Gerenciamento de Conflito SPI no Kit BitDogLab: Uma Abordagem com Estados Persistentes em RAM e Soft Resets

Marcony Henrique Bento Souza [1], **Danielly dos Santos Almeida** [1], **Priscila Lima Rocha**[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Maranhão
Campus Monte Castelo, São Luís - MA.

{marcony.henrique, danielly.s}@acad.ifma.edu.br, priscila.rocha@ifma.edu.br

Abstract. *This article presents an alternative strategy for the management of access to hardware resources in the BitDogLab educational development kit. In this way, the implementation of a persistent software architecture is proposed to get around the SPI bus contention of the BitDogLab educational kit, using a persistent state machine, which uses specific sections of RAM memory that preserve the context after soft resets. The functional validation of the prototype showed the effectiveness of the approach, legitimizing it as a viable strategy for IoT systems that face resource conflicts at the hardware level.*

Resumo. *Este artigo apresenta uma estratégia alternativa para o gerenciamento de acesso a recursos de hardware no kit educacional de desenvolvimento BitDogLab. Desta forma, se propõe a implementação de uma arquitetura de software persistente para contornar a contenção de barramento SPI do kit educacional BitDogLab, utilizando uma máquina de estados persistentes, que usa seções específicas da memória RAM que preservam o contexto após soft resets. A validação funcional do protótipo mostrou a eficácia da abordagem, legitimando-o como uma estratégia viável para sistemas IoT que enfrentam conflitos de recursos a nível de hardware.*

1. Introdução

Sistemas embarcados, antes projetados para funções únicas e de baixo custo, tornaram-se multifuncionais com a ascensão da Internet das Coisas (IoT) [Denardin and Barriquello 2019, Taurion 2005]. Essa evolução gerou um desafio em plataformas com recursos limitados, que se trata do gerenciamento de múltiplos periféricos que competem por barramentos de comunicação compartilhados, como o Serial Peripheral Interface (SPI).

Este trabalho acadêmico tem como principal temática o conflito de acesso ao periférico de comunicação SPI, notado durante um estudo de caso da plataforma educacional BitDogLab em que se desenvolvia um sistema para monitoramento de transportes escolares em zonas com baixa conectividade. No microcontrolador utilizado existem dois periféricos responsáveis pela comunicação SPI, denominados SPI0 e SPI1, porém o kit educacional BitDogLab é construído de tal forma que as conexões responsáveis pelo periférico SPI1 são usadas por outros componentes soldados [Raspberry Pi Foundation 2024]. Portanto, o SPI0 é compartilhado entre os componentes

principais na coleta e armazenamento de dados, além da conexão em rede. Os componentes usados são, o MFRC522, um leitor de Identificação por Radiofrequência (RFID) e as tags RFID; um módulo responsável pela leitura e escrita no cartão SD e um cartão SD com armazenamento de 16GB e a conexão Wi-Fi que é gerenciada pelo microcontrolador que se baseia no processador RP2040 (dual-core ARM Cortex-M0+). Tentativas de alternar o uso do barramento SPI entre o Wi-Fi e os outros periféricos resultaram em um erro crítico no sistema (PANIC), tornando inviáveis as abordagens tradicionais de gerenciamento de recursos, como mutexes.

A principal contribuição deste trabalho acadêmico se dá validação de uma arquitetura de software que contorna o problema e não no desenvolvimento de um produto adequado para comercialização. A arquitetura se baseia em um sistema de estados persistentes, que armazena os estados em uma região da memória RAM que é apagada em reinicializações ocorridas por perca de alimentação (hard reset), mas não em reinicializações desencadeadas pelo Watchdog Timer (WDT) [Wolf 2016], uma característica do microcontrolador RP2040 que permite a persistência de dados entre ciclos de reinicialização leve (soft resets). O WDT se trata de um temporizador de hardware que, embora tradicionalmente usado para recuperar o sistema de travamentos, é aqui empregado como ferramenta ativa para forçar reinicializações controladas. Uma reinicialização é acionada na transição entre os estados de coleta e transmissão de dados e permite que o barramento SPI seja completamente reinicializado antes de ser alocado a outro periférico, essa abordagem foi a única que garantiu o funcionamento dos periféricos de maneira confiável dentro do contexto discutido [Kopetz 2003].

Diante do exposto, o objetivo geral deste trabalho é validar uma arquitetura de software resiliente, baseada em estados persistentes e reinicializações controladas, para contornar o conflito de hardware no barramento SPI do kit BitDogLab. Como objetivos específicos, busca-se: (i) implementar uma máquina de estados que preserve seu contexto na RAM após soft resets; (ii) empregar o Watchdog Timer (WDT) como mecanismo ativo para a transição entre os modos de operação; e (iii) demonstrar a viabilidade da solução em um protótipo funcional.

Dessa forma, se faz necessário a pesquisa, desenvolvimento e implementação de alternativas para gerenciamento de recursos de hardware compartilhados em ambientes multicore. Como as abordagens tradicionais se mostraram ineficazes para prevenir o travamento de hardware (PANIC) observado, fez-se necessário o desenvolvimento da arquitetura alternativa

2. Fundamentação e Trabalhos Relacionados

2.1. Mecanismos de resiliência de firmware

Em [Regenscheid 2018], diretrizes técnicas são propostas para apoiar a resiliência de firmware contra situações adversas, sendo baseado nos principios de Proteção dos dados colhidos, Detecção de falhas e ataques e Recuperação do firmware em integridade. Entretanto, devido ao hardware do Raspberry Pico W, com multiplos periféricos requisitando acesso ao mesmo barramento SPI, o sistema falha e a execução do firmware é extinguida.

Uma das técnicas de reinicialização e recuperação de firmware é o WatchDog, artificio comum para forçar paradas após erros durante a execução

[Mahmood and McCluskey 1988]. Devido a sua capacidade de funcionar de forma concorrente ao processamento principal, é uma técnica que pode tão igualmente perigosa, já que seu controle de forma incorreta pode se sobrepor a execução principal atrapalhar o processo normal do sistema.

2.2. Gerenciamento de recursos compartilhados

Há uma variedade de técnicas para o gerenciamento de recursos hardware compartilhados, em especial utilizando de RTOS. [Anderson 1993] através de escalonador, faz as reservas de recursos com base nas piores cargas de trabalho,[Stankovic and Ramamritham 1991] fornece suporte em tempo real para sistemas com multiprocessadores, usando um sistema de planejamento de prioridades dinâmico. [Kurose 1992] derivou limites de ocupação e atraso para acesso de recursos compartilhados e usando escalonamento baseado em FIFO. Existem ainda propostas de um framework, [Gopalan and Kang 2007] apesar de apresentar a solução apenas em âmbito teórico.

As técnicas de exclusão mútua, ou mutexes são as mais recorrentes para gerenciar o controle de acesso a recursos de hardware [Fiestas and Bustamante 2019], entretanto, em sistemas multicore essa abordagem pode ser fatal, devido ao invalidamento da variável de bloqueio que é constantemente transferida de um núcleo para o outro. A comparação entre as propostas é detalhada a seguir.

Tabela 1. Comparativo entre Propostas Encontradas

Artigos	Abordagem	Características
[Mahmood and McCluskey 1988]	Watchdog Timer	Coprocessador dedicado, Monitoramento contínuo, Independência, Sobreposição de software
[Anderson 1993]	Metascheduler	Work ahead, Software mediador, Abstração de recursos
[Stankovic and Ramamritham 1991]	Spring Kernel	Prioridades, Restrições de tempo, Tempo Real
[Kurose 1992]	Previsão de limites de desempenho dinamicamente	Redimensionamento, Deadlocks
[Kurose 1992]	Mutex	Velocidade, Segurança, Bloqueios indevidos

3. Metodologia

A plataforma de hardware utilizada é o kit educacional BitDogLab, que possui um microcontrolador Raspberry Pi Pico W. A principal restrição de design deste kit é que os pinos

do segundo periférico de comunicação, o SPI1, são utilizados por outros componentes na placa. Isso força que todos os módulos que utilizam essa comunicação — o leitor RFID (MFRC522), o módulo de cartão SD e o chip de conectividade Wi-Fi — compartilhem o mesmo barramento SPI0, gerando o conflito de hardware abordado neste estudo.

3.1. Arquitetura Proposta

A solução foi desenvolvida para garantir a operação contínua do sistema de monitoramento, apesar da impossibilidade de compartilhamento dinâmico do barramento SPI0 com os materiais disponíveis. A arquitetura usa os dois núcleos do processador RP2040 e implementa um ciclo de vida baseado em estados que são preservados através de reinicializações.

3.2. Divisão de Responsabilidades e Identificação do Conflito

A arquitetura de software atribui ao Núcleo 0 o gerenciamento dos periféricos responsáveis por coleta e armazenamento de dados (leitor RFID e cartão SD) e da interface com o usuário, enquanto o Núcleo 1 é dedicado exclusivamente às operações que utilizam Wi-Fi e comunicação via protocolo MQTT. Essa separação tem como objetivo isolar as tarefas de tempo real das latências da rede. Porém, o conflito surge porque todos esses componentes — RFID, cartão SD e o módulo Wi-Fi — estão fisicamente conectados ao mesmo periférico SPI0 no microcontrolador. O uso do barramento pelo módulo Wi-Fi deixa-o em um estado indefinido que causa um travamento fatal (PANIC) ao tentar inicializar outro periférico no mesmo barramento sem usar a reinicialização com WDT.

3.3. Sistema de Estados Persistentes em RAM

Para manter o contexto mesmo com as reinicializações, a continuidade do sistema é mantida através de estados persistentes. Uma estrutura de dados C é mapeada para um endereço de memória específico (0x20040000), uma região da SRAM que o RP2040 não inicializa durante um soft reset [Raspberry Pi Foundation 2024]. Para garantir a integridade dos dados após uma reinicialização, um ”valor mágico”(0xDEADBEEF) é escrito na estrutura. Na inicialização, o sistema verifica a presença deste valor para diferenciar um estado válido, salvo na execução anterior, e memória não inicializada [Wolf 2016].

A escolha da memória RAM no lugar do cartão SD para persitência dos estados ocorre pois tentativas de alternar o uso do barramento SPI resultaram em um erro crítico (PANIC), exigindo uma reinicialização para restaurar a operação e o uso do cartão SD seria muito lento para garantir que o estado fosse salvo antes da reinicialização forçada pelo WDT. A persistência é garantida ao mapear uma estrutura de dados para o endereço de memória 0x20040000, uma região da SRAM que o RP2040 não inicializa durante um soft reset. Esta técnica, no entanto, não preserva os dados em caso de um ciclo de energia completo. O risco de sobreposição de memória é mitigado pelo mapeamento explícito para um endereço fixo, uma prática controlada no firmware que isola esta região da alocação dinâmica de memória da aplicação.

O comportamento do sistema é baseado em uma máquina de estados finitos com cinco modos operacionais: SYSTEM_MODE_RFID_SD (coleta e armazenamento), SYSTEM_MODE_SD_READ_SEND (leitura e preparação

para envio), SYSTEM_MODE_SD_CLEANUP (limpeza de dados enviados), SYSTEM_MODE_NORMAL_WIFI (conectividade) e SYSTEM_MODE_POST_SEND (pós-envio). A transição entre esses modos com WDT é o que permite o funcionamento do sistema e está explicada em formato de fluxograma na Figura 1.

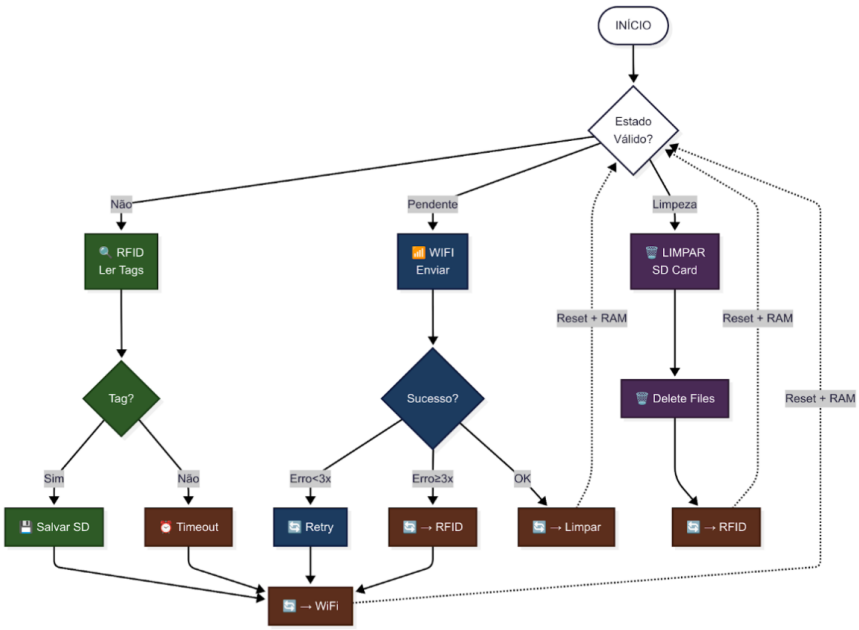

Figura 1. Fluxograma representando o funcionamento do código

O sistema inicia sempre no estado de validação, onde verifica a validade do estado persistente armazenado na memória RAM. Esta verificação inicial determina qual caminho o sistema seguirá: se o estado não for válido (primeira inicialização), o sistema direciona para o modo RFID; se existirem dados pendentes para transmissão, vai diretamente para o modo WIFI; se estiver configurado para limpeza, segue para o modo de limpeza .

No modo RFID, o sistema entra em um loop de detecção de tags através da decisão “Tag”. Caso nenhuma etiqueta RFID seja detectada dentro do tempo limite, o sistema segue para “Timeout” e posteriormente para a reinicialização. Quando uma etiqueta é detectada, o sistema processa os dados e executa a operação de salvamento para armazenamento local no cartão SD, seguindo então para a reinicialização para iniciar o processo de transmissão. No modo WIFI, o sistema tenta transmitir os dados coletados através da decisão “Sucesso”. Se o envio falha e o número de tentativas é menor que três, segue para uma nova tentativa. Se o envio falha três ou mais vezes, deve retornar ao modo RFID. Quando o envio é bem-sucedido, o sistema transita para iniciar o processo de limpeza.

O modo de limpeza executa a operação “Delete Files” para deletar os arquivos já transmitidos do armazenamento local, seguindo para retornar ao modo RFID e reiniciar o ciclo completo. As transições na parte inferior do fluxograma representam pontos de reset coordenado onde o sistema utiliza soft resets via watchdog timer, preservando o estado na RAM através das linhas pontilhadas ”Reset + RAM”que retornam ao estado inicial de validação. Esta arquitetura garante que o sistema sempre continue operando

do ponto onde parou, mesmo após reinicializações, criando um ciclo contínuo de coleta, transmissão e limpeza de dados.

A transição entre estados que precisam do periférico SPI é o ponto onde o conflito existe. A solução inovadora deste trabalho é realizar essa transição através de uma reinicialização controlada. Dessa forma, o WDT, tradicionalmente usado para recuperação de falhas, se torna um componente ativo e fundamental na lógica de controle do programa.

4. Resultados

A validação da arquitetura foi realizada com testes em ambiente de laboratório, focada em verificar a operação dos mecanismos de resiliência, em vez de coletar métricas formais de desempenho. O protótipo consistiu na placa BitDogLab, equipada com um Raspberry Pi Pico W, um leitor RFID MFRC522, um módulo de cartão SD e um display OLED para feedback visual. As figuras 2 e 3 são o diagrama do circuito e o protótipo montade respectivamente.

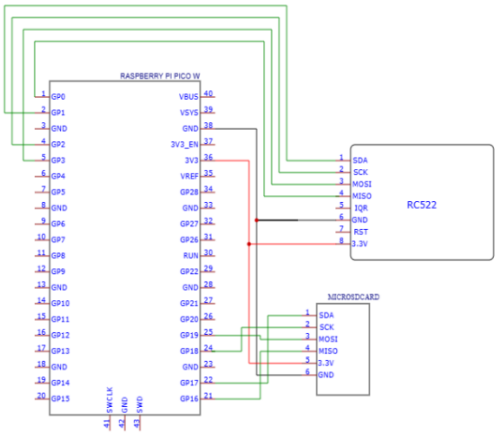

Figura 2. Diagrama de conexão dos módulos RFID e Cartão SD ao barramento SPI0 do Raspberry Pi Pico W

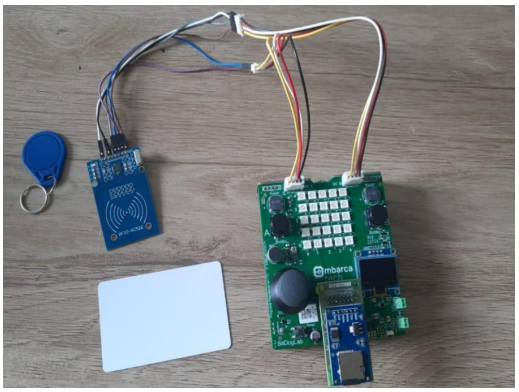

Figura 3. Protótipo do sistema de monitoramento

A validação da arquitetura foi realizada em ambiente de laboratório, com foco em verificar a operação dos mecanismos de resiliência em vez de coletar métricas formais de desempenho. O protótipo, composto pela placa BitDogLab, um leitor RFID MFRC522, um módulo de cartão SD e um display OLED, foi submetido a três cenários para simular a operação real: (i) Coleta Offline, onde o sistema operou por 30 minutos em um ambiente sem conexão com a rede WiFi, armazenando os arquivos apenas no cartão SD, realizando 15 leituras RFID e alcançando 100% de recuperação de estados após reinicializações forçadas pela ausência de conectividade. Em seguida, no segundo cenário, (ii) Transmissão Online, o dispositivo possuía 5 registros já armazenados na memória RAM e o sistema transmitiu com sucesso todos os dados via MQTT durante um período de 10 minutos, também com 100% de recuperação de estado. Finalmente, na terceira etapa, (iii) Ciclo Completo, que durou 60 minutos, o protótipo possuia conexão WiFi e era capaz de

alternar entre realizar leituras RFID e transmiti-las, coletando 25 leituras e transmitindo todos os 25 registros.

Porém, no ciclo completo, a quantidade de resets ultrapassou o mínimo necessário, pois mesmo com conectividade disponível, o microcontrolador, por vezes, não conseguia realizar a conexão e acionava uma reinicialização para tentar novamente. Em um cenário real com ainda maior instabilidade em comparação ao ambiente de laboratório, poderia acarretar em uma maior quantidade de reinicializações, afetando tanto a disponibilidade do protótipo quanto a vida útil da bateria utilizada. Por outro lado, mesmo neste cenário, os testes mostraram que não houve perda de dados. Portanto, a arquitetura proposta pode ser considerada validada, pois mesmo com as limitações existentes o sistema foi capaz de alternar entre os diferentes periféricos e realizar as ações necessárias para coleta e transmissão de dados.

Tabela 2. Resumo dos Cenários de Validação da Arquitetura

Cenário teste	**Duração**	**Leituras RFID**	**Transmissões MQTT**	**Recuperação de Estados**
Coleta Offline	30 min	15	0	100%
Transmissão Online	10 min	5	5	100%
Ciclo Completo	60 min	25	25	100%

Os resultados observados confirmaram o funcionamento da arquitetura proposta. O sistema apresentou a capacidade de coletar dados via RFID, armazená-los localmente no cartão SD e, quando a conectividade estava disponível, realizar a transição de estado via soft reset e transmitir com sucesso os dados para um servidor MQTT. A principal constatação foi a identificação da limitação crítica: embora a arquitetura de software funcione, a necessidade de reinicializações constantes para alternar entre os modos de coleta e transmissão confirma que a configuração de hardware com um único barramento SPI compartilhado é inadequada para uma aplicação de produção contínua.

5. Conclusão

Este trabalho demonstrou a viabilidade de uma arquitetura de software resiliente para contornar um conflito de hardware na BitDogLab. A combinação de um sistema de estados persistentes em RAM com o uso do watchdog timer para gerenciar transições de estado provou ser um método eficaz para garantir a integridade dos dados e a operação contínua em um protótipo de monitoramento de estudantes em transportes escolares. A hipótese de que seria possível manter a operação através de reinicializações controladas foi validada como prova de conceito.

A principal limitação, no entanto, está na própria solução, que funciona como um método para contornar a restrição de hardware. A dependência de soft resets para a operação normal do sistema, embora funcional, não é ideal para um ambiente de produção que exige alta disponibilidade e estabilidade.

Como trabalhos futuros, a prioridade máxima é a resolução do conflito em nível de hardware. A solução consiste na migração dos periféricos de armazenamento e lei-

tura (cartão SD e RFID) para o barramento SPI1, que está disponível no Raspberry Pi Pico W, mas não na BitDogLab que utiliza os pinos do SPI1 em outros componentes soldados no kit educacional. Essa alteração de hardware eliminaria o problema, permitindo que a arquitetura de software desenvolvida opere de forma contínua e estável, com maior desempenho e permitindo o uso em ambientes reais. Adicionalmente, para mitigar o consumo de energia em cenários de conectividade instável, futuros trabalhos podem investigar uma estratégia de transmissão em lotes, ativando o módulo Wi-Fi apenas em intervalos de tempo pré-definidos.

Referências

Anderson, D. P. (1993). Metascheduling for continuous media. *ACM Trans. Comput. Syst.*, 11(3):226–252.

Denardin, G. W. and Barriquello, C. H. (2019). *Sistemas operacionais de tempo real e sua aplicaçao em sistemas embarcados*. Editora Blucher.

Fiestas, J. I. and Bustamante, R. E. (2019). A survey on the performance of different mutex and barrier algorithms. In *2019 IEEE XXVI International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, pages 1–4.

Gopalan, K. and Kang, K.-D. (2007). Coordinated allocation and scheduling of multiple resources in real-time operating systems. In *Workshop on Operating Systems Platforms for Embedded Real-Time applications*, page 48.

Kopetz, H. (2003). Time triggered architecture. *ERCIM News*, (52):10–11.

Kurose, J. (1992). On computing per-session performance bounds in high-speed multihop computer networks. *Measurement and Modeling of Computer Systems*, 20:128–139.

Mahmood, A. and McCluskey, E. (1988). Concurrent error detection using watchdog processors-a survey. *IEEE Transactions on Computers*, 37(2):160–174.

Raspberry Pi Foundation (2024). Raspberry Pi Pico W Product Brief.

Regenscheid, A. (2018). Platform firmware resiliency guidelines.

Stankovic, J. A. and Ramamritham, K. (1991). The spring kernel: A new paradigm for real-time systems. *IEEE Softw.*, 8(3):62–72.

Taurion, C. (2005). *Software Embarcado-A nova onda da Informática*. Brasport.

Wolf, M. (2016). *Computers as Components: Principles of Embedded Computing System Design*. Morgan Kaufmann, Cambridge, 4 edition.