Briefing Detalhado: Arquiteturas para Sistemas Embebidos com ESP32-C3

Este documento serve como um guia abrangente sobre os principais temas, conceitos e factos apresentados nas fontes fornecidas para o curso "Arquiteturas para Sistemas Embebidos", com foco no ESP32-C3. Abrange desde a configuração do ambiente de desenvolvimento, programação de periféricos, técnicas de transferência de dados, até sistemas operativos em tempo real e gestão de energia.

1. Introdução ao ESP32-C3 e Ambiente de Desenvolvimento

O ESP32-C3 é um "System-on-a-Chip (SoC) da Espressif Systems com um núcleo RISC-V de 32 bits, conectividade Wi-Fi e Bluetooth LE, e diversos periféricos." ("Guia Essencial ESP32-C3"). O curso utiliza o kit **ESP32-C3-DevKitM-1** e o módulo **ESP32-C3-MINI-1** para as atividades laboratoriais.

1.1. Toolchain e Fluxo de Compilação

O desenvolvimento para o ESP32-C3 é suportado pelo **ESP-IDF (Espressif IoT Development Framework)**. O fluxo de compilação C/C++ envolve várias etapas e ferramentas:

- **CMake**: "uma ferramenta de meta-construção que gera arquivos de construção (makefiles ou projetos IDE) a partir de descrições de alto nível." ("Guia Essencial ESP32-C3"). É utilizado para configurar o projeto.
- **Ninja**: "um sistema de construção que se concentra na velocidade, executando comandos de construção paralelos de forma muito eficiente com base nos arquivos gerados pelo CMake." ("Guia Essencial ESP32-C3"). Executa a compilação de forma otimizada.
- **ESP-IDF (command line tools)**: Inclui comandos como idf.py set-target esp32c3, idf.py menuconfig (para configurar o projeto), idf.py build, e idf.py -p PORT flash para gravar o firmware no dispositivo.
- O ambiente de desenvolvimento pode ser baseado em linha de comando ou no **Visual Studio Code** com a extensão ESP-IDF.

1.2. Variáveis em C/C++

A gestão de variáveis em C/C++ é fundamental na programação de sistemas embebidos:

- **Tipos**: Primitivos, Derivados e Definidos pelo Utilizador.
- **Visibilidade (Scope)**: A região do programa onde uma variável é acessível.
- **Lifetime**: O período durante o qual uma variável existe na memória.
- **Alocação**:
- **Locais (automáticas vs. estáticas)**: Variáveis automatic (locais padrão) são "alocadas na stack, criadas quando a função é chamada e destruídas quando a função termina, tendo um tempo de vida limitado à execução da função." Em contraste, variáveis static são "alocadas no segmento de dados ou BSS, são

inicializadas apenas uma vez e persistem durante toda a execução do programa, mantendo seu valor entre chamadas de função." ("Guia Essencial ESP32-C3").

- **Globais**: Podem ser acessadas por qualquer parte do programa. A palavra-chave extern é usada para declarar que uma variável ou função é definida noutro ficheiro-fonte.
- **Heap Alocadas (Dinâmicas)**: Memória alocada em tempo de execução usando malloc e libertada com free.

2. Periféricos Essenciais do ESP32-C3

2.1. GPIO (General Purpose Input/Output)

Os pinos GPIO são "pinos digitais num microcontrolador que podem ser configurados como entradas ou saídas de propósito geral." ("Guia Essencial ESP32-C3"). As tarefas de laboratório iniciais focam-se na manipulação de GPIOs para controlar LEDs (blink) e ler/escrever valores.

2.2. Timers e Watchdog Timers

- **Timers**: Periféricos essenciais para gerar eventos periódicos e controlo de tempo. O ESP32-C3 possui um **System Timer** (52-bit counters e comparators) e **Timer Group** (general-purpose timers com prescalers de 16 bits e time-base counters de 54 bits).
- **Controlo de Frequência**: Pode ser feito com reset assíncrono (fout = fin / k) ou síncrono (fout = fin / (k+1)), onde k é o valor de comparação.
- **Watchdog Timers (WDTs)**: "Um temporizador eletrónico que é usado para detetar e recuperar de falhas de software, como loops infinitos, reiniciando o sistema se não for "alimentado" regularmente." ("Guia Essencial ESP32-C3"). O ESP32 possui múltiplos estágios configuráveis para MWDT e RWDT, com ações como interrupção, reset do CPU ou reset do sistema.

2.3. PWM Generators (LED Controllers)

- **PWM (Pulse Width Modulation)**: "Uma técnica para controlar a potência média entregue a um dispositivo eletrónico através da variação da largura dos pulsos de um sinal digital." ("Guia Essencial ESP32-C3").
- **Duty Cycle**: "a proporção do tempo em que o sinal está em um estado "ligado" (alto) em relação ao período total do sinal." ("Guia Essencial ESP32-C3"). Variar o *duty cycle* permite ajustar o brilho de um LED.
- O ESP32-C3 possui seis geradores PWM independentes (canais) e quatro timers, com resolução máxima de 14 bits, suportando *duty cycle fading*.

2.4. ADC (Analog-to-Digital Converter) e DAC (Digital-to-Analog Converter)

- **ADC**: "Um dispositivo eletrónico que converte um sinal analógico (contínuo) em um sinal digital (discreto)." ("Guia Essencial ESP32-C3").

- **Sampling (Amostragem)**: "o processo de capturar o valor de um sinal analógico em intervalos de tempo discretos." ("Guia Essencial ESP32-C3").
- **Quantization (Quantização)**: "o processo de aproximar os valores das amostras contínuas para um conjunto finito de valores discretos predefinidos, convertendo-os em representações digitais com base na resolução de bits do ADC." ("Guia Essencial ESP32-C3").
- **Nyquist Frequency**: "A metade da taxa de amostragem de um sistema de sinal discreto. A amostragem deve ser feita a pelo menos o dobro da frequência mais alta presente no sinal analógico para evitar aliasing." ("Guia Essencial ESP32-C3").
- O ESP32-C3 suporta ADCs SAR (Successive Approximation Register), com modos "one-shot" e "continuous".
- **DAC**: "Um dispositivo eletrónico que converte um sinal digital (discreto) em um sinal analógico (contínuo)." ("Guia Essencial ESP32-C3"). O ESP32-C3 não possui DACs integrados, mas funcionalidades semelhantes podem ser implementadas usando GPIOs (com Delta-Sigma Modulator) ou PWM.

3. Técnicas de Transferência de Informação

A transferência eficiente de dados é crucial em sistemas embebidos.

- **Polling**: "o CPU verifica repetidamente o estado de um dispositivo, o que é ineficiente pois consome ciclos de CPU mesmo quando não há dados prontos." ("Guia Essencial ESP32-C3"). Aceitável para tarefas simples e de baixa frequência.
- **Interrupts**: "o dispositivo notifica o CPU apenas quando precisa de atenção, tornando-o mais eficiente em termos de uso do CPU." ("Guia Essencial ESP32-C3"). Essencial para sistemas responsivos e de alta taxa de dados.
- **DMA (Direct Memory Access)**: "Uma funcionalidade que permite que certos periféricos num sistema de computador acedam à memória principal independentemente do CPU, melhorando a eficiência na transferência de dados." ("Guia Essencial ESP32-C3"). Ideal para cópia de grandes buffers ou transferência de dados intensiva sem sobrecarregar o CPU.

4. Interfaces Seriais

As interfaces seriais são fundamentais para a comunicação com periféricos externos e outros sistemas.

- **RS232 (UART)**: Um padrão de comunicação serial assíncrona, amplamente utilizado para comunicação entre computadores e periféricos. Requer três sinais básicos (Tx, Rx, GND).
- **SPI (Serial Peripheral Interface)**: "uma interface de comunicação serial síncrona full-duplex de alta velocidade, geralmente usando quatro fios (MISO, MOSI, SCLK, SS)." É "sempre Master/Slave, com um único Master controlando vários Slaves." ("Guia Essencial ESP32-C3").

- **I2C (Inter-Integrated Circuit)**: "uma interface de comunicação serial síncrona de dois fios (SDA, SCL), popular para comunicação entre componentes em curtas distâncias." É "half-duplex e suporta múltiplos Masters e Slaves na mesma bus, exigindo endereçamento de dispositivos." ("Guia Essencial ESP32-C3").

O desenvolvimento de **Device Drivers** é essencial para interagir com sensores como o TC74 (I2C) e o BME280 (I2C/SPI), abstraindo a complexidade da comunicação do hardware.

5. Real-time Operating Systems (RTOS)

Um **RTOS (Real-time Operating System)** é "um sistema operativo concebido para executar aplicações com restrições de tempo, garantindo que as operações sejam concluídas dentro de prazos definidos." ("Guia Essencial ESP32-C3"). Diferencia-se de um sistema de propósito geral pela sua capacidade de garantir o cumprimento de *deadlines*.

5.1. Conceitos Fundamentais

- **Real-time Systems**: Podem ser "hard" ou "soft". Em sistemas "hard", "a falha em cumprir um prazo (deadline) é considerada uma falha crítica do sistema, com consequências potencialmente catastróficas." Em sistemas "soft", "a falha ocasional em cumprir um prazo é tolerável e resulta apenas numa degradação da performance ou qualidade, sem causar falhas completas do sistema." ("Guia Essencial ESP32-C3").
- **Tasks (Tarefas)**: "As unidades de execução independentes e concorrentes dentro de um sistema operativo em tempo real, geridas pelo agendador." ("Guia Essencial ESP32-C3").
- **Scheduler (Agendador)**: Componente do RTOS que decide qual tarefa deve ser executada a qualquer momento, com base em prioridades e estados das tarefas.
- **Shared Variables (Variáveis Partilhadas)**: Variáveis acessadas por múltiplas tarefas, exigindo mecanismos de sincronização.
- **Critical Sections (Secções Críticas)**: Partes do código que acessam recursos partilhados e devem ser executadas atomicamente.
- **Mutexes (Mutual Exclusion)**: "Um mecanismo de sincronização usado em sistemas multitarefas para garantir que apenas uma tarefa possa aceder a um recurso partilhado por vez, prevenindo condições de corrida." ("Guia Essencial ESP32-C3").
- **Semaphores**: "Mecanismos de sincronização em sistemas multitarefas usados para controlar o acesso a recursos partilhados ou sinalizar eventos entre tarefas." ("Guia Essencial ESP32-C3").
- **Queues (Filas)**: "Uma estrutura de dados FIFO (First-In, First-Out) usada em RTOSs para comunicação entre tarefas, permitindo o envio e receção de mensagens." ("Guia Essencial ESP32-C3").

5.2. FreeRTOS no ESP32

O **FreeRTOS** é um RTOS popular e de código aberto utilizado no ESP32-C3. Funções essenciais incluem:

- xTaskCreate(...): Para criar novas tarefas.
- vTaskDelay(...), xTaskDelayUntil(...): Para atrasar a execução de tarefas.
- xTimerCreate(...), xTimerStart(...): Para criar e gerir timers.
- Funções para Queues, Semaphores e Mutexes (xQueueCreate, xSemaphoreTake, xSemaphoreGive, etc.).

A utilização de um RTOS permite uma modularização e gestão de concorrência mais eficazes em comparação com sistemas "bare-metal".

6. Gestão de Energia

A gestão de energia é "crítica em modern computing e embedded systems" devido à necessidade de prolongar a vida útil da bateria e reduzir o calor. O consumo de energia é afetado pela tensão de alimentação e frequência de operação.

6.1. Modos de Baixo Consumo do ESP32-C3

O ESP32-C3 oferece vários modos de baixo consumo:

- **Light Sleep Mode**: "desliga o CPU digital e a maior parte da RAM e periféricos digitais, mas mantém a RAM de baixa potência e o RTC ativo, permitindo um rápido despertar por eventos externos ou timers RTC. É útil para aplicações que precisam de respostas rápidas mas podem dormir entre eventos." ("Guia Essencial ESP32-C3").
- **Deep Sleep Mode**: "desliga quase todo o chip, incluindo a RAM, mantendo apenas o RTC e parte da RAM de baixa potência, resultando no menor consumo de energia. É ideal para aplicações com longos intervalos entre atividades, como sensores remotos que transmitem dados poucas vezes ao dia." ("Guia Essencial ESP32-C3").
- Outros modos incluem o Active Mode e o Hibernate Mode (não detalhado nas fontes).

O despertar dos modos de suspensão pode ser por timers RTC, GPIOs externos, etc. A exploração do modo *standby* de sensores como o TC74 em conjunto com os modos de sono do ESP32-C3 permite otimizar ainda mais o consumo.

7. Conectividade Wireless: WiFi

O ESP32-C3 possui capacidades de **WiFi** e **Bluetooth Low Energy (BLE)**.

7.1. Conceitos de WiFi

- **Redes WiFi**: Organizadas em torno de Access Points (APs) ou operando em modo ad-hoc (como ESP-NOW).
- **Bandas de Frequência**: O WiFi utiliza bandas ISM (Industrial, Scientific, and Medical).

- **Protocolos/Serviços**: O ESP-IDF fornece APIs para WiFi (station, softAP, espnow).
- **Restrições na Coexistência (WiFi e BLE)**: "O ESP32-C3 possui um único rádio para WiFi e BLE, o que significa que eles compartilham o mesmo hardware. Embora o chip suporte coexistência, não podem operar em modo contínuo e simultâneo com alta performance. Pode haver degradação de desempenho ou latência em cenários de uso intensivo de ambos, e o programador precisa gerenciar cuidadosamente o tempo de ar e a prioridade entre os dois protocolos." ("Guia Essencial ESP32-C3").

7.2. Modos WiFi do ESP32-C3

O ESP32-C3 pode operar como:

- **Station**: Conecta-se a um Access Point existente.
- **SoftAP (Access Point)**: Cria a sua própria rede WiFi para outros dispositivos se conectarem.
- **ESP-NOW**: Um protocolo de comunicação sem conexão rápido para pequenas quantidades de dados, útil para comunicação direta entre ESPs sem um router.

Este briefing fornece uma visão consolidada dos tópicos abordados nas fontes, destacando as ideias mais importantes e as suas aplicações práticas no contexto do desenvolvimento de sistemas embebidos com o ESP32-C3.

ASE_2024-25_Slides-T-06

1. What distinguishes serial and parallel interfaces in a computer system?

As interfaces paralelas transmitem múltiplos bits em simultâneo, utilizando várias linhas de dados (uma por bit), enquanto as interfaces seriais transmitem os bits um a um, em sequência, usando apenas uma ou poucas linhas. As interfaces seriais são mais simples, económicas e fiáveis para comunicações a longa distância.

2. What are the advantages of serial interfaces?

- Menor número de fios/pinos, o que reduz a complexidade do hardware.
- Menor custo de implementação.
- Melhor desempenho em longas distâncias, com menos interferência eletromagnética.
- Menor consumo de energia.
- Mais adequadas para ligações entre microcontroladores e sensores em sistemas embebidos.

3. What is the purpose / preferred use?

- **RS232 (UART):** Comunicação simples ponto-a-ponto, ideal para debug, terminais e ligação de dispositivos periféricos em curtas distâncias.
- **SPI (Serial Peripheral Interface):** Comunicação rápida com periféricos como memórias flash, sensores e ecrãs; adequada para altas taxas de transferência.
- **I2C (Inter-Integrated Circuit):** Ligação de vários dispositivos de baixa velocidade no mesmo barramento; ideal para sensores, relógios RTC e EEPROMs.

4. What signals make up the interface?

- **RS232 (UART):** Tx (transmissão), Rx (recepção), GND (terra).
- **SPI:** MISO (Master In Slave Out), MOSI (Master Out Slave In), SCLK (clock), SS (Slave Select).
- **I2C:** SDA (dados), SCL (clock).

5. How is it characterized regarding communication (bi)directionality (simplex, half-duplex, full-duplex)?

- **RS232 (UART):** Full-duplex – permite envio e recepção simultânea.
- **SPI:** Full-duplex – transmissão e recepção ocorrem em paralelo.
- **I2C:** Half-duplex – comunicação bidirecional alternada (não simultânea).

6. Is there symmetry between the communication ends, or is it a Master/Slave configuration?

- **RS232 (UART):** Simétrica – ambos os lados são equivalentes.

- **SPI:** Master/Slave – um Master controla os Slaves.
- **I2C:** Master/Slave – permite múltiplos Slaves e múltiplos Masters.

7. What is the (typical) maximum transmission rate (bits per second)?

- **RS232 (UART):** Tipicamente até 115200 bps; possível até 1 Mbps em alguns casos.
- **SPI:** Pode atingir entre 10 a 50 Mbps, dependendo do hardware.
- **I2C:** 100 kbps (Standard Mode), 400 kbps (Fast Mode), até 3.4 Mbps (High-Speed Mode).

8. What voltage levels are used (for each logic level)?

- **RS232 (UART):** ±12 V (padrão RS232); 0 V e 3.3/5 V em versões com lógica TTL.
- **SPI:** 0 V (LOW) e 3.3 V ou 5 V (HIGH), conforme o sistema.
- **I2C:** 0 V (LOW) e 3.3 V ou 5 V (HIGH), com resistências pull-up nas linhas.

9. How is a data frame organized? How are the bits arranged in a transmission?

- **RS232 (UART):** 1 bit de start, 5–9 bits de dados, paridade opcional, 1 ou mais bits de stop.
- **SPI:** Blocos de 8 bits (ou múltiplos), com envio e recepção sincronizados pelo clock.
- **I2C:** Start bit, endereço (7 ou 10 bits), bit de leitura/escrita, ACK/NACK, dados, Stop bit.

10. How is synchronization achieved in data transfer? What type of clock is used?

- **RS232 (UART):** Assíncrona – não utiliza clock partilhado; sincronização por bit de start.
- **SPI:** Síncrona – o clock é gerado pelo Master e enviado por SCLK.
- **I2C:** Síncrona – o Master fornece o clock na linha SCL.

11. What configuration parameters need to be programmed?

- **RS232 (UART):** Baud rate, bits de dados, paridade, bits de stop.
- **SPI:** Frequência do clock, polaridade e fase do clock (CPOL/CPHA), tamanho do frame (número de bits), dispositivo selecionado (SS).
- **I2C:** Endereço dos dispositivos, velocidade do clock, configuração de resistências pull-up.

12. How does the BME280 sensor switch between I2C and SPI modes?

O BME280 muda de modo de comunicação com base no estado do pino **CSB (Chip Select Bar)**:

- **CSB ligado a VDDIO:** o sensor opera em modo **I2C**.

- **CSB ligado a GND:** o sensor opera em modo **SPI**.

O modo é fixado na inicialização – **não pode ser comutado dinamicamente** sem reiniciar o sensor.

13. Which components does the breakout board add "around" the BME280 sensor?

As breakout boards do BME280 normalmente incluem:

- **Regulador de tensão** (para aceitar VCC de 5 V e gerar 3.3 V para o sensor).
- **Resistências pull-up** nas linhas I2C (SDA, SCL).
- **Conversores de nível lógico** (em algumas versões) para compatibilidade com microcontroladores de 5 V.
- **Pinagem acessível** para SPI e I2C, com identificação clara.

14. What is the supply voltage and pin of the BME280 breakout board?

O pino de alimentação é geralmente rotulado como **VCC** (ou VIN).

- A alimentação típica aceita **3.3 V a 5 V**, dependendo da placa.
- Internamente, o sensor opera a **3.3 V**, sendo regulado se necessário.

15. How to manage efficiently, in the breadboard, the possibility of using the I2C and SPI modes (one at a time)?

Sugere-se:

- Ligar o pino **CSB** a uma **ligação comutável (ex: jumper ou switch)**.
- **Evitar ligar SPI e I2C ao mesmo tempo**. Usar dois conjuntos de fios e ativar apenas um modo por vez.
- Ter o firmware preparado para detetar ou configurar corretamente o modo pretendido.
- Opcionalmente, usar dois microcontroladores ou dois projetos separados para testes limpos.

16. How to visualize the 4 SPI signals with a 2-channel oscilloscope?

Soluções práticas:

- Medir **dois sinais de cada vez** (ex: SCLK e MOSI, depois SCLK e MISO).
- **Sincronizar o trigger** com SCLK para analisar a correspondência dos dados.
- Utilizar **uma sonda lógica (logic analyzer)**, se disponível, para monitorizar todos os sinais SPI em simultâneo (ideal).
- Documentar a sequência das capturas para reconstrução temporal.

17. How should a good schematic diagram be organized?

- **Clareza visual**: blocos funcionais separados (sensor, MCU, alimentação).
- **Etiquetas consistentes** em sinais e pinos.
- Uso de **nomes simbólicos** e não apenas nomes físicos (ex: "SCL" em vez de "GPIO6").
- **Identificação das ligações de alimentação**, resistências e modos de operação.
- Inserção de **notas explicativas**, especialmente em pontos críticos (ex: configuração do CSB).
- Conformidade com boas práticas de **engenharia eletrónica** (e.g., GND comum, filtragem, desacoplamento).

18. What capabilities does the device driver need to expose to the application?

- Inicialização e configuração do sensor.
- Leitura de dados ambientais (temperatura, humidade, pressão).
- Gestão de modos de operação (sleep, forced, normal).
- Funções para conversão e compensação dos dados brutos.
- Possivelmente, função de calibração automática ou manual.

19. Who should manage the calibration of the values? Is it the user of the device driver or the device driver itself (internally)?

A calibração deve ser gerida **internamente pelo device driver**, utilizando os **coeficientes de calibração** lidos da EEPROM do sensor, logo após a inicialização.

O utilizador **não deve** ter de lidar com os detalhes da calibração.

20. When should calibration data be obtained from the sensor?

Na **inicialização do driver** – ao configurar o sensor, o driver deve ler os **registos de calibração** e armazenar internamente os coeficientes necessários para processar os dados brutos.

21. What needs to be public in the device driver, and what should be private?

Público (visível no .h):

- Estruturas de dados principais.
- Protótipos de funções como bme280_init(), bme280_read_data().

Privado (oculto, apenas no .c):

- Variáveis internas de estado.
- Coeficientes de calibração.
- Funções auxiliares e rotinas de comunicação.

22. What should be the contents of the ".h" and ".c" of the device driver source files?

.h:

- Includes necessários.
- Definições de tipos, structs, constantes.
- Protótipos das funções públicas.
- Documentação básica de uso.

.c:

- Implementação das funções.
- Leitura e escrita nos registos do sensor.
- Gestão de calibração e conversões.
- Acesso ao bus I2C ou SPI.

23. Where should the public ".h" source file of the device driver be included?

Deve ser incluído no **ficheiro de aplicação** que utiliza o sensor, através de:

#include "bme280_driver.h"

Idealmente, o .h estará numa pasta include/ acessível no CMakeLists.txt ou Makefile.

24. Which user-defined data types could be useful?

- struct bme280_data para agrupar temperatura, pressão e humidade.
- enum bme280_mode para selecionar o modo de operação (ex: FORCED, NORMAL).
- struct bme280_config para definir parâmetros de oversampling, filtros, etc.

25. How can the power consumption of the sensor be decreased? And for the whole system (microcontroller + sensor)?

Sensor (BME280):

- Utilizar modo **sleep** sempre que não for necessária medição contínua.
- Operar no modo **forced**, fazendo medições apenas quando necessário.
- Reduzir o oversampling e filtros internos, se a precisão permitir.

Sistema (MCU + sensor):

- Colocar o microcontrolador em **light sleep ou deep sleep** entre eventos.
- Sincronizar medições com **eventos externos ou timers RTC**.
- Reduzir a frequência de amostragem e transmissão.
- Usar **GPIOs com pull-up/down configurados** para evitar consumo flutuante.

ASE_2024-25_Slides-T-08

1. What is an ADC?

Um ADC (Conversor Analógico-para-Digital) é um dispositivo eletrónico que converte sinais analógicos (contínuos) num formato digital (discreto), permitindo que sistemas digitais interpretem grandezas físicas como tensão, temperatura ou luz.

2. What is a DAC?

Um DAC (Conversor Digital-para-Analógico) converte sinais digitais (discretos) em sinais analógicos (contínuos), permitindo que sistemas digitais actuem sobre dispositivos analógicos.

3. What is a continuous signal?

É um sinal que varia de forma contínua no tempo e pode assumir um número infinito de valores dentro de um intervalo. Exemplo: tensão analógica variável.

4. What is a discrete signal?

É um sinal representado por valores em instantes de tempo discretos, ou seja, amostrados. Resulta do processo de conversão de sinais contínuos via amostragem.

5. What is the sampling process in analog-to-digital conversion?

A amostragem (sampling) consiste em capturar o valor de um sinal analógico em intervalos regulares no tempo. O resultado é uma sequência de valores discretos que representam o sinal original.

6. What is the quantization process in analog-to-digital conversion?

A quantização converte os valores contínuos amostrados num conjunto finito de valores digitais, de acordo com a resolução do ADC. Introduz erro (erro de quantização), uma vez que aproxima valores reais por representações digitais.

7. What is the reconstruction process in digital-to-analog conversion?

A reconstrução converte a sequência de dados digitais num sinal analógico contínuo. Pode ser feita por filtragem (ex: filtros passa-baixo) após a saída de um DAC para suavizar o sinal escalonado.

8. What is the sampling rate of ADCs/DACs?

É a frequência a que são feitas as amostragens (ADC) ou as actualizações de saída (DAC), expressa em amostras por segundo (Hz). Exemplo: 1 kHz = 1000 amostras por segundo.

9. What is the bit resolution of ADCs/DACs?

É o número de bits usados para representar cada amostra. Uma resolução de 10 bits, por exemplo, permite representar 2^{10} = 1024 níveis distintos.

10. What is the Nyquist frequency? What is aliasing?

A frequência de Nyquist é metade da taxa de amostragem. Para evitar aliasing (sobreposição de frequências), a taxa de amostragem deve ser, no mínimo, o dobro da frequência mais alta do sinal. Aliasing ocorre quando essa condição não é cumprida, gerando distorção.

11. What is the reference voltage of ADCs and DACs?

É a tensão máxima que o ADC ou DAC considera como valor de topo. Define o intervalo de entrada/saída analógica. Ex: se Vref = 3.3 V e o ADC tem 10 bits, o valor máximo digital representa 3.3 V.

12. How does a Successive Approximation Register (SAR) ADC work?

O ADC SAR utiliza um processo iterativo para converter sinais analógicos. Um comparador compara o sinal de entrada com uma tensão gerada internamente por um DAC binário, ajustando bit a bit desde o MSB até ao LSB até atingir a aproximação mais próxima. É eficiente e de baixa latência.

ESP32 Aspects

13. How many ADCs are implemented in the ESP32-C3?

O ESP32-C3 implementa **1 ADC SAR**.

14. How many ADC input channels does the ESP32-C3 provide?

Fornece **7 canais de entrada analógica** (ADC1_CHANNEL_0 a ADC1_CHANNEL_6).

15. What distinguishes the "one-shot" and "continuous" modes of the ESP32 ADCs?

- **One-shot:** uma única amostra é capturada por pedido.
- **Continuous:** o ADC amostra continuamente com uma frequência definida, permitindo fluxos contínuos de dados.

16. How is the output of the ESP32 ADCs encoded? Natural binary? Two's complement? Other?

A codificação é feita em **binário natural** (natural binary), sem sinal.

17. How to determine the voltage applied to the ADC from the digital sample value?

Utiliza-se a fórmula:

$$V_{in} = \frac{\text{valor digital}}{2^n - 1} \times V_{ref}$$

Onde *n* é a resolução em bits e *Vref* é a tensão de referência.

18. What is the importance of the calibration process in the ADC?

A calibração corrige desvios e não linearidades, melhorando a precisão das leituras. No ESP32-C3, existem mecanismos para compensar variações de fabrico, temperatura e tensão.

19. The ESP32-C3 does not provide DACs. How can a similar functionality be implemented using GPIOs (with a Delta-Sigma Modulator) or PWM?

- **PWM:** Modula a largura dos pulsos para controlar a tensão média. Um filtro passa-baixo pode suavizar o sinal, convertendo-o em analógico.
- **Delta-Sigma Modulator:** Técnica digital que altera rapidamente o estado de um GPIO com base num algoritmo de oversampling e quantização para gerar uma média analógica precisa.

ASE_2024-25_Slides-T-09

1. What are real-time systems? Are "real-time" and "fast" synonymous?

Sistemas em tempo real são sistemas cujo comportamento correto depende não só da lógica dos resultados, mas também do **tempo em que esses resultados são produzidos**. **"Tempo real" não significa "rápido"** – um sistema pode ser rápido mas não garantir que as respostas ocorram dentro de prazos específicos (deadlines).

2. What is a deadline, and a deadline miss, in real-time systems?

Uma **deadline** é o instante de tempo até ao qual uma determinada tarefa deve ser concluída. Uma **deadline miss** ocorre quando a tarefa não termina dentro desse prazo, podendo comprometer a fiabilidade do sistema.

3. What is the difference between hard and soft real-time systems?

- **Hard real-time:** falhar uma deadline é **inaceitável** e pode resultar em consequências **catastróficas** (ex: controlo de sistemas médicos ou automóveis).
- **Soft real-time:** falhas ocasionais são toleradas, resultando apenas numa degradação da performance (ex: streaming de áudio/vídeo).

4. How severe is missing a deadline in a hard real-time system? And in a soft real-time system?

- **Hard real-time:** a falha é considerada **crítica** e pode implicar falhas de segurança ou de sistema.
- **Soft real-time:** causa **perda de qualidade**, mas o sistema continua a funcionar.

5. What is a Real-time Operating System (RTOS)?

Um RTOS é um sistema operativo desenhado para garantir que **tarefas críticas são executadas dentro de prazos definidos**, com previsibilidade e determinismo.

6. What distinguishes a RTOS from a general-purpose operating system?

Um RTOS prioriza o **cumprimento de deadlines e o determinismo**, ao passo que sistemas operativos de uso geral (como Windows ou Linux desktop) otimizam a **eficiência média** e o **throughput** sem garantias temporais estritas.

7. Which services and abstractions are typically provided by a RTOS?

- Criação e gestão de **tarefas (tasks)**
- **Agendamento (scheduler)** com prioridades
- **Semáforos** e **mutexes** para sincronização
- **Filas (queues)** para comunicação entre tarefas
- **Timers**
- Gestão de **seções críticas**

- Manipulação de **interrupções**

8. What are the pros and cons of an RTOS-based system (compared to a standalone/bare-metal system)?

Vantagens:

- Modularidade e organização do código
- Melhor gestão de concorrência
- Facilidade de manutenção e escalabilidade
- Suporte a múltiplas tarefas com diferentes prioridades

Desvantagens:

- Aumento da complexidade e da necessidade de memória
- Overhead introduzido pelo RTOS
- Pode ser desnecessário em aplicações muito simples

9. Name (list) a few examples of RTOSs (free, academic, commercial).

- **Free/open-source:** FreeRTOS, Zephyr
- **Académico:** RIOT, ChibiOS
- **Comercial:** VxWorks, QNX, Micrium µC/OS

10. What are RTOS tasks?

São unidades de execução independentes e concorrentes geridas pelo RTOS, com contextos próprios (stack, registos) e associadas a prioridades.

11. What is the role of RTOS scheduler?

O **agendador (scheduler)** decide qual tarefa deve ser executada num dado momento, com base em critérios como **prioridade** e **estado das tarefas**.

12. What are the typical task states in a RTOS?

- **Ready:** pronta para execução
- **Running:** em execução
- **Blocked:** à espera de evento (ex: semáforo, delay)
- **Suspended:** desativada explicitamente

13. What is the purpose of task priorities?

Permitem ao agendador determinar **quais tarefas têm precedência** na execução. Tarefas com prioridade mais alta preemptam as de menor prioridade.

14. What is the RTOS tick?

É um **evento periódico** (interrupção) que marca a passagem do tempo no RTOS, utilizado para gerir delays, timeouts e agendamento baseado em tempo.

15. What are RTOS timers?

São mecanismos que permitem executar ações após um certo tempo ou com periodicidade, sem bloquear tarefas. Podem ser usados para eventos temporizados.

16. What is a shared variable?

É uma variável acedida por múltiplas tarefas ou interrupções. Sem proteção, o acesso concorrente pode causar **condições de corrida (race conditions)**.

17. Why is mutual exclusion of access to shared variables important?

Evita **acessos simultâneos** que podem corromper dados. Garante que **apenas uma tarefa acede** a uma variável crítica de cada vez.

18. What are critical sections (in the code)?

São segmentos de código onde o acesso a **recursos partilhados** ocorre e que devem ser executados **sem interrupções ou preempção**.

19. What are queues?

Estruturas FIFO usadas para **comunicação entre tarefas**, permitindo envio e receção de dados/mensagens de forma segura e sincronizada.

20. What are semaphores and mutexes?

- **Semáforos:** mecanismos de **sinalização** e controlo de recursos. Permitem bloquear e desbloquear tarefas com base na disponibilidade do recurso.
- **Mutexes (Mutual Exclusion):** garantem **exclusividade de acesso** a recursos partilhados. Ao contrário dos semáforos binários, têm mecanismos de prevenção de prioridade inversa.

ASE_2024-25_Slides-T-11

1. Why is power management critical in modern computing and embedded systems?

A gestão de energia é crítica para:

- **Prolongar a autonomia de baterias**, especialmente em dispositivos móveis e IoT.
- **Reduzir o aquecimento e o desgaste dos componentes**.
- **Minimizar o consumo energético total**, fundamental em aplicações alimentadas por fontes limitadas (como painéis solares ou baterias).
- **Melhorar a sustentabilidade e a eficiência energética global** dos sistemas modernos.

2. How is the power consumption affected by the supply voltage and operating frequency of the system?

O consumo de energia tem uma relação aproximadamente quadrática com a **tensão de alimentação (V)** e linear com a **frequência de operação (f)**:

$$P \propto V^2 \times f$$

Ou seja:

- Aumentar a frequência **aumenta o consumo de forma linear**.
- Aumentar a tensão **aumenta o consumo de forma quadrática**.

Portanto, operar a frequências e tensões mais baixas reduz significativamente o consumo energético.

3. Which power modes are predefined in ESP32 and the corresponding features?

O ESP32-C3 define vários **modos de consumo reduzido**, cada um com diferentes compromissos entre consumo e latência de despertar:

- **Active Mode:** CPU e periféricos totalmente operacionais.
- **Light Sleep:** CPU e a maioria dos periféricos digitais desligados; RAM de baixa potência e RTC continuam ativos. Rápido despertar.
- **Deep Sleep:** CPU e RAM desligados; apenas o RTC e parte da RAM de baixa potência permanecem ativos. Muito baixo consumo.
- **Hibernate (não detalhado no documento):** modo de consumo ainda mais baixo, com perda quase total do contexto.

4. How is a reduced power management mode entered in ESP32?

A entrada nos modos de baixo consumo é feita por software, através de chamadas à API do **ESP-IDF** (como esp_light_sleep_start() ou esp_deep_sleep_start()), geralmente após configurar os **eventos de despertar** (wake-up sources) desejados.

5. What are the typical wake-up sources in ESP32?

As fontes de despertar mais comuns incluem:

- **Timers RTC:** permitem acordar após um tempo definido.
- **GPIOs externos:** sinais digitais podem despertar o chip.
- **Eventos de periféricos:** como UART, sensores, etc.
- **Touch pads ou sensores de temperatura**, dependendo da configuração.

Estas fontes podem ser combinadas para despertar o ESP32 de forma flexível e eficiente.

ASE_2024-25_Slides-T-12

1. How are WiFi networks organized? What is the "anatomy" of a WiFi network?

As redes WiFi são organizadas em torno de um **Access Point (AP)**, que atua como ponto central de comunicação para múltiplos dispositivos (Stations). Os elementos principais são:

- **AP (Access Point):** fornece o serviço de rede.
- **Stations (STA):** dispositivos clientes que se ligam ao AP.
- **SSID:** nome identificador da rede.
- **Canal:** frequência específica usada na comunicação.

Há também redes em modo **ad-hoc** (sem AP), como **ESP-NOW**, onde dispositivos comunicam diretamente.

2. What are the frequency bands used in WiFi?

WiFi opera em bandas ISM:

- **2.4 GHz** – mais comum, mas sujeita a interferência (Bluetooth, micro-ondas, etc.).
- **5 GHz** – maior largura de banda e menos interferência, mas menor alcance.
- (Bandas de **6 GHz** disponíveis para WiFi 6E – **não suportadas** pelo ESP32-C3.)

3. What are the capabilities of each WiFi version?

Versão	Velocidade Máxima	Banda	Características
802.11b	~11 Mbps	2.4 GHz	Primeira geração, obsoleta.
802.11g	~54 Mbps	2.4 GHz	Compatível com 802.11b.
802.11n	~600 Mbps	2.4 / 5 GHz	MIMO, mais estável.
802.11ac	>1 Gbps	5 GHz	Alta velocidade, OFDM.
802.11ax (WiFi 6)	>10 Gbps	2.4 / 5 / 6 GHz	MU-MIMO, OFDMA, eficiência elevada.

ESP32-C3 suporta WiFi 4 (802.11n), com comunicação em **2.4 GHz**.

4. What are the purposes, advantages and constraints of ISM bands?

- **Propósito:** Reservadas internacionalmente para usos Industriais, Científicos e Médicos (ISM).
- **Vantagens:** Uso livre (sem licenciamento), baixo custo, ampla disponibilidade.

- **Restrições:** Elevada interferência e partilha com outros dispositivos (Bluetooth, ZigBee, micro-ondas), podendo afetar desempenho.

5. What are the protocols/services/applications that share the RF spectrum bands with WiFi?

- **Bluetooth / BLE**
- **ZigBee**
- **ESP-NOW**
- **Sistemas de alarme sem fios**
- **Micro-ondas (interferência acidental)**
- **Dispositivos IoT em geral**

Todos competem no **espectro de 2.4 GHz**, aumentando a probabilidade de colisões e degradação da qualidade da ligação.

6. Is WiFi adequate for safety-critical applications?

Não. O WiFi **não garante determinismo nem latências fixas**, sendo baseado em protocolo CSMA/CA com acessos imprevisíveis ao meio. É, por isso, **inadequado para aplicações críticas em tempo real**, como sistemas de controlo industrial ou médico.

7. What WiFi version is supported by ESP32-C3?

O ESP32-C3 suporta **WiFi 4 (IEEE 802.11n)** na banda **2.4 GHz**, com suporte a:

- Modo **Station**
- Modo **SoftAP**
- **ESP-NOW**

8. Which WiFi networking APIs are provided with ESP/IDF?

O framework **ESP-IDF** fornece várias APIs para gestão de WiFi:

- **esp_wifi_init(), esp_wifi_set_mode(), esp_wifi_start()**
- **APIs para configuração de SSID, password, canal, eventos WiFi**
- Suporte a **ESP-NOW** (comunicação direta, sem infraestrutura)
- APIs para **scan, conexão automática, reconexão, DHCP**, etc.

9. What are the restrictions on using WiFi and BLE simultaneously in an ESP32-C3 SoC?

O ESP32-C3 utiliza **um único rádio para WiFi e BLE**, o que implica:

- **Partilha do tempo de antena** entre WiFi e BLE.
- Não é possível operar ambos **simultaneamente com alta performance**.

- Pode ocorrer **degradação de desempenho** ou **latência elevada** em cenários intensivos.
- O programador deve gerir **prioridades e coexistência** com cuidado.