA. O adaptador recebe os dados transmitidos por PA2.

Faça o mesmo percurso nos dois ambientes

Na aula 9, abra o circuito, execute o programa e mova o controle virtual do MPU6050. No terminal, acompanhe a leitura de Ax. No LED em PC13, observe o estado do alarme durante a ida e a volta pelo mesmo ângulo.

Na folha de montagem, siga os nomes das redes e confira alimentação, GND e os terminais do modelo do sensor. O botão KEY e o LED PC13 já estão na placa. Ao alterar um pino em Módulos externos, confira também a configuração correspondente no código.

> Experimente o display
> Na aula 10, o Exemplo aplicado conta entradas no ST7789. Pressione e solte KEY para registrar cada entrada. Mantenha-o pressionado em uma das tentativas e confirme se o contador cresce apenas uma vez.

Abrir o contador de lotação no ST7789 →

05 / DISPLAY • ENSAIO GUIADO

Investigue o contador de lotação

Um visitante pressiona KEY ao entrar. O painel ST7789 conta uma entrada por acionamento válido, mostra quantas vagas foram ocupadas e muda de estado quando atinge a capacidade.



3 entradas · aberto

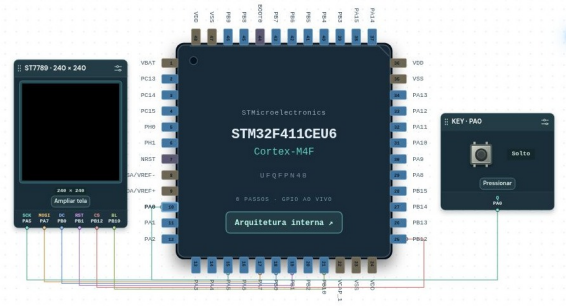
8 entradas · lotado

Capturas do display geradas pela execução do firmware CMSIS no simulador. Três entradas deixam o painel aberto. Na oitava, a última célula fica vermelha e aparece LOTADO.

01 Abra a atividade e compile. Na aula 10, selecione Exemplo aplicado e abra Ver circuito. Use F7 e F5. No módulo ST7789, amplie a tela. Comece com 0/8 e a tecla solta.

02 Separe pressão e soltura. Pressione e solte KEY três vezes, deixando o contato estabilizar entre as ações. Espere 3/8. Mantenha a quarta pressão e observe o valor até soltar. O código usa 20 ms para validar a mudança de estado do botão.

03 Teste o limite. Complete oito entradas e tente uma nona. Registre o número, o estado do painel e a cor da última célula. Para experimentar outra capacidade, altere CAPACIDADE em main.c e recompile.



Bancada da aula 10 antes da execução. O módulo à esquerda usa SPI. O botão à direita representa KEY, integrado à BlackPill em PA0.

> Conte os bytes que mudam
> Com capacidade 8, cada célula da barra ocupa 22 × 16 pixels. Em RGB565, a cor daquela célula exige 704 bytes. O total da atualização também inclui o novo número, os comandos da janela e, na última entrada, o texto de lotação.

Abrir o circuito do contador →

06 / INSTRUMENTOS • ANALISADOR LÓGICO

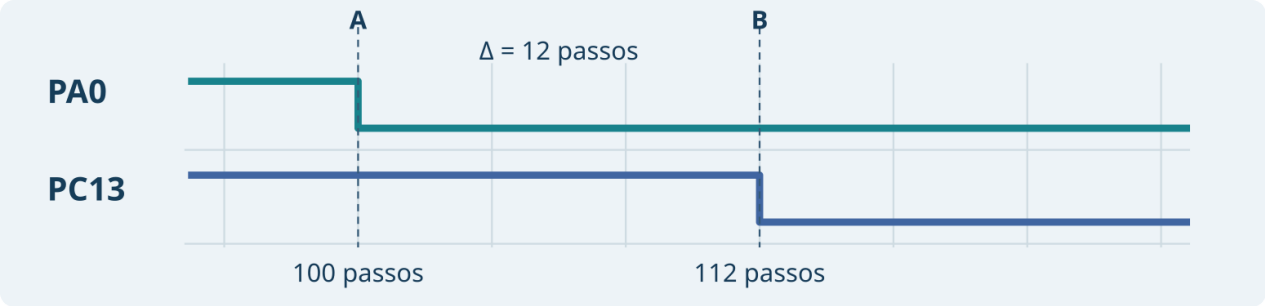
Meça entre duas bordas

Escolha os canais conforme a pergunta. Para investigar o tempo de resposta de uma saída GPIO, é preciso registrar o instante de entrada e a transição de saída correspondente.

01 Defina os canais antes de executar. Na aula 6, abra Analisar sinais e selecione PA0 e PC13. Ajuste Janela para que as duas transições caibam na mesma captura.

02 Capture o evento. Execute, pressione e solte KEY. Use Congelar para reter o trecho. Última borda · CH1 ajuda a encontrar uma transição recente do primeiro canal.

03 Posicione A e B. Clique na descida de PA0 e depois na descida de PC13. A diferença B − A expressa o intervalo de resposta observado. Para medir período, use duas bordas consecutivas de mesma polaridade no mesmo canal.



Exemplo de leitura em passos do motor. A resposta ocupa 12 passos entre as bordas escolhidas. Os números ilustram a medição.

Conserve a unidade original

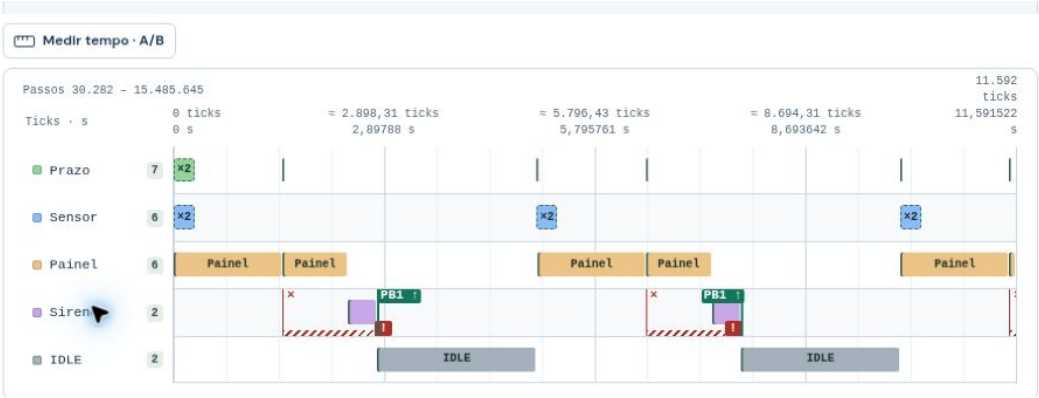
Passos do motor indexam a execução simulada. Quando o painel também apresenta segundos simulados ou ticks do kernel, registre o valor e a unidade exibidos. Uma escala com ≈ indica estimativa. Duas bordas visualmente coincidentes podem ficar separadas no CSV.

> Exportar a evidência
> Exporte CSV para comparar instantes numericamente e VCD para visualizar ondas em outra ferramenta. O VCD representa a sequência com uma unidade nominal por passo. Calcule o tempo físico com uma medição em hardware.

07 / THREAD VIEWER • LEITURA DA CAPTURA

O que o escalonador consegue mostrar

Comece pela decisão que deseja explicar. Qual tarefa estava pronta quando a CPU ficou livre? Para que ela precisava de CPU? A resposta costuma exigir o estado do kernel, um evento do programa e uma indicação do periférico.



Captura da sirene com duas perdas completas. A barra roxa mostra Sirene selecionada. × marca o prazo vencido, a hachura termina na subida verde de PB1 e ! marca o registro posterior. ×2 reúne duas seleções próximas na mesma linha.

01 Preserve a ocorrência. F7 compila e F5 executa. Em Editor e sinais, abra Threads. Congelar captura retém o trace. Para confrontar o trace com um contador do programa no mesmo instante, use Pausar CPU antes de ler Progresso da prática.

02 Encontre a tarefa no instante escolhido. Use Enquadrar para localizar o conjunto retido. Abra Histórico de escalonamento, filtre por nome e localize uma seleção. Amplie com a roda ou o gesto de zoom e arraste para mover a janela. Clique em uma barra e consulte Estados das threads. O cabeçalho indica cursor, fim da janela ou captura atual.

03 Meça duas coisas separadamente. Em Medir tempo · A/B, Posicionar A e Posicionar B definem o intervalo. Alinhar às seleções facilita medir um despacho inteiro. Para usar as bordas de um GPIO, desmarque o alinhamento. Ampliar A–B mostra a região em detalhe.

> Um intervalo curto pode atravessar zero ticks
> Uma medição de 0,000213 s ocupa cerca de 0,213 período a 1.000 Hz e pode começar e terminar no mesmo tick do kernel. Registre o relógio medido, a diferença dos contadores inteiros e a frequência.

Abrir o projeto da primeira investigação →

08 / INVESTIGAÇÃO 01 • ESPERA POR UM EVENTO

Por que Traffic desaparece?

Na travessia de pedestres, Button ganha pequenos intervalos de CPU enquanto o botão está solto. Traffic tem prioridade maior e passa quase todo esse período sem uma barra. Verifique se está aguardando uma requisição válida.

src/main.c

sample_button · signal_traffic · request

Button P1 lê PA0. Traffic P2 controla o sinal. O projeto usa um semáforo binário.

01 Prepare o controle. Abra a prática Travessia de pedestres da trilha intermediária e clique em Abrir projeto da prática. Mantenha POLL_MS em 100 e KEY solto. Compile, execute e, em Threads, congele um trecho depois da inicialização.

02 Leia a janela sem estímulo. Filtre Button em Histórico de escalonamento. Ache duas consultas vizinhas, distante a primeira da inicialização. Confira que reaparecem por volta de 100 ticks (0,100 s). Consulte Traffic em Estados das threads entre as consultas. Ela deve estar em Bloqueadas.

03 Produza uma liberação. Volte ao vivo. Em Controles do circuito, pressione Pedido · PA0 e mantenha até acender o âmbar em PB1. Solte a tecla. Congele a captura. Em Latência de E/S, escolha PA0 / Descida e PB1 / Subida e use Medir na linha do tempo. Procure a seleção de Traffic logo após a leitura de Button.

04 Encontre a condição no código. Em sample_button, lab_button() lê PA0. Só a transição false→true chama xSemaphoreGive(request). Em signal_traffic, xSemaphoreTake(request, portMAX_DELAY) libera a continuação que acende o âmbar. Segurar a tecla não gera um novo pedido a cada consulta.

> Se Traffic ainda não aparecer
> Confira primeiro se PA0 desceu e se Button leu a porta depois da borda. Um toque inteiramente entre consultas pode escapar. Se PB1 subiu, procure o escritor no histórico e amplie o intervalo. Se não há leitura de estados no ponto, capture outra travessia.

Abrir a prática da travessia →

09 / INVESTIGAÇÃO 01 • REPETIR COM A FASE CONTROLADA

Meça a espera pela próxima leitura

Faça a mesma pressão em duas fases do laço de Button. O experimento separa a espera pela consulta do tempo entre a consulta e a saída. Use o firmware intermediário da página anterior.

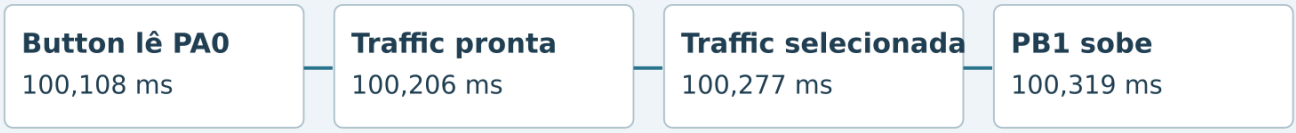
01 Pegue a amostra rápida. Com a CPU parada e PA0 solto, marque breakpoint na linha int pressed = lab_button(); de sample_button. Execute até parar nessa linha. Retire o breakpoint, pressione PA0 no circuito e mantenha a tecla enquanto continua a CPU. Solte após o âmbar.

02 Pegue a amostra lenta. Reinicie com PA0 solto. Desta vez pare na linha xTaskDelayUntil(&next, pdMS_TO_TICKS(POLL_MS)), depois da leitura. Repita a pressão, volte a executar e capture as mesmas duas bordas.

03 Reduza o intervalo de consulta. Em src/main.c, troque apenas POLL_MS de 100 para 20. Recompile e repita as duas fases. Para voltar à referência, restaure 100 e recompile.

Fase da pressão	POLL_MS = 100	POLL_MS = 20
Antes da leitura	≈ 0,213 tick · 0,000213 s	≈ 0,213 tick · 0,000213 s
Depois da leitura	≈ 99,920 ticks · 0,099920 s	≈ 19,920 ticks · 0,019920 s

Recorte ampliado após 99,709 ms de espera pela consulta



PA0 caiu em 0,399 ms. Do nível de entrada à luz âmbar passaram 99,920 ms.

Execução de 100 ms com pressão após a consulta. O detalhe amplia os últimos 0,211 ms entre a leitura de PA0 e a saída PB1. Os instantes em ms usam o relógio simulado.

> Teste a hipótese de atraso de despacho
> Traffic ficou pronta em 100,206 ms e foi selecionada em 100,277 ms, cerca de 0,071 ms depois. A maior parcela ocorreu antes da leitura em 100,108 ms. Registre sua fase, o intervalo PA0↓→PB1↑ e a maior parcela.

10 / INVESTIGAÇÃO 02 • PRAZO DE PONTA A PONTA

Localize o atraso da sirene

O requisito começa quando o sensor aceita o pedido e termina quando o alarme liga. A Sirene trabalha por 400 ms, mas a resposta pode levar mais de 2,8 s. Descubra em qual tarefa esse tempo foi gasto.

src/main.c + include/sinais_prazo.h

sensor · painel · sirene · entregar_resposta

Sensor P4 acorda Painel P3 e Sirene P2 a cada 5.000 ms. O prazo relativo é 1.500 ms.

01 Capture uma solicitação completa. Abra o Exemplo resolvido da aula 20. Compile e execute até o primeiro registro “Sirene ligada” no Terminal serial. Pause a CPU, abra Threads e congele a captura. Em Ocorrência, escolha o primeiro pedido e use Ir ao atraso.

02 Meça o atraso com A/B. Em Medir tempo · A/B, desmarque Alinhar às seleções. Com Ir ao atraso já ampliado, coloque A sobre a marca de vencimento × e B em Sirene ligada · PB1 ↑. Leia B−A. O pedido anterior não tem glifo próprio na linha da Sirene. Para o intervalo completo, use o registro da ocorrência ou o terminal.

03 Leia três instantes distintos. Amostra #1: o pedido chegou no tick 1, o vencimento caiu no tick 1.501 e PB1 subiu no tick 2.802. A/B sobre a hachura mede o atraso de 1.301 ticks (1,301 s). Do pedido até PB1, a resposta totalizou 2.801 ticks (2,801 s).

04 Explique de onde vieram os números. No histórico, encontre Sirene selecionada no tick 2.402. Do pedido à primeira seleção passaram 2.401 ticks, e daí à resposta, 400. O “!” é o registro escrito pelo firmware, 59 passos depois da borda PB1 neste ensaio. A primeira barra de Sirene começa antes de o alarme estar ligado.

> Quando um programa próprio tem prazo?
> O trace lê trocas de tarefa do kernel. Nesta atividade, cortex_pedido_v1 e cortex_deadline_v1 registram pedido e prazo, e o visualizador também identifica a saída PB1. Em outro projeto, escolha os eventos e o limite a partir do requisito. Duas bordas com A/B permitem medir a resposta mesmo sem a anotação automática.

Abrir o firmware de referência da sirene →

11 / INVESTIGAÇÃO 02 • INTERVENÇÃO SOB CONTROLE

Mantenha o painel aceso e libere a CPU

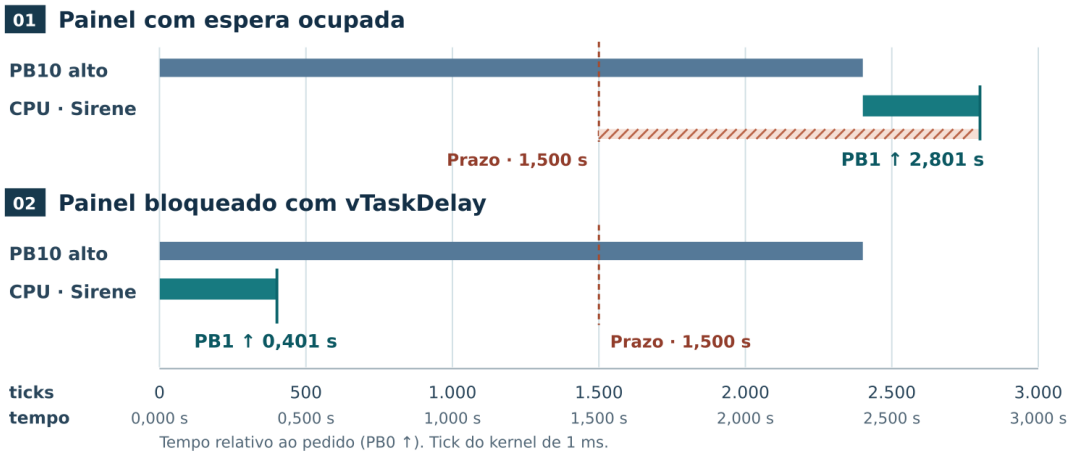
O painel precisa manter a saída azul por 2,400 s. Compare as mesmas duas solicitações, com a mesma prioridade e o mesmo tempo de luz, alterando somente a maneira como painel() espera.

Em painel(), dentro do laço

Referência	ocupar_janela(ticks_ms(ANIMACAO_PAINEL_MS));
Experimento	vTaskDelay(ticks_ms(ANIMACAO_PAINEL_MS));

01 Abra a segunda versão. O Exemplo aplicado da aula 20 já contém a substituição. Compare src/main.c, confirme PRIO_PAINEL = 3 e PRIO_SIRENE = 2, recompile e execute a primeira solicitação nas mesmas condições.

02 Confira o efeito no painel. No terminal, leia “Sirene ligada em 401 ms”. Uma ocorrência atendida deixa de mostrar Ir ao atraso. Consulte a tabela de prazos do relatório e examine o canal azul PB10. Nas duas versões a subida ocorreu no tick 2 e a descida no tick 2.402.



Primeiro pedido em duas execuções CMSIS + FreeRTOS a 1.000 Hz. O marcador de vencimento parte da mesma liberação. Os valores são medições do simulador.

> Compare o relatório com os ticks do firmware
> Com a espera bloqueante, Sirene executou no tick 2 e respondeu no tick 402. Margem de 1.099 ticks (1,099 s). Nos relatórios abaixo, os cursores sobre PB0↑ e PB1↑ mediram 2,800445 s e 0,400703 s. Os contadores inteiros do firmware deram 2.801 e 401 ticks. Registre a origem de cada número.

Abrir a versão da aula 20 com espera bloqueante →

Relatório exportado · espera ocupada →

Relatório exportado · espera bloqueante →

12 / INVESTIGAÇÃO 02 • DEPURAÇÃO EM INSTRUÇÕES

Pare a CPU antes de a sirene ligar

A aba Threads mostra quando a Sirene executou. Para achar a instrução que muda PB1, use uma segunda execução com breakpoint. Guarde a medição contínua da página anterior antes de começar.

01 Pare no código que entrega a resposta. Na aula 20, abra src/main.c e encontre a chamada entregar_resposta("Sirene ligada", inicio) em sirene(). Siga para include/sinais_prazo.h. Use F7 para compilar. Clique no texto da linha led_externo(verde, 1); e pressione F9 para marcar o breakpoint. Era a linha 73 no build medido. Pressione F5 para executar até parar. Se o modo de depuração ficar lento, teste a versão corrigida, em que a chamada ocorre mais cedo.

02 Veja qual contexto foi interrompido. Sem retomar, em Threads consulte a tarefa selecionada. Em Watches & breakpoints adicione PC e PSP. PC permite cruzar a instrução com Código de máquina; PSP é o ponteiro da pilha de processo neste contexto de tarefa. F11 executa uma instrução. Siga a chamada até a escrita em GPIOB→BSRR.

03 Observe o pino mudar. Adicione a expressão mem32(0x40020414) & 2. O endereço é GPIOB→ODR na STM32F411. A máscara 2 isola PB1. Neste build, o PC parou em 0x080005E0 e a watch mostrou 0. Após sete F11 ainda valia 0. No oitavo passo, mostrou 2 e o PC chegou a 0x080005B2. Outra compilação pode mudar os endereços e a contagem de passos.

Campo	O que escrever no caderno
PC antes / depois	Endereços exibidos por seu build, instrução de escrita e tarefa selecionada
PB1 antes / depois	Watch 0 antes de PB1 ligar e 2 depois da escrita em BSRR
Evento de programa	A saída muda antes da mensagem “Sirene ligada” no terminal

> Separe a depuração da medição
> O breakpoint pausa o núcleo inteiro, incluindo as demais tarefas. Com F11 você observa a ordem dos efeitos. Remova os pontos, reinicie e repita sem pausas para medir os tempos de resposta.

Abrir o firmware para depurar →

13 / INVESTIGAÇÃO 03 • FILA E CAPACIDADE

A fila copiou ou perdeu o pulso?

Uma produtora envia três estruturas em rápida sequência e reutiliza o mesmo espaço de memória. Verifique a cópia pelo conteúdo que a consumidora realmente recebe. Use a aula 17, Exemplo aplicado, com pulsos P, M e G.

src/main.c

produzir · exibir · pulsos · ultimo_rascunho

Produtora P2 envia com portMAX_DELAY. Consumidora P1 alterna PC13 e escreve na serial.

01 Preveja a sequência. Localize Pulso rascunho, a atribuição rascunho = sequencia[i] e xQueueSend(pulsos, &rascunho, portMAX_DELAY). Logo depois, produzir troca o nome por “?” e a duração por 1. Se a fila guardasse somente o endereço dessa variável, os dados poderiam mudar antes da leitura.

02 Meça três saídas. Compile, execute e aguarde dois grupos no terminal. No Analisador lógico, escolha PC13 e congele. A queda acende o LED. Meça queda→subida no primeiro P, M e G. Em Threads, localize o tick 50 dentro de P e leia o estado das duas tarefas.

03 Relacione saídas e cópia. Confirme no Terminal serial o registro Pulso P, Pulso M, Pulso G. O comprimento das três saídas confirma que os valores originais de duração também chegaram. Na capacidade 2, o terceiro envio pode suspender a produtora até a consumidora retirar o primeiro item.

Item	Queda e subida de PC13	Duração medida
P	tick 1 → tick 76	75 ticks · 0,075 s
M	tick 226 → tick 376	150 ticks · 0,150 s
G	tick 526 → tick 826	300 ticks · 0,300 s

> CPU ociosa durante o pulso
> No tick 50, PC13 está baixo enquanto Produtora e Consumidora estão bloqueadas e IDLE foi selecionada. O nível escrito permanece no GPIO durante vTaskDelay. A largura de P mede quanto tempo o LED ficou aceso.

Abrir a fila P/M/G da aula 17 →

14 / INVESTIGAÇÃO 03 • PRESSÃO SOBRE A FILA

Meça o custo de retirar uma vaga

A saída começa pelo mesmo grupo P/M/G com uma e duas vagas. Para testar a espera por espaço, meça o intervalo entre o início de dois grupos e descubra quando a produtora pôde sair do terceiro envio.

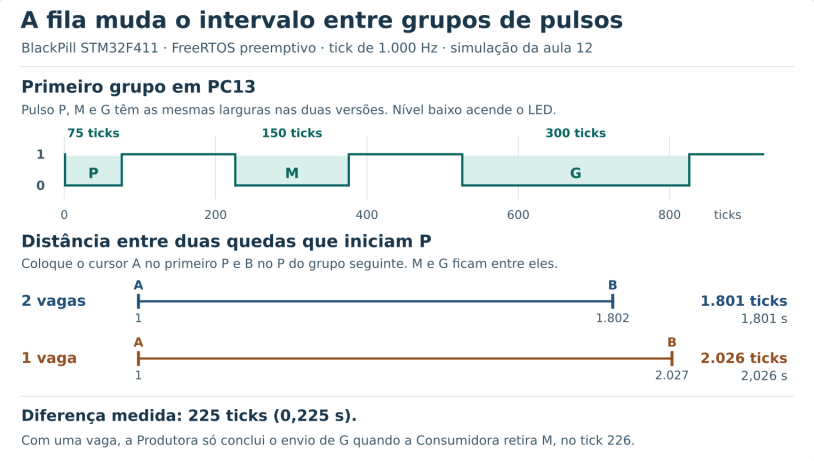
Em main(), altere apenas a capacidade

Referência	pulsos = xQueueCreate(2, sizeof(Pulso));
Experimento	pulsos = xQueueCreate(1, sizeof(Pulso));

01 Registre primeiro o período original. Antes de executar, no Analisador lógico ajuste Janela para 4.194.304 passos e escolha PC13. Congele assim que surgir o segundo P na serial. Marque A na queda que inicia P e B na queda que inicia o P seguinte, com M e G entre eles. Com duas vagas, os ticks foram 1 e 1.802. O clique no gráfico dá uma leitura aproximada.

02 Repita e procure o desbloqueio. Troque a capacidade para 1, recompile e reinicie. Congele o primeiro grupo. Em Histórico de escalonamento, localize a seleção de Produtora no tick 226. Ela volta a executar quando a consumidora retira M e abre a vaga necessária para enviar G.

03 Compare períodos equivalentes. Com uma vaga, o segundo P começa no tick 2.027. O período medido cresceu de 1.801 para 2.026 ticks (1,801 para 2,026 s). Os pulsos conservaram as larguras. Restaure a capacidade 2 e confira o resultado original.



Quedas de PC13 e estado do produtor nas duas execuções. A alteração do período entre grupos aparece 225 ticks depois, embora os pulsos do primeiro grupo coincidam.

> O que a captura identifica
> xQueueSend usa espera indefinida. A contagem na serial mostra que os três itens chegaram. A relação entre a retirada de M e a vaga disponível vem da ordem do código, dos estados e das seleções. Apenas contar os despachos não informa o número de mensagens.

15 / INVESTIGAÇÃO 03 • CONTAGEM DE DESCARTES

Diferencie fila cheia de mensagem perdida

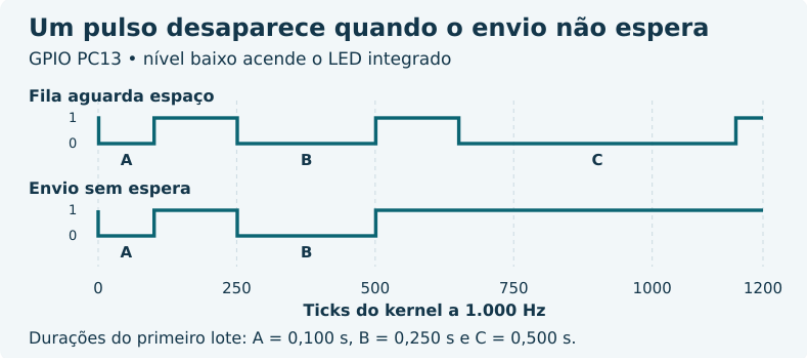
Mude para o Exemplo resolvido da aula 17. As mensagens desta versão se chamam A, B e C e têm outras durações. Agora a pergunta é quantas tentativas viram saídas quando a produtora desiste de uma fila cheia.

01 Conte três lotes sem perda. Mantenha a fila com 2 vagas. Em src/main.c, observe xQueueSend com portMAX_DELAY e o intervalo de 1.800 ms entre os grupos. Execute até o tick 5.000 (5,000 s). No terminal, conte três aparições de cada letra. As larguras de PC13 foram 100, 250 e 500 ticks.

02 Torne o descarte observável. No envio em produzir, use o trecho abaixo. Ele faz a mesma tentativa e informa qual item não conseguiu entrar. O argumento 0 devolve o controle logo se não houver vaga. Recompile e parta do reset com as mesmas prioridades.

Substitua a única chamada a xQueueSend em produzir()

```
if (xQueueSend(pulsos, &sequencia[i], 0) != pdPASS) {
    char msg[] = "Descartado A\r\n";
    msg[11] = sequencia[i].nome;
    lab_print(msg);
}
```



PC13 no primeiro grupo do ensaio com espera indefinida e com timeout zero. A duração do pulso corresponde à mensagem que a consumidora retirou.

> Feche o balanço
> Aos 5.000 ticks, o ensaio com timeout zero totalizou 9 tentativas, 6 recebimentos e 3 descartes de C. A indicação de descarte apareceu antes de Pulso A e Pulso B em cada grupo, pois a produtora P2 ainda executava. Com a espera original chegaram os 9 itens. Restaure portMAX_DELAY para comparar novamente.

Abrir a referência A/B/C da aula 17 →

16 / INVESTIGAÇÃO 04 • MUTEX E DEPENDÊNCIA CIRCULAR

Duas tarefas bloqueadas provam deadlock?

Uplink e Archive fazem transferências que precisam de rádio e armazenamento. Com a CPU em IDLE e as duas tarefas bloqueadas, uma fotografia isolada é compatível tanto com espera normal quanto com travamento.

src/main.c

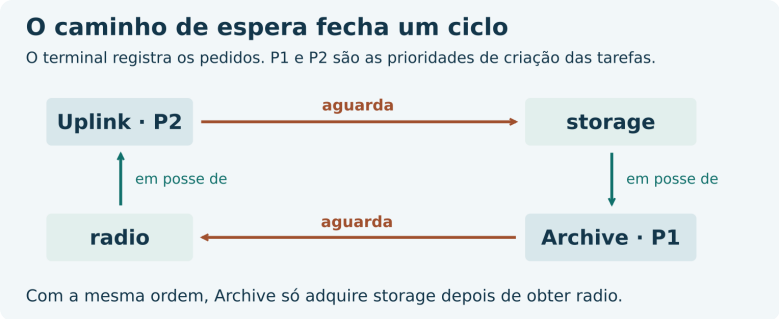
uplink · archive · complete_transfer · completed_transfers

O objeto radio e o objeto storage são mutexes distintos. A compilação inicial usa FIX_CONCURRENCY = 0.

01 Cruze contador e tempo. Execute até depois de 2.000 ticks (2,000 s). Em Threads, pause a CPU e leia Progresso da prática → Transferências concluídas. Na serial, as duas tarefas anunciaram o recurso que ainda precisam. Localize os dois últimos pedidos.

02 Reconstrua o ciclo a partir do código. Em uplink, a tarefa toma radio, espera 400 ms e tenta storage sem devolver radio. Em archive, o ramo original toma storage, espera 200 ms e tenta radio. Cada segunda chamada a xSemaphoreTake usa portMAX_DELAY. Nenhuma das tarefas alcança a devolução do primeiro recurso.

03 Quebre apenas uma dependência. Mude #define FIX_CONCURRENCY 0 para 1. Recompile e repita. Agora as duas tomam radio primeiro e storage depois. Confira os dois registros “transfer OK” e o avanço de completed_transfers nas mesmas janelas.



As setas de espera e posse vêm da ordem das chamadas a xSemaphoreTake. As prioridades indicadas são as de criação.

Abrir o projeto dos dois mutexes →

17 / INVESTIGAÇÃO 04 • VERIFICAÇÃO INDEPENDENTE

Prove se as transferências avançam

Repita o código defeituoso e o corrigido até o mesmo tick. As mensagens de aquisição mostram as intenções das tarefas. O contador em complete_transfer() responde à pergunta decisiva sobre a conclusão do trabalho.

Quantas transferências terminam?

Contador completed_transfers • quatro leituras do firmware

Instante do kernel Relógio simulado	500 0,5 s	1000 1,0 s	1500 1,5 s	2000 2,0 s
Ordem inversa	**0**	**0**	**0**	**0**
Ordem comum	**1**	**2**	**3**	**4**

Em cada leitura, Uplink e Archive estão bloqueadas e IDLE está selecionada.

Transferências concluídas nas simulações com FIX_CONCURRENCY = 0 e 1. O incremento conta uma transferência concluída, independentemente de qual tarefa executou.

Se os contadores…	Confira na serial e no código
permanecem em 0	Procure as duas mensagens de pedido e a ausência de transfer OK. Localize a segunda aquisição que cada tarefa espera.
aumentam de 0 para 4	Procure uma conclusão de cada origem. Confira em archive que a ordem de aquisição agora coincide com a de uplink.

01 Corrobore a correção. Em cada versão pause a CPU depois de 2.000 ticks, leia Transferências concluídas e confira as últimas mensagens na serial. Após a correção, aparecem Uplink: transfer OK e Archive: transfer OK. Só volte a executar depois de copiar os dois registros.

02 Distingua a espera temporária. Volte aos instantes de 500 e 1.500 ticks. Ao ver as duas tarefas bloqueadas, consulte o contador nas observações vizinhas. Se há incremento, elas conseguiram sair da espera entre suas leituras.

> A hipótese rival sobrevive a um único instante
> Nos quatro instantes, as duas versões mostraram IDLE selecionada, Uplink e Archive bloqueadas. A diferença aparece no progresso entre observações e nas duas chamadas de aquisição. Guarde o contador inicial, final e as mensagens que fecham o ciclo.

18 / INVESTIGAÇÃO 05 • PRIVAÇÃO DE CPU

Report está pronta. Por que não executa?

A aquisição continua gerando amostras e o relatório deixa de avançar. Para distinguir de um deadlock, procure quem está usando a CPU enquanto o contador de relatórios permanece imóvel.

src/main.c

acquisition · report · sample_count · report_count

Acquisition P3 espera a hora de uma amostra. Report P1 deveria alternar o LED a cada 250 ms.

01 Capture após a partida. Execute com FIX_CONCURRENCY em 0. Aguarde “REPORT ready” e duas mensagens “ACQUIRE sample”. Pause em Threads. Em Estados das threads, procure Report em Prontas. Abra um trecho posterior à partida e descubra qual tarefa aparece nas seleções.

02 Identifique por que a prioridade importa. No laço de acquisition, a leitura repetida de xTaskGetTickCount() mantém P3 pronta até o próximo período. Em Progresso da prática, confira os contadores após 500, 1.000, 1.500 e 2.000 ticks. No binário original, só o contador de aquisição evolui.

03 Teste a espera periódica. Troque FIX_CONCURRENCY para 1. O ramo corrigido chama xTaskDelayUntil(&last_release, period). Recompile, repita a janela e confira “REPORT advanced” e as alternâncias de PC13. A espera entre amostras deixa tempo para Report trabalhar.

A aquisição avança. E os relatórios?

Contador report_count • quatro leituras do firmware

Instante do kernel Relógio simulado	500 0,5 s	1000 1,0 s	1500 1,5 s	2000 2,0 s
Aquisição com espera ocupada	**0**	**0**	**0**	**0**
Aquisição com espera periódica	**2**	**4**	**6**	**8**

Nos dois ensaios, sample_count registra 0, 1, 2 e 3 amostras nesses instantes.

Contador de relatórios em quatro instantes da execução. Na versão de espera ocupada, a aquisição continuava produzindo amostras mesmo com o contador de relatórios parado.

> Diagnóstico para o relatório
> A 2.000 ticks (2,000 s), sample_count era 3 nas duas versões, e report_count passou de 0 para 8. A assinatura observada foi Report pronta, P3 selecionada e nenhum relatório novo. Para comparar com o deadlock, registre também o estado e a tarefa em execução na janela.

Abrir o projeto de starvation →

19 / DIAGNÓSTICO ADICIONAL • CICLO DE VIDA

Quando só resta IDLE

No exemplo Duas tarefas, duas pilhas, a janela recente pode exibir exclusivamente IDLE. Antes de concluir que o scheduler parou, descubra se as tarefas ainda deveriam existir.

01 Examine o projeto. Na galeria de projetos de exemplo, procure Duas tarefas, duas pilhas. Small tem prioridade 2 e Large tem prioridade 1. Em src/main.c, encontre a medida da menor folga de pilha e o encerramento explícito vTaskDelete(NULL) de cada tarefa.

02 Recupere o início da execução. Depois que o terminal imprimir as duas medições, pause a CPU. Em Threads, procure Small → Large → IDLE em Últimas trocas. Enquadrar inclui a longa execução de IDLE e pode comprimir as primeiras seleções. Abra Histórico de escalonamento e use o filtro Small ou Large para localizar cada despacho.

03 Verifique o fim de vida. O ensaio reteve os três despachos. Ao inspecionar os dois contextos após o retorno das respectivas tarefas, a condição é Excluída. O aviso de registros anteriores descartados só aparece quando houve descarte. Neste caso as chamadas de vTaskDelete explicam a ausência posterior.

04 Varie uma reserva. Mude o argumento de pilha de Small de 256 para 384 na chamada a xTaskCreate, recompile e reinicie. Preserve as 512 palavras de Large. Compare a folga mínima em palavras, a ordem de seleção e o número de relatórios da serial.

Execução	Menor folga de Small	Menor folga de Large
Reservas 256 / 512 palavras	232 palavras	488 palavras
Reservas 384 / 512 palavras	360 palavras	488 palavras

> O que essa medição permite concluir
> A execução percorreu o mesmo caminho com duas reservas diferentes. Somar 128 palavras à reserva de Small acrescentou 128 à folga medida. Outras chamadas, interrupções e entradas de dados podem mudar a menor folga de uma execução mais longa.

20 / THREAD VIEWER • LIMITES E REGISTRO

Quando falta uma barra na captura

O histórico conserva uma quantidade finita de seleções. Uma janela curta, um período longo de bloqueio ou a perda dos registros mais antigos produzem telas parecidas. Faça estes testes antes de apontar uma falha no código.

Sintoma	Verificação	Como avançar
Linha vazia, LED funcional	Histórico de escalonamento, filtro por nome, em Toda a execução	Localize uma seleção retida e amplie a janela ao redor dela. Confira a tarefa que escreveu o GPIO.
Tarefa aparece em Bloqueadas	Verifique o instante no cabeçalho de Estados das threads	Leia a chamada que pode bloquear e procure a liberação por evento, tempo ou recurso.
Uma barra longa parece única	Verifique ×N e o número de seleções mostrado	Amplie junto ao ponto de interesse. As barras compactadas agrupam seleções vizinhas.
O cursor não mostra estado	Confira se o painel informa histórico descartado ou sem registro	Capture uma nova ocorrência. A faixa de estados retidos pode diferir do histórico de despachos.
O rastreio não inicia	Confira se o build é do projeto, se o scheduler iniciou e se o binário possui os símbolos do kernel reconhecido	Use o terminal e breakpoints em xTaskCreate e vTaskStartScheduler para localizar uma falha de partida.

O que fica guardado no FreeRTOS

Nesta versão, o modelo mantém até 65.536 seleções no histórico estendido e 1.024 seleções no conjunto recente. O histórico de mudanças de estado tem retenção própria. O tempo coberto varia com a taxa de eventos. Fixar lista congela a visualização daquele histórico. Exportar relatório em Threads conserva a análise da janela selecionada.

> Registro mínimo de uma investigação
> Escreva projeto e versão, frequência de tick, hipótese, estímulo, evento A, evento B, diferença medida com unidades, estados e uma evidência independente. Altere uma única variável, repita a mesma janela e anexe as duas capturas. Liste o que ainda depende de medir na placa.

Baixar os resultados de referência em CSV →

21 / ARQUITETURA • MEMÓRIA E EXCEÇÕES

Siga o dado dentro do núcleo

Comece pela pergunta que o código permite responder. Um registrador mostra o valor produzido pela instrução. A memória mostra o que foi efetivamente gravado no endereço.

Experimento com uma chamada de função

01 Localize a rotina da aula 4. Abra a instrução PUSH e anote o MSP antes da sua execução. Na BlackPill F411, a SRAM começa em 0x20000000. A aba Memória & pilhas permite consultar o endereço indicado pelo ponteiro.

02 Avance uma instrução. Use F11. Ao salvar dois registradores de 32 bits, a diferença entre o valor antigo e o novo do MSP é de 8 bytes. Leia as duas palavras e identifique a cópia do endereço de retorno.

03 Acompanhe o retorno. Avance até POP e confira MSP e PC. Em Arquitetura, siga a instrução e o caminho de dados que o modelo representa. Compare a leitura da SRAM com os endereços efetivamente acessados.

Pergunta	Instrumento	Registro a guardar
Qual instrução vem a seguir?	Código de máquina / CPU	Endereço no PC e instrução
Quais bytes mudaram?	Memória & pilhas	Endereço, largura e conteúdo
Por que a CPU entrou na ISR?	NVIC / Eventos	Interrupção, prioridade e estado
O que o programa reservou?	Arquitetura / memória	Capacidade da placa e regiões informadas pelo build

> Por que a ordem importa
> Um breakpoint na entrada da ISR captura a CPU depois da aceitação da exceção. Para distinguir solicitação e atendimento, registre também a origem da pendência e os estados de máscara e habilitação.

Abrir o exemplo de pilha →

22 / DOCUMENTAÇÃO • ENTREGA

Registre um ensaio repetível

A trilha UTFPR reúne 20 aulas. Cada uma apresenta dois exemplos resolvidos e três laboratórios, A, B e C. São 40 exemplos e 60 laboratórios. Entregue um relatório por aula, com as três respostas e as evidências correspondentes.

Campo do registro	Exemplo de preenchimento
Projeto e versão	UTFPR 06, Laboratório A, main.c, cópia do projeto usada no ensaio
Configuração	BlackPill STM32F411CE, botão PA0 e LED PC13 ativos em nível baixo
Estímulo	Partir com KEY solto, pressionar, manter e soltar uma vez
Critério	Verificar a sequência 1 → 0 → 1 em PA0 e a resposta em PC13
Resultado próprio	Instantes A e B, diferença, unidades e captura com ambos os canais
Interpretação	Comparar com a previsão e justificar a discrepância observada

Prepare o arquivo para entregar

01 Preencha o relatório da aula. Registre identificação e, em A, B e C, descreva o procedimento, os parâmetros, a medição e a interpretação. Use capturas que mostrem canais, escalas e cursores.

02 Exporte e reabra o HTML. Confira, fora do simulador, se as figuras estão legíveis e se as três respostas estão completas. Envie o arquivo e o projeto pelo canal indicado pelo professor.

03 Preserve os arquivos de trabalho. Meus projetos e os rascunhos usam o armazenamento local do navegador. Baixe o projeto JSON para reabrir na bancada. Exporte o ZIP para continuar no ambiente de desenvolvimento quando necessário.

> Continue em outro computador
> O HTML da aula reúne as três respostas. Salvar rascunho JSON conserva as respostas e figuras para edição posterior. Abra esse arquivo em Abrir rascunho JSON, no relatório da mesma aula. Em Threads, Exportar relatório guarda a janela de escalonamento.

Abrir o manual de laboratórios da disciplina →

23 / PLACA REAL • VALIDAÇÃO

Leve o ensaio para a placa

A sequência de instruções, as operações de registrador e os estados de periféricos apoiam a investigação no simulador. A latência e as formas de onda de uma montagem física dependem da placa, do clock, da carga e do sensor utilizado.

Preparação e gravação

01 Confirme alvo, binário e montagem. Identifique o modelo gravado no encapsulamento, o endereço de Flash previsto e a revisão dos módulos. Abra Montagem no protoboard. Faça as ligações com a placa desenergizada e confira as referências de GND.

02 Escolha a interface. Em Placa real, siga o procedimento específico da placa e do gravador. Na BlackPill por USART1, use adaptador USB–UART TTL de 3,3 V. TXD do adaptador vai a PA10, RXD a PA9 e GND a GND. Se a placa estiver alimentada pelo USB, deixe o VCC do adaptador desconectado.

03 Grave e verifique. Coloque BOOT0 em 1 e dê RESET para entrar no carregador. Selecione a porta, aguarde a identificação e confira o resultado da verificação da memória. Depois, retorne BOOT0 a 0 e dê RESET para executar o firmware.

Dois pontos de prova, um mesmo estímulo

Para medir a resposta do botão, conecte dois canais do instrumento de bancada a PA0 e PC13, com referência comum. Use a borda de descida de PA0 como disparo. Registre a taxa de amostragem, a alimentação e a diferença até a descida de PC13. Repita as medidas para observar sua dispersão.

> Relate os tempos separadamente
> Anote passos, ticks e segundos simulados na coluna da simulação. Na coluna da placa, identifique o instrumento e o tempo medido. A duração da reprodução na tela descreve a apresentação da captura ao usuário.

Abrir a bancada em Placa real →