GRADUAÇÃO · SISTEMAS EMBARCADOS · UTFPR CURITIBA

Manual dos laboratórios de Sistemas Embarcados

Prof. Eduardo N. dos Santos

Na primeira aula, você vai executar um programa em Assembly e ver o que muda nos registradores do microcontrolador a cada instrução. Abra o exemplo resolvido da aula 1. O projeto já vem com a placa e o código que você vai usar.

Em cada aula

1. **Comece pelos exemplos.** O primeiro traz o código e a solução comentada. O segundo usa o mesmo assunto em outra situação. Execute os dois antes de passar aos exercícios.
2. **Faça os laboratórios A, B e C.** Cada um tem um projeto próprio. Abra-o pelo link do enunciado para trabalhar com a placa, os arquivos e os pinos daquela atividade.
3. **Entregue um relatório por aula.** Na aba Relatório da aula, reúna as três respostas e as medições que o roteiro pede. Exporte o HTML e envie-o com os projetos usados.

> **Primeira execução na BlackPill**
> Com o exemplo aberto, pressione F7 para compilar. Depois, pressione F11 e observe qual instrução será executada em seguida. Na aba **Arquitetura**, acompanhe os registradores enquanto avança pelo programa.

Onde encontrar as instruções

Plano das 20 aulas · Instrumentos da bancada · Relatório e avaliação · Boas práticas · Anexo de análise do código · Diagrama de estados no editor

PLANEJAMENTO DA DISCIPLINA

Aulas 1 a 10 · Assembly e periféricos

Nas cinco primeiras aulas, você escreve programas em Assembly e acompanha registradores, memória e pilha. A partir da aula 6, passa a trabalhar com os pinos da BlackPill e com os periféricos ligados a eles.

Aulas 1 a 10 · Assembly e periféricos

Aula	Assunto	O que procurar na bancada
01	Registradores, aritmética e flags	Qual instrução produziu o valor e alterou Z?
02	Desvios condicionais e laços	Em qual iteração o desvio deixa de ser tomado?
03	Organização e acesso à SRAM	Qual byte muda e quais bytes são preservados?
04	Sub-rotinas, pilha e convenção de chamada	Onde R4 e o endereço de retorno ficam salvos?
05	Operações com bits e barramentos GPIO	Qual sequência de quatro bits aparece em PA3…PA0?
06	GPIO digital e polaridade de entradas e saídas	Quais níveis correspondem ao botão e ao LED ativos?
07	SysTick e interrupções periódicas	Onde o programa registra a passagem pelo handler?
08	USART e protocolos de comando	Qual comando produz cada estado de saída?
09	I²C, aquisição e histerese	Como duas inclinações iguais podem produzir estados distintos?
10	SPI e atualização de displays	Quais pixels mudam a cada entrada no contador de lotação?

Aulas 11 a 20 · Máquinas de estados e FreeRTOS →

PLANEJAMENTO DA DISCIPLINA · AULAS 11 A 20

Aulas 11 a 20 · Máquinas de estados e FreeRTOS

Nas aulas 11 a 15, você programa controladores que respondem a botões, temporizadores e falhas. Na aula 16 começam os projetos com FreeRTOS. Neles, você acompanha quando cada tarefa executa, por que fica bloqueada e como isso afeta o tempo de resposta.

Aulas 11 a 20 · Modelos de comportamento e sistemas operacionais

Aula	Assunto	O que procurar na bancada
11	Estados, eventos e transições	Em que estado o mesmo toque acende a luz? Em que estado inicia o pisca?
12	Temporizadores e guardas de transição	Qual condição decide a transição no instante em que o temporizador vence?
13	Sequências e protocolos de eventos	O que ocorre depois de um evento fora de ordem?
14	Falhas e recuperação	Quando a falha é confirmada? O que permite sair do estado seguro?
15	Máquinas cooperativas	Qual evento confirma que o trabalho solicitado chegou ao fim?
16	Tarefas periódicas e escalonamento	Qual tarefa ganha a CPU quando as duas estão prontas?
17	Filas, buffers circulares e comunicação entre tarefas	A fila preserva a ordem dos pulsos? Em que ponto um buffer cheio impede uma escrita?
18	Sinalização e sincronização por eventos	Qual evento acorda a tarefa? Ela espera semáforo, notificação ou bits de um grupo?
19	Mutexes, exclusão mútua e herança de prioridade	Quando uma tarefa espera pelo mutex, a prioridade de quem o detém muda?
20	Preempção, tempo de resposta e deadlines	Por quanto tempo a sirene foi preemptada e qual foi sua folga ou atraso?

As condições, os projetos e os critérios de cada ensaio estão no plano de aulas da trilha.

OPERAÇÃO DA PLATAFORMA

Ferramentas da bancada

Escolha a ferramenta de acordo com o que precisa observar. Para saber se o botão foi lido, acompanhe o pino de entrada. Se a dúvida for quanto tempo o programa levou para responder, acompanhe também o pino de saída.

Da pergunta ao registro no relatório

O que investigar	Onde e como observar
Registradores e memória	Em **Arquitetura** ou **Memória & pilhas**, observe o valor antes e depois de F11. Se estiver lendo a SRAM, anote o endereço e se a instrução acessa um byte, meia palavra ou uma palavra.
Escrita em um endereço	Em **Watches & breakpoints**, configure um watchpoint de escrita com o endereço e a largura do acesso. Execute o programa. Quando a execução parar, confira qual instrução fez a escrita. Para apenas acompanhar o valor, use um Watch.
Entradas e saídas	Abra **Ver circuito** para acionar a entrada. Em **Analisar sinais**, selecione os pinos e marque as duas bordas com os cursores A e B. As ligações da placa física estão em **Montagem no protoboard**.
Mudança de estado	Abra **Estados** e selecione a transição. O editor mostra a atribuição correspondente. Coloque um breakpoint nessa linha e acompanhe a variável de estado em Watches.
Dados na interface	Para a USART, envie o comando pelo **Terminal serial** e confira a resposta. Para o I²C, abra o decodificador e procure o endereço do dispositivo, o sentido da transferência e os bytes recebidos.
Interrupção	No **NVIC**, confira se a interrupção está habilitada, se há uma solicitação pendente e se alguma máscara impede o atendimento. Um breakpoint na entrada da rotina permite confirmar que ela foi chamada.
Execução de uma tarefa	Em **Threads**, congele a captura e selecione a linha da tarefa. Posicione o cursor no intervalo que quer examinar e leia o estado da tarefa. Use **Exportar relatório** para guardar a captura.

> **Qual unidade de tempo usar?**
> O painel pode mostrar passos do simulador, ticks do kernel ou segundos simulados. Use a unidade indicada na tela. No arquivo VCD, 1 ns por passo é uma convenção de exportação. Para obter um tempo físico, faça a medição na placa.

ENTREGA DOS ENSAIOS

O que entregar

Abra **Relatório da aula** na página da atividade. O formulário tem um espaço para cada laboratório. Preencha os três no mesmo arquivo.

1. Informe o nome dos integrantes, a turma, a placa utilizada e as configurações que você alterou.
2. Antes de executar o projeto, escreva em **Hipótese** o resultado que espera. Em **Procedimento e instrumentos**, diga o que alterou no programa e qual instrumento usou para verificar a mudança.
3. Em **Resultados e unidades**, informe o que mediu. Para uma medida de tempo, identifique os dois eventos que marcam o intervalo. Em **Interpretação**, responda à pergunta do roteiro usando a sua medição.
4. Em **Capturas de evidência**, inclua as imagens pedidas em PNG ou JPEG. Se estiver comparando duas versões, identifique cada uma na legenda.
5. Selecione **Exportar relatório HTML** e reabra o arquivo para conferir as três respostas. Exporte também os projetos usados em `.cortex.json` e envie-os com o relatório pelo canal indicado pelo professor. Inclua CSV, VCD ou relatório de threads nas atividades que pedirem esses arquivos.

Critérios de avaliação

Critério	O que permite verificar o trabalho
Projeto	O projeto entregue abre, compila e permite repetir o teste descrito.
Medições	Os valores têm unidade e a captura mostra o intervalo ou a mudança que a atividade pede.
Análise	A resposta usa o código e as medições para explicar o comportamento do programa.
Conclusão	O relatório identifica as condições testadas e discute diferenças entre a previsão e a medida.

PROJETO DE FIRMWARE

Como organizar o firmware para os testes

Defina o início e o fim da medição

No projeto da sirene, o tempo de resposta começa quando chega o pedido e termina quando a saída é acionada. Use esses mesmos eventos em todas as medições. Anote o prazo e indique se uma resposta exatamente nesse limite é aceita.

Localize a leitura, a decisão e a saída

No inclinômetro, encontre a leitura do sensor, a comparação com os limiares e a escrita no LED. Para testar a histerese, passe pelo mesmo ângulo uma vez na subida e outra na descida. Em cada passagem, anote o estado anterior do alarme.

Filas, notificações e mutexes

Decisão de projeto e ensaio correspondente

Necessidade	Escolha e pergunta de teste
Transportar valores em ordem	**Fila.** Numere as mensagens e reduza a velocidade do consumidor. Confira o que o produtor faz quando a fila enche.
Avisar que há trabalho	**Notificação ou semáforo.** Gere dois avisos antes de a tarefa voltar a executar. Verifique se o programa precisa contar os dois ou se basta saber que há trabalho pendente.
Proteger um recurso compartilhado	**Mutex.** Localize as operações que reservam e liberam a UART, inclusive nos caminhos de erro. Meça quanto tempo cada tarefa retém o recurso. Na aula 19, observe a herança de prioridade enquanto a outra tarefa espera.



Na aula 19, ambas as tarefas precisam obter o mesmo mutex antes de escrever a mensagem inteira.

Tratamento de falhas

Confira o retorno das funções que criam as tarefas, filas e mutexes. No produtor, trate o caso de fila cheia. Nas rotinas de interrupção, use a função `FromISR` correspondente e confira as prioridades permitidas no projeto. Se houver um prazo a verificar na placa, deixe acessíveis os pinos que marcam o pedido e a resposta.

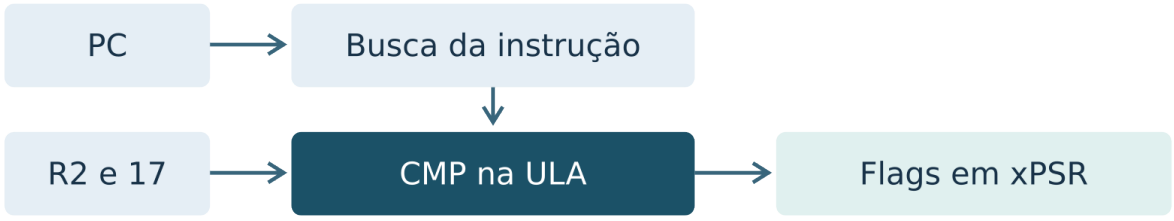
ANEXO · ANÁLISE DO CÓDIGO · 1 DE 4

Assembly · registradores e flags

No código da aula 1, a soma grava o resultado em R2. A última instrução compara esse valor com 17. Execute o trecho passo a passo e acompanhe R2 e o bit Z em xPSR.

```
movs r0, #12
movs r1, #5
adds r2, r0, r1
subs r3, r0, r1
cmp  r2, #17
```

1. Compile com F7 e abra **Código de máquina**. Em **Arquitetura**, acompanhe o PC. Avance com F11 até a instrução `cmp`, antes de executá-la. Nesse ponto, R2 deve conter 17 e R3 deve conter 7.
2. Pressione F11 mais uma vez. A comparação calcula 17 − 17 para atualizar os flags. Confira que Z passou a 1 e que R2 ainda contém 17.
3. Troque a primeira constante de 12 para 9 e recompile. Nos mesmos pontos, confira a nova soma (R2 = 14), a subtração (R3 = 4) e o resultado da comparação (Z = 0).



O PC indica qual instrução buscar. Ao executar `cmp r2, #17`, a CPU lê R2 e atualiza os flags em xPSR.

Quando o código estiver em C

No **Fluxograma**, localize a decisão que quer acompanhar. Marque um breakpoint na linha correspondente. Uma linha de C pode gerar várias instruções. Se precisar saber qual delas escreveu na memória ou no periférico, abra **Código de máquina** e avance pelo endereço da instrução.

ANEXO · ANÁLISE DO CÓDIGO · 2 DE 4

GPIO e interrupções na BlackPill

Na aula 6, o programa consulta o botão em PA0. Na aula 18, uma tarefa dispara a interrupção por software. Use o procedimento correspondente ao projeto que abriu.

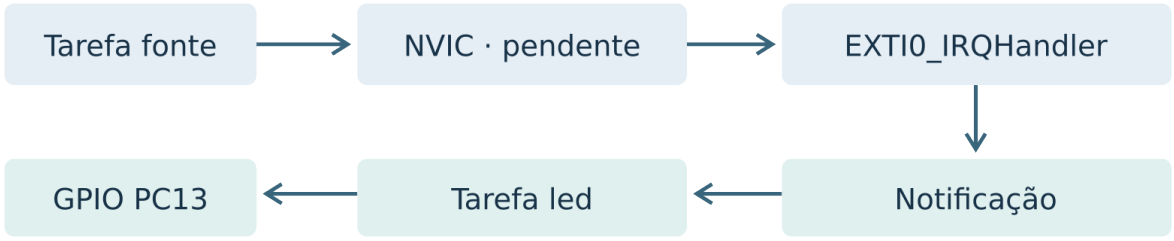
Aula 6 · botão e LED

Confira as ligações em **Chip e conexões**. Em **Analisador lógico**, selecione PA0 no canal 1 e PC13 no canal 2. Execute o projeto, pressione o botão, mantenha-o pressionado e solte. Congele a captura com os dois sinais visíveis desde o repouso.

No projeto-base, PA0 cai para zero quando o botão é pressionado e PC13 cai para zero quando o LED acende. Marque A na primeira borda e B na segunda. Leia o intervalo em passos. Se as bordas aparecerem sobrepostas, amplie a captura ou use os instantes das duas transições no CSV.

Aula 18 · interrupção e notificação da tarefa

A tarefa `fonte` coloca `EXTI0_IRQn` em estado pendente. A rotina de interrupção chama `vTaskNotifyGiveFromISR` e libera a tarefa `led`. Quando o escalonador a seleciona, ela executa a escrita em PC13.



O pedido parte da tarefa `fonte`. A última etapa é a escrita da tarefa `led` em PC13.

Primeiro, coloque um breakpoint na entrada da rotina para confirmar que ela executou. Retire o breakpoint antes de medir. Em **Threads**, localize o início da execução de `led`. No analisador lógico, localize a mudança em PC13. O intervalo entre esses eventos inclui as instruções que a tarefa executa antes de escrever no pino.

ANEXO · ANÁLISE DO CÓDIGO · 3 DE 4

Máquinas de estados · lanterna e alarme de porta

Na lanterna da aula 11, o primeiro toque acende o LED e o segundo inicia o pisca. O botão é o mesmo nas duas vezes. O que muda é o estado da lanterna antes de cada toque.

Lanterna de três modos

Abra o projeto da lanterna, deixe KEY solto e selecione PA0 e PC13 no analisador. Pressione KEY por mais de 20 ms para passar pelo filtro do botão. Solte, aguarde o reconhecimento da soltura e pressione outra vez. Congele a captura durante o pisca, antes do terceiro toque.

Previsão para a lanterna após o filtro da entrada

Estado anterior	Evento	Estado seguinte e leitura esperada
APAGADA	Pressão confirmada	FIXA. PC13 passa a zero e o LED acende.
FIXA	Pressão confirmada	PISCA. A primeira fase é acesa e cada fase dura 250 ticks de 1 ms.
PISCA	Pressão confirmada	APAGADA. PC13 volta a um.

Em PISCA, meça de uma borda de descida de PC13 à próxima borda de descida. As duas fases de 250 ticks completam um período de 500 ms. Se o analisador exibir passos, anote esse valor em passos e use o contador do firmware para conferir o período em milissegundos.

Alarme de porta

Abra o projeto do alarme. Pressione KEY para abrir a porta e solte após cerca de 1 s no relógio do firmware. Depois, mantenha-a aberta por mais de 2 s. O temporizador começa na abertura confirmada. Se a porta fechar durante a espera, o firmware cancela o alarme. Se o fechamento e o vencimento forem reconhecidos na mesma atualização, o fechamento tem prioridade.

Na tabela do relatório, inclua o estado inicial, o evento, o instante e a saída prevista e observada. Registre um evento ignorado e um teste que ultrapasse o limite de tempo.

BANCADA · DIAGRAMA DE ESTADOS

Como usar a aba Estados

O diagrama da lanterna mostra em que parte do código um toque muda a luz fixa para o modo intermitente. Você pode abrir essa linha diretamente pelo diagrama.

Localize a transição de FIXA para PISCA

1. Abra o projeto da lanterna e selecione **Estados**. Se houver mais de um controlador, use **Máquina** para escolher a variável e o arquivo.
2. Selecione a seta de `FIXA` para `PISCA`. Marque um breakpoint na atribuição indicada, compile e execute. Pressione e solte KEY até chegar à linha. Avance uma instrução ou linha de cada vez até a atribuição terminar e confira `mem32(estado)` em Watches.
3. Troque o destino dessa atribuição para `APAGADA`. Volte a **Estados** e confira a nova seta. Recompile e repita os dois toques. Observe o que acontece em PC13 no segundo toque.

Outros projetos da disciplina

Aula	Experimento	Evidência no relatório
12 · Guardas	Pressione KEY antes e depois de o flash terminar de recarregar.	Anote o instante do toque e o estado seguinte em cada tentativa.
13 · Sequências	Envie `!1;` pela USART2. Partindo do mesmo estado, repita com os caracteres fora de ordem.	Indique o primeiro caractere que levou a uma transição diferente.
14 · Falhas	Provoque a falha, retire o estímulo e tente rearmar o sistema.	Identifique a condição para detectar a falha e a condição para voltar a operar.
15 · Cooperação	Na catraca, siga o pedido no primeiro controlador e a confirmação no segundo.	Registre os estados dos dois controladores e o evento de confirmação.

> **Quando faltar uma transição no diagrama**
> O editor reconhece estados e ramos explícitos no código C. Se o próximo estado for calculado durante a execução ou definido em uma macro, confira a cobertura indicada no painel e acompanhe a variável com um Watch.

ANEXO · ANÁLISE DO CÓDIGO · 4 DE 4

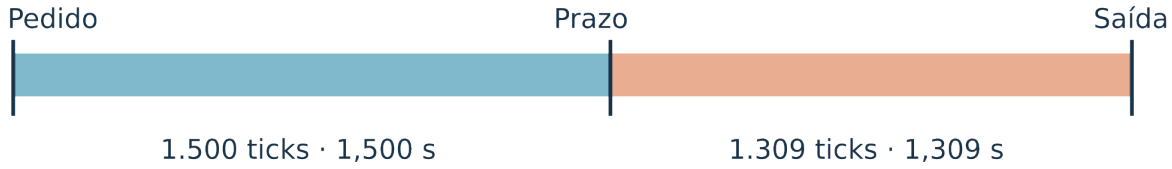
FreeRTOS · tempo de resposta da sirene

Na aula 20, a subida de PB0 marca o pedido e a subida de PB1 marca o acionamento da sirene. Localize `PRIO_PAINEL`, `PRIO_SIRENE` e `PRAZO_ALARME_MS` no projeto. Abra **Threads** para ver quando a sirene ganhou a CPU entre essas duas bordas.

Uma ativação observada por configuração no motor da BlackPill

Prioridade do painel	Pedido	Início da sirene	Resposta	Resultado
3	3 ticks	2.412 ticks	2.812 ticks	1.309 ticks de atraso
1	4 ticks	11 ticks	411 ticks	1.093 ticks de folga

Os dados da tabela vêm de duas execuções do simulador. A sirene tem prioridade 2, o tick vale 1 ms e o prazo é de 1.500 ms. Em cada linha, use o instante do pedido daquela execução.



O gráfico começa no pedido. A parte após o prazo de 1.500 ms corresponde ao atraso da primeira execução.

Com o painel em prioridade 3, passaram 2.812 − 3 = **2.809 ms** entre pedido e resposta. A sirene excedeu o prazo em 1.309 ms. Com o painel em prioridade 1, o tempo caiu para 411 − 4 = **407 ms**, com 1.093 ms de folga.

Em **Threads**, congele a captura e selecione **Medir tempo · A/B**. Compare as bordas de PB0 e PB1 com os contadores da UART para o mesmo pedido. Os contadores avançam de um em um tick. Os cursores podem indicar um valor entre dois ticks.

O exemplo resolvido traz outra captura, com 2.801 ticks na UART para uma previsão de 2.800. Use os valores da sua execução. Para verificar um prazo na placa física, meça os mesmos sinais na placa.

Para testar uma carga maior, abra Controle de motor no limite ou Concentrador de telemetria no limite. Em uma janela com vários períodos, anote a porcentagem de tempo em IDLE e a menor folga entre o fim de uma ativação e seu prazo. Altere a carga e compare.

Fontes para consulta · STM32F411CE e documentação da ST · Estados de tarefas no FreeRTOS · Mutexes no FreeRTOS.