DOI: 10.36558/rsc.v11i2.7267

Sistemas Operacionais Para Software Embarcado: Um Mapeamento Sistemático Da Literatura

Aloysio Augusto Rabello de Carvalho
UNIFESP
aloysio.rabello@unifesp.br

Luiz Eduardo Galvão Martins
UNIFESP
legmartins@unifesp.br

ABSTRACT

Embedded systems are constantly increasing today, becoming ubiquitous in people's lives. Today there are a wide range of devices, so every day we have new technologies, optimizations, and techniques being applied to embedded systems as they need a high degree of reliability, and security, pose new challenges to development. This systematic mapping of the literature (**MSL**) aims to search for signs that show the real impact of using a real-time operating system (**RTOS**) in embedded software. We carry out an **MSL** using the ACM Digital Library, IEEE Digital Library, ISI Web of Science, Science @ Direct, Springer Link, and Scopus databases, looking for articles that relate **RTOS** and Embedded Systems to be reviewed. 245 articles were found on the topic. We excluded articles that were out of context maintaining 21 articles on performance, **RTOS**, embedded system, embedded software development, advantages and disadvantages regarding the use of **RTOS**. We conducted an **MSL** with the purpose of identifying the main methods, difficulties, and solutions in the embedded system development. After the research we realized there are no articles that lead to a reasonable conclusion because none of the articles analyzed clearly show the relationship between the use of an **RTOS** in embedded systems and the real impact, pros, cons, and obstacles for adopting it.

CCS Concepts

•**Computer systems organization** → **Real-time operating systems;** *Embedded software;*

Keywords

Operational systems; Embedded Software; Embedded System; Real-Time Operating System

1. INTRODUÇÃO

Sistemas embarcados estão crescendo constantemente na atualidade, se tornando onipresentes na vida das pessoas. Dessa forma, os projetos que versam sobre desenvolvimento de software embarcado sofreram um grande aumento em relação à quantidade, em seu tamanho, e até mesmo complexidade. Acompanhando, assim a evolução da complexidade dos componentes eletrônicos que necessitam de um alto grau de confiabilidade e segurança[20]. Hoje contamos com um vasto número de opções para hardware embarcado, contendo diversas marcas, recursos, potências e até mesmo componentes eletrônicos,consequentemente, estes aspectos citados anteriormente corroboram para novos desafios no desenvolvimento de softwares embarcados. Engenheiros e analistas de sistemas buscam novas soluções para estes problemas, sendo assim, este mapeamento sistemático da literatura ou revisão bibliográfica, tem como objetivo investigar o estado da arte dos estudos relacionados ao uso de **RTOS** no desenvolvimento de software embarcado, sendo assim durante esta **MSL** iremos buscar indícios que mostrem o real impacto do uso de um **RTOS** no desenvolvimento de softwares embarcados. Esta **MSL** segue as diretrizes propostas por [17], agrupando as atividades do processo de revisão em etapas: planejamento, condução e comunicação da **MSL**. Para alcançarmos os objetivos desta revisão sistemática, buscamos nas principais bases de dados procurando artigos que nos ajude a responder as perguntas pertinentes da nossa pesquisa. Após uma análise dos 245 artigos encontrados, classificamos e aceitamos apenas 21 relevantes para esta pesquisa. Foi aplicado um **MSL** com o propósito de conseguir respostas para as questões de pesquisa.

1.1 Sistema Embarcado

Um sistema embarcado é um sistema microprocessado encapsulado, com desenvolvimento de hardware e software otimizado para execução de um conjunto de funcionalidades que, nor-malmente, não podem ser modificadas sem que todo o sis-tema volte a etapa de desenvolvimento, ao contrário de um computador pessoal (**PC**), onde seu propósito é geral e pode executar uma extensa gama de tarefas. Em um sistema em-barcado temos normalmente como prioridades o tamanho, *performance* e gerenciamento energético. Seus software normalmente é chamado de *firmware* e armazenado em sua mem ória flash ou **ROM** ao invés de discos rígido e por muitas vezes o sistema pode ser executado com recursos limitados, como por exemplo sem utilizar periféricos como *mouse*, monitor ou até teclado. Ademais na grande maioria dos sistemas

embarcados e a sua comunicação com o mundo exterior se passa por sensores, leds, pequenas telas **LCD**, motores, *interfaces* **USB** ou Serial e internet ou rede local. Temos como exemplo: roteadores, bomba de infusão de insulina, leitores **MP3**, impressoras e até semáforos de transito.

1.2 Sistema Operacional de Tempo Real

RTOS é a parte fundamental para o funcionamento de **RTES** *(Real-Time Embedded Systems)*, como por exemplo gasto de energia, tempo de resposta e uso eficiente de memória. Em um **RTES** a principal função é a resposta precisa em um tempo de resposta pré-definido. Estes sistemas são divididos em três classes, *hard, soft e firm* **RTOS**.

Table 1: Tipos de RTOS

RTOS	Descrição	Exemplo
Hard	Prazo tratado de forma muito estrita, ou seja, uma tarefa tem tempo pré-definido para início e fim, sendo sempre concluída dentro do período de tempo atribuído.	Sistemas embarcados em aeronaves ou sistema na área da medicina
Firm	Prazo não é tratado de forma tão estrita como no Hard-RTOS, neste caso pode acontecer pequenos atrasos que pode não causar grandes impactos, porem podem causar efeitos indesejados no funcionamento do sistema.	Sistemas multimídia.
Soft	Neste caso alguns atrasos são aceitos pelo sistema. Existe um prazo pré-definido para uma determinada tarefa, porem pequenos atrasos de tempo são aceitáveis, ou seja, os prazos nestes sistemas não são tratados de forma firme.	Sistema de transações online e sistema de cotação de preços.

Em um **RTOS** existem vários requerimentos como, previsibilidade das tarefas, pouca variação de violação de *deadlines*, configurabilidade do sistema, eficiência dos componentes do sistema, isto é, granularidade de tempo, *threads*, gerenciamento de recursos entre outros [15].

O objetivo primário de um **RTOS** é o uso eficiente de técnicas de agendamentos de tarefas, uma boa política de gerenciamento de recursos e uma performance previsível de tarefas [18].

1.3 MSP430

São microcontroladores **RISC** (*Reduced Instruction Set Computer*) de 16 bits fabricado pela *Texas Instruments*, voltados para aplicações de baixo consumo energético e disponível em quatro famílias. Seu microcontrolador possui um conjunto de 27 instruções físicas e 24 instruções emuladas com um total de 16 registradores.

2. MÉTODOS

2.1 Mapeamento Sistemático da Literatura

A revisão da literatura ou revisão bibliográfica, é a fundamentação teórica adotada para tratar o tema e o objetivo de pesquisa [8]. Uma revisão sistemática é desenvolvida para reunir e avaliar evidencias disponíveis referentes a um tema de pesquisa especifico [4], atuando como um meio para identificar, avaliar e interpretar toda pesquisa disponível e relevante. A metodologia utilizada na **MSL** deve ser confiável, rigorosa e que permita auditagem [19]. A revisão sistemática envolve conjuntos de atividades dentro de conjuntos de fases no seu processo de condução, previamente estabelecidos por protocolos da revisão, estes protocolos irão conduzir de forma sintetizada todo o processo de condução da revisão. A seguir, na figura 1, pode-se ver o processo de condução a ser utilizado [4] [16].[REESCREVER]

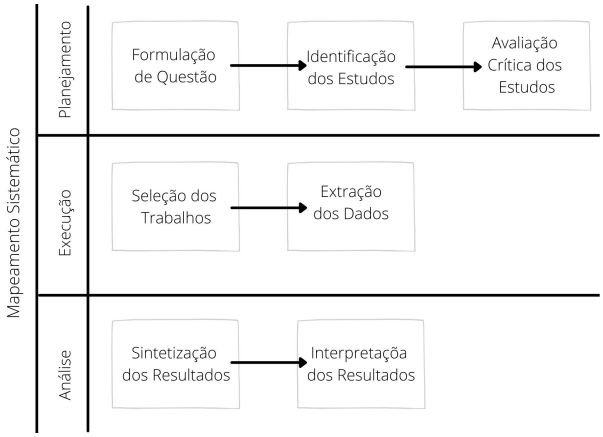

Figure 1: Processo do MSL Fonte: Autor

2.2 Objetivos

O objetivo deste mapeamento sistemático da literatura (**MSL**) é investigar o estado da arte sobre a utilização de sistemas operacionais no desenvolvimento de software embarcado, buscando identificar vantagens, desvantagens, principais características, implicações no ciclo de vida de desenvolvimento do software embarcado, indicações de uso. A condução do **MSL** seguiu 3 etapas: planejamento, execução e análise dos resultados obtidos. Estas serão apresentadas no decorrer deste trabalho.

3. EXECUÇÃO

3.1 Definição das Questões de Pesquisa

- Questão 01 - Quais são os sistemas operacionais embarcados mais utilizados no desenvolvimento de software embarcado?
  - Essa pergunta tem por objetivo avaliar o mercado de software embarcado, tentar entender qual os **SO**'s mais utilizados e por quê. E quais são os planos que as empresas desenvolvedoras de software embarcado planejam para o futuro.
- Questão 02 - Quais a vantagens do uso dos sistemas operacionais embarcados?
  - Essa pergunta tem por objetivo avaliar o ganho que podemos ter ao se utilizar um **SO** embarco em nossos projetos, qual o ganho em qualquer

aspecto do ciclo de vida de um projeto de desenvolvimento.

- Questão 03 - Quais as desvantagens ou dificuldades?
  - Essa pergunta tem por objetivo avaliar quais problemas que podemos ter ao se utilizar um **SO** embarco em nossos projetos, qual pode ser a perda em qualquer aspecto do ciclo de vida de um projeto de desenvolvimento e quais são as dificuldades de se desenvolver um sistema embarcado utilizando um **SO**.
- Questão 04 - Quais são as principais características dos sistemas operacionais embarcados?
  - Essa pergunta tem por objetivo avaliar as principais características que os desenvolvedores buscam ao utilizar um **SO** embarcado em seu projeto.
- Questão 05 - Quais os impactos de usar sistemas operacionais no desenvolvimento de software embarcado?
  - Essa pergunta tem por objetivo avaliar quaisquer eventuais ganho ou perdas ao se utilizar um **SO** no desenvolvimento de software embarcado.

3.2 Seleção de Fontes

As Strings de buscas foram aplicadas nas seguintes bases de dados:

- ACM Digital Library (http://portal.acm.org)
- IEEE Digital Library (http://ieeexplore.ieee.org)
- ISI Web of Science (http://www.isiknowledge.com)
- Science@Direct (http://www.sciencedirect.com)
- Scopus (http://www.scopus.com)
- Springer Link (http://link.springer.com)

3.3 Idiomas

Foram considerados apenas artigos publicados em inglês.

3.4 Termos PICOC

A primeira etapa para criar as Strings de busca foi encontrar os nossos termos para *population, intervention, comparison, outcome, context* (**PICOC**), na tabela 2 podemos ver todos os termos **PICOC** utilizados para a criação da String de busca:

Table 2: Termos PICOC

PICOC	Terms
Population	*Embedded Operating System*
Intervention	*Embedded operating systems used in embedded software development*
Comparison	*Embedded software development with and without embedded operating systems*
Outcome	*Pros and cons in using embedded operating systems in embedded software development*
Context	*Embedded software development*

3.5 Busca

3.5.1 String de Busca

A String de busca utilizada para executar a pesquisa nas bases de dados foi a seguinte: *("Embedded operating system" OR RTOS OR "Real Time Operating System") AND "Embedded Software Development".* e todas as buscas e seleções de artigos foram executadas entre agosto e setembro de 2020. Após executar as Strings de busca nas bases de dados selecionadas obtivemos como resultado ao todo 245 artigos, divididos como pode ser visto na figura 2.

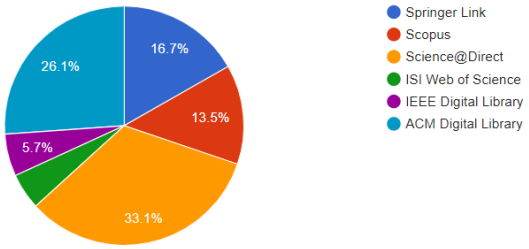

Figure 2: Total de Artigos por Base de Dados Fonte: www.parsif.al

3.5.2 Período de Busca

Foram considerados artigos com publicação no período entre 2000 e 2020.

3.6 Critérios

3.6.1 Critérios de Inclusão:

Foram incluídos artigos seguindo os seguintes critérios:

- Artigos que apresentem/discutam características de sistemas operacionais embarcados
- Artigos que apresentem estudos/discussão usando sistemas operacionais embarcados no desenvolvimento de software embarcado
- Estudos que apresentem vantagem e/ou desvantagem do uso de sistemas operacionais embarcados no desenvolvimento de software embarcado
- Estudos que comparem sistemas operacionais embarcados

3.6.2 Critérios de Exclusão:

Foram excluídos artigos seguindo os seguintes critérios:

- Artigos duplicados
- Artigos que não estejam escritos em Inglês
- Artigos resumidos
- Artigos cujo o texto completo não está disponível.

Nos resultados obtidos nas bases de dados, foram encontrados um total de 245 artigos com tema sobre **RTOS** e sistema embarcado, a princípio excluímos apenas os artigos que estavam fora do escopo, como por exemplo, artigos que demonstrava cursos relacionados a softwares embarcados, desenvolvimento de apenas uma funcionalidade, resumos e artigos que apresentava funcionalidades não relacionadas ao escopo do projeto.

3.7 Seleção

3.7.1 *Artigos Selecionados por Ano*

Foram selecionados artigos entre 2000 e 2019, após a seleção os artigos ficaram distribuídos da seguinte forma como pode ser visto na figura 3:

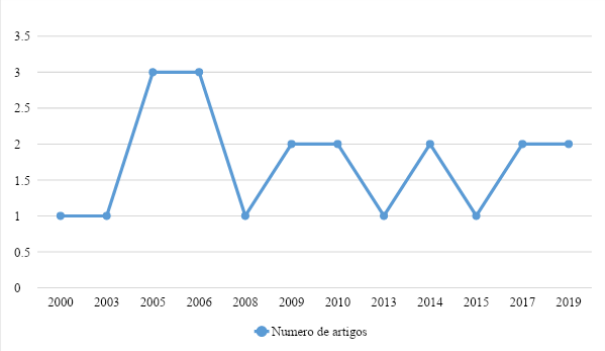

Figure 3: Artigos Selecionados por Ano Fonte: www.parsif.al

3.7.2 *Etapas de seleção*

A Figura 4 evidencia que, ao todo foram utilizadas 5 etapas para esse **MSL**, durante a primeira etapa utilizamos as strings de busca nas bases de dados, na segunda etapa fizemos a seleção dos artigos apenas pelo título, na terceira etapa fizemos a seleção dos artigos pelo *abstract*, na quarta etapa selecionamos os artigos pela introdução e conclusão e por fim na quinta etapa executamos uma leitura completa nos artigos selecionados.

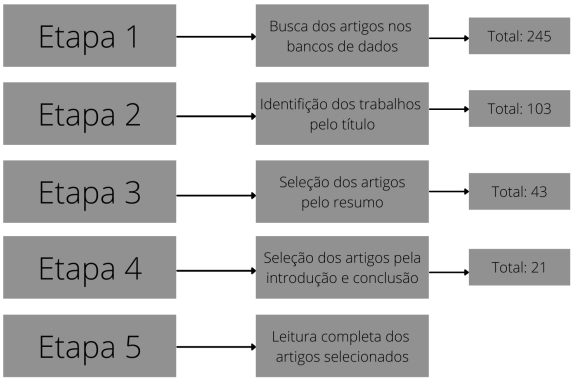

Figure 4: Etapas do MSL Fonte: Autor

3.7.3 *Artigos Selecionados, Rejeitados, Aceitos e Duplicados*

Na figura 5 podemos ver a quantidade total de artigos encontrados em cada banco de dados, a quantidade de artigos duplicados, rejeitados e aceitos no **MSL**.

4. RESULTADOS

Nesta seção apresentaremos os 22 artigos aceitos para esse **MSL**, juntamente com a metodologia utilizada para avaliar os artigos,assim como a relevância de cada para o desenvolvimento e conclusão.

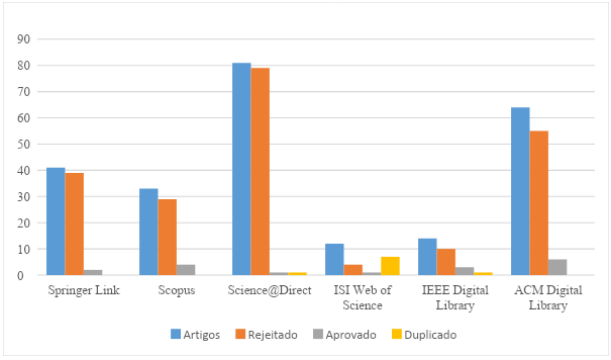

Figure 5: Artigos Selecionados por Banco de Dados Fonte: www.parsif.al

4.1 Qualidade dos Artigos

4.1.1 *Citações*

Durante o processo de avaliação dos artigos no **MSL** mantemos o número de citações dos principais artigos, os aceitos, para avaliação do impacto que estes artigos tiveram nas pesquisas da área de software embarcado durante as últimas décadas. Temos abaixo a tabela 3 contendo os artigos selecionados para esse **MSL** com o total de citações até 2020.

Table 3: Contagem de Artigos Aceitos por Citação

Artigo	Citação
Advanced ECU software development method for fuel cell systems	2
A hardware/software approach to detect memory corruptions in embedded systems	1
Comparison of component frameworks for real-time embedded systems	7
Concurrency Emulation and Analysis of Parallel Applications for Multi-Processor System-on-Chip Co-Design	3
Debugging protocol for remote cross development environment	5
Reconciling run-time evolution and resource-constrained embedded systems through a component-based development framework	3
Research of "Stub" remote debugging technique	2
Teaching embedded software development utilising QNX and Qt with an automotive-themed coursework application	0
Virtual execution environment for real-time TDL components	1
Whole Program Compilation for Embedded Software: The ADSL Experiment	1

Temos abaixo a tabela 4 contendo os artigos aceitos para esse **MSL** com o total de citações até 2020.

Na figura 6 podemos ver um gráfico com a contagem de artigos e citações, onde no eixo vertical temos o somatório de artigos e no eixo horizontal o somatório de citações.

4.1.2 *Pontuação*

Utilizamos as seguintes perguntas para classificar os artigos a seguir:

Table 4: Contagem de Artigos Aceitos por Citação

Artigo	Citação
An RTOS API Translator for Model-Driven Embedded Software Development	3
Code Generation from Formal Models for Automatic RTOS Portability	2
Cross-Kernel Control-Flow–Graph Analysis for Event-Driven Real-Time Systems	12
Embedded software - Issues and challenges	2
Global Optimization of Fixed-Priority Real-Time Systems by RTOS-Aware Control-Flow Analysis	8
Implementation Synthesis of Embedded Software under the Group-Based Scheduling Model	1
Model-driven development of RTOS-based embedded software	0
Optimizing Resource Usage in Component-Based Real-Time Systems	14
RTOS modeling in SystemC for real-time embedded SW simulation: A POSIX model	31
RTOS Scheduling in Transaction Level Models	21
RTOS support in C-language toolchains	0
The wild west: Conquest of complex hardware-dependent software design	1

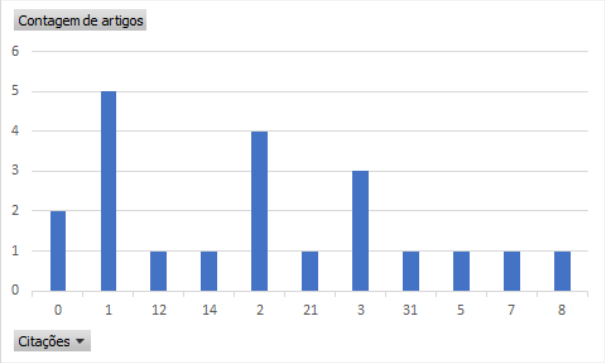

Figure 6: Artigos Selecionados por Citação Fonte: Autor

- Artigo que apresente características de sistema operacional embarcado.
- Artigo que apresenta discussão de sistema operacional embarcado.
- Artigo que apresenta discussão sobre uso **RTOS** em desenvolvimento de sistema embarcado.
- Artigo que estuda desvantagens do uso de **RTOS** em desenvolvimento de software embarcado.
- Artigo que apresenta vantagens do uso de **RTOS** em desenvolvimento de software embarcado.
- Artigo que apresenta comparação entre sistemas embarcados que utilizam um **RTOS** e sistemas embarcados que não utilizam um **RTOS**.

E utilizamos para responder cada uma das perguntas acima as seguintes repostas com a pontuação máxima a ser atribuída para cada uma:

- Sim (A ser utilizada quando o artigo responde a pergunta) (1.0).
- Parcialmente (A ser utilizada quando o artigo responde de forma parcial a pergunta)(0.5).
- Não (A ser utilizada quando o artigo não atende a pergunta)(0.0).

Sendo assim o somatório das notas atribuídas a cada pergunta que o artigo atende pode ser no máximo a pontuação 7, como por exemplo quando o artigo responde sim a todas as perguntas qualificatórias. Iremos utilizar a pontuação 4,5 para separar as pesquisas de maior impacto das de menor impacto para a esse **MSL**.

Os artigos selecionados com pontuação inferior a 4,5 podem ser visto na tabela 5 a seguir:

Table 5: Artigos Selecionados com Pontuação

Ano	Título	Nota
2005	Advanced ECU software development method for fuel cell systems.	2.5
2010	A hardware/software approach to detect memory corruptions in embedded systems.	2.0
2014	Comparison of component frameworks for real-time embedded systems.	4.0
2008	Concurrency Emulation and Analysis of Parallel Applications for Multi-Processor System-on-Chip Co-Design.	3.0
2000	Debugging protocol for remote cross development environment.	3.0
2013	Reconciling run-time evolution and resource-constrained embedded systems through a component-based development framework.	4.0
2009	Research of “Stub” remote debugging technique.	3.0
2014	Teaching embedded software development utilising QNX and Qt with an automotive-themed coursework application.	1.0
2007	Virtual execution environment for real-time TDL components.	4.0
2001	Whole Program Compilation for Embedded Software: The ADSL Experiment.	4.0

Esses arquivos citados na tabela 5 serão arquivos com menor impacto, porém, ainda serão utilizados para comparações neste **MSL**.

Os artigos selecionados com pontuação superior a 4,5 podem ser visto na tabela 6:

4.2 Resumo dos artigos

- *Advanced **ECU** software development method for fuel cell systems.*
  - Este artigo mostra o desenvolvimento de uma unidade de controle de eletrônica e um célula de combustível de um veículo. O sistema operacional OSEK **RTOS** foi utilizado com sucesso para desenvolver o sistema eletrônico do carro utilizado um Motorola PowerPC555. Mostra um comparativo do código, evidenciando uma redução em tamanho e complexidade. [28]

Table 6: Artigos Aceitos com Pontuação

Ano	Título	Nota
2006	An RTOS API Translator for Model-Driven Embedded Software Development.	6.0
2019	Code Generation from Formal Models for Automatic RTOS Portability.	6.0
2015	Cross-Kernel Control-Flow–Graph Analysis for Event-Driven Real-Time Systems.	5.5
2010	Embedded software - Issues and challenges.	5.0
2017	Global Optimization of Fixed-Priority Real-Time Systems by RTOS-Aware Control-Flow Analysis.	4.5
2006	Implementation Synthesis of Embedded Software under the Group-Based Scheduling Model.	5.0
2006	Model-driven development of RTOS-based embedded software.	5.5
2005	Optimizing Resource Usage in Component-Based Real-Time Systems.	5.5
2005	RTOS modeling in SystemC for real-time embedded SW simulation: A POSIX model.	5.5
2003	RTOS Scheduling in Transaction Level Models.	5.0
2017	RTOS support in C-language toolchains.	5.0
2009	The wild west: Conquest of complex hardware-dependent software design.	5.0

- *A hardware/software approach to detect memory corruptions in embedded systems.*
  - Este artigo demonstra uma implementação para detectar erros de memória corrompida em um sistema embarcado utilizando um **RTOS**. [21]
- *An **RTOS API** Translator for Model-Driven Embedded Software Development.*
  - Este artigo apresenta o desenvolvimento orientado a modelo de que pode produzir códigos específicos pra um sistema embarcado utilizando um **RTOS**, uma *Application Programming Interface* (**API**) **RTOS** genérica capaz capturar e de traduzir chamadas típicas de um **RTOS**. E o desenvolvimento de uma *Automated Transformation Tool* (**ATT**) que produz código transformando a **API** genérica em uma **API** específica para **RTOS**. [14]
- *Code Generation from Formal Models for Automatic **RTOS** Portability.*
  - Este artigo demonstra um gerador de código para traduzir códigos de um sistema embarcado para um sistema embarcado baseado em um **RTOS**, automatizando a portabilidade de partes do sistema operacional *(context switching, memory management, security aspects, etc,).* [11]
- *Comparison of component frameworks for real-time embedded systems.*
  - Survey, mostra uma comparação entre componentes de frameworks que podemos utilizar em sistemas embarcados utilizando um **RTOS**. Com foco em componentes relacionados a conectividade e aplicabilidade em todo ciclo de vida do sistema embarcado. [24]
- *Concurrency Emulation and Analysis of Parallel Applications for Multi-Processor System-on-Chip Co-Design.*
  - Este artigo demonstra a utilização de paralelismo em um sistema embarcado que conta com a funcionalidade de multi-core, emulando as primitivas do sistema operacional como *task creation, scheduling, synchronization* e etc, esta emulação garante ainda a compatibilidade independente do sistema operacional. [2]
- *Cross-Kernel Control-Flow–Graph Analysis for Event-Driven Real-Time Systems.*
  - O artigo incorpora a semântica de um **RTOS** em um controle de fluxo entre *kernels*, assim criando uma visão global de todas as possibilidades de execução em um **RTOS**. [6]
- *Debugging protocol for remote cross development environment.*
  - O artigo demonstra algumas ferramentas para o desenvolvimento de um sistema embarcado, especificamente utilizando um **RTOS**, demonstra por exemplo, monitor de recursos, *debugging shell* interativo e *binary utilities.* [27]
- *Embedded software - Issues and challenges.*
  - O Artigo mostra a utilização de algumas ferramentas no desenvolvimento de sistema embarcados, demonstra *desingning*, documentação, análise, compilação, codificação, teste e optimização utilizando **RTOS**. [23]
- *Global Optimization of Fixed-Priority Real-Time Systems by **RTOS**-Aware Control-Flow Analysis.*
  - Este artigo combina a semântica de um **RTOS** com *deterministic fixed-priority scheduling* e *flow-sensitive compiler optimizations.* [7]
- *Implementation Synthesis of Embedded Software under the Group-Based Scheduling Model.*
  - Este artigo apresenta uma implementação de um método para sistemas operacionais terem suporte ao group-based scheduling model. Assim implementando task grouping, priority assignment e task generation satisfazendo requerimentos de um **RTOS**. [30]
- *Optimizing Resource Usage in Component-Based Real-Time Systems.*
  - Este artigo demonstra como alocar componentes em um gerenciador de tarefas de tempo real, mostra como reduzir o custo de processamento e memoria em 48% e 32% respectivamente. [10]
- *Reconciling run-time evolution and resource-constrained embedded systems through a component-based development framework.*

  - Este artigo mostra a evolução dos sistemas embarcados e evolução do tempo de execução dos softwares, propõe uma implementação genérica utilizando *component-based, execution time evolution infrastructure*, para conciliar a as alternativas evolutivas e requerimentos de desempenho. [22]

- *Research of "Stub" remote debugging technique.*
  - Este artigo analisa e estuda técnicas stub mode para debugging usando *trace object* e *exception handlers* em um sistema embarcado utilizando um **RTOS**. [13]

- ***RTOS** modeling in SystemC for real-time embedded SW simulation: A **POSIX** model.*
  - Neste artigo, são propostas técnicas para modelar com precisão as funcionalidades **RTOS** detalhada no topo do kernel de execução **SystemC**. O modelo permite *timed-simulation* no **SystemC**. A tecnologia de simulação foi aplicada ao desenvolvimento de uma biblioteca de simulação *Portable Operating System Interface* (**POSIX**) de alto nível. [25]

- ***RTOS** Scheduling in Transaction Level Models.*
  - Este artigo demonstra o problema imposto pelo modelo **RTOS** para converter uma *Transaction Level Models* (**TLM**) para uma **TLM** com agendamento suportado pelo RTOS. [29]

- ***RTOS** support in C-language toolchains.*
  - Neste artigo, são analisadas as relações mútuas entre vários mecanismos-chave internos a uma *toolchain* da linguagem C para **RTOS**. A discussão é realizada com referência a uma *toolchain* de código aberto baseada em (**GCC**) para a linguagem C e o FreeRTOS. [3]

- *Teaching embedded software development utilising QNX and Qt with an automotive-themed coursework application.*
  - Artigo demonstra a utilização do **OS QNX RTOS** integrado com o QT para interações humano máquina. [1]

- *The wild west: Conquest of complex hardware-dependent software design.*
  - Artigo demonstra algumas dificuldades para o desenvolvimento de sistemas embarcados heterogêneos, principalmente pela a vasta gama de processadores e plataformas atualmente. [12]

- *Virtual execution environment for real-time **TDL** components.*
  - Mostra como a execução virtualizada é aplicada ao paralelismo e ao tempo de execução em tempo real utilizando **TDL** como dependências de dados. [9]

- *Whole Program Compilation for Embedded Software: The ADSL Experiment.*
  - O artigo demonstra a implementação de um modem **ADSL** utilizando alguns conceitos pouco explorados em sistemas embarcados, como por exemplo, reuso de código, módulos, conceitos de orientação a objetos e a linguagem C++. [5]

5. CONCLUSÕES

Apesar da pesquisa resultar muitos artigos que discutem o uso de **RTOS** em sistemas embarcados, não foi possível inferir sobre o real ganho em utilizar um **RTOS** em sistemas embarcados. Alguns discorrem sobre otimizações, técnicas secundárias ou alguma modificação em um **RTOS** que aumentam o desempenho do sistema embarcado (como por exemplo o artigo Optimizing Resource Usage in Component-Based Real-Time Systems. [10]), porém não foi encontrado um comparativo, de um mesmo sistema embarcado, com e sem o **RTOS** ou algum relato que nos mostre, de forma clara, o real ganho, em um mesmo sistema, que utiliza ou não um **RTOS**, indicando uma redução ou aumento no tamanho, ou na complexidade do código. Não foi encontrado evidências que apontem o real ganho na utilização de **RTOS** em um sistema embarcado, durante as buscas encontramos inúmeras publicações, mas dentro do escopo deste **MSL**, restavam poucos resultados. Após classificar os artigos, optamos por mantermos até mesmo os de baixa pontuação, esses serão utilizados como referência e para comparações durante este projeto de pesquisa. Os estudos de maior pontuação terão maior impacto no projeto. Tivemos alguns bons artigos para estudo, por exemplo, *Code Generation from Formal Models for Automatic **RTOS** Portability. [11]* ou no *An **RTOS** Application Programming Interface **API** Translator for Model-Driven Embedded Software Development.* [14], que utiliza um gerador de código para traduzir códigos para a utilização em um **RTOS**. Outros resultados que falam de portabilidade, não utilizam uma ferramenta similar (outro exemplo, ***RTOS** modeling in SystemC for real-time embedded SW simulation: A Portable Operating System Interface (POSIX) model.* [25]) que utiliza uma técnica de modelar a funcionalidades de um RTOS. Podemos analisar um pouco do ganho de cada técnica: que apesar de ser vantajosa no quesito portabilidade (pois o sistema funciona de forma automática sem ter que realizar a portabilidade) ele tem um desempenho um pouco prejudicado em relação a outros que fazem a portabilidade do **OS** de forma manual.

5.1 Respostas Para as Perguntas de Pesquisa

1. Quais são os sistemas operacionais embarcados mais utilizados no desenvolvimento de software embarcado?
   - Durante o **MSL** achamos referências a alguns **OS** – como por exemplo, no artigo ***RTOS** support in C-language toolchains.* [3] que referencia o FreeRTOS - O **OS RTOS** da **TI** não foi encontrado nos artigos selecionados, talvez por ser um **OS** de uso exclusivo nos dispositivos da própria **TI**, com base nos artigos encontrados e selecionados é possível identificar que o FreeRTOS é o sistema mais citado. Foi encontrada durante as pesquisas uma *market survey* de 2017 [26] onde mostra a tendencia de sistema operacional atualmente em uso em projetos como pode ser visto na tabela 7 e também de sistema operacional que as principais

empresas do mercado estão consideram utilizar nos próximos 12 messes, como pode ser visto na tabela 8.

Table 7: Sistema Operacional Embarcado Uso Atual

Sistema	Uso
Embedded Linux	22%
FreeRTOS	20%
In-House/Custom	19%
Android	13%
Debian (Linux)	13%
Ubuntu	11%
Vs Embedded 7/Standard	8%
Texas Instruments RTOS	5%
Micrium (uC/OS-III)	5%
Microsoft (Windows 7 Compact or Earlier) (uC/OS-II)	5%

Table 8: Projeção de Uso de Sistema Operacional Embarcado Nos Próximos 12 Messes

Sistema	Projeção
FreeRTOS	28%
Embedded Linux	27%
In-House/Custom	19%
Android	17%
Debian (Linux)	12%
Ubuntu	11%
Micrium (uC/OS-III)	9%
Texas Instruments RTOS	8%
Micrium (uC/OS-II)	6%
Vs Embedded 7/Standard	6%

Como pode ser visto nas duas tabelas 7 e 8 a lista de sistema passa por poucas mudanças entre os sistemas operacionais atualmente em uso e da projeção de uso durante o ano de 2018.

2. Quais a vantagens do uso dos sistemas operacionais embarcados?

   - Faltam artigos que respondem de forma clara, alguns artigos - como por exemplo, *The wild west: Conquest of complex hardware-dependent software design.* [12] e *Embedded software - Issues and challenges.* [23] - mostram alguns benefícios do sistema operacional genérico no desenvolvimento de software embarcado, como por exemplo: *performance*, modularidade, eficiência e portabilidade. No entanto os dados colhidos com esse **MSL** não são o suficiente para afirmar o real beneficio de se utilizar um **OS** em um sistema embarcado, pois nenhum dos artigos mostra de forma clara os benefícios e malefícios usando como base o mesmo sistema embarcado.

3. Quais as desvantagens ou dificuldades?

   - Não tivemos resultados que possam ajudar a responder esta pergunta, pois em nenhum dos artigos foi encontrado algum comparativo que mostre de forma clara algum comparativo relacionado à complexidade do código ou algum teste relacionado a desempenho em sistema com **OS** e sem **OS**, porem alguns artigos citam como algumas vantagens a simplicidade de se criar códigos modulares utilizando o OS como base, facilidade de implementação do código e ainda facilidades para lidar com o gerenciamento de tempo do sistema embarcado ao se utilizar um **RTOS**.

4. Quais são as principais características dos sistemas operacionais embarcados?

   - Com base nos artigos é possível notar que as características com maior impacto em sistemas embarcados são:
     (a) Kernel
     (b) Device Drivers and Board Initialization
     (c) Networking
     (d) Wireless Connectivity
     (e) File System
     (f) Control-Flow
     (g) Paralelismo
     (h) Concorrência
     (i) Instrumentation
     (j) run-time behavior
     (k) system-level analysis tools
     (l) context-switching

   Na figura 7 podemos ver um gráfico que representa em quantos artigos cada uma das características acima foram citadas.

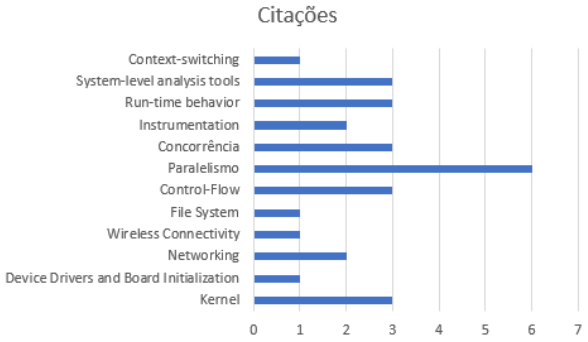

Figure 7: Características Citadas Fonte: Autor

5. Quais os impactos de usar sistemas operacionais no desenvolvimento de software embarcado?

   - Os artigos que demonstram essa abordagem nos indicam que em primeiro momento há um impacto negativo para a performance. O uso de **OS** faz com que o sistema tenha uma perda de performance, e em segundo lugar nos relatam uma melhora significativa para o desenvolvimento, transformando-o em uma tarefa menos complexa, pois as principais funções relativas ao **OS** já estão implementadas. [10]

6. AGRADECIMENTOS

A todos os meus familiares, amigos e a UNIFESP por todo o apoio.

7. REFERENCES

[1] P. Barrie and G. Morison. Teaching embedded software development utilising qnx and qt with an automotive-themed coursework application. In *2014 6th European Embedded Design in Education and Research Conference (EDERC)*, pages 6–10, 2014.

[2] G. Beltrame, L. Fossati, and D. Sciuto. Concurrency emulation and analysis of parallel applications for multi-processor system-on-chip co-design. In *Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, page 7–12. Association for Computing Machinery, 2008.

[3] I. C. Bertolotti. Rtos support in c-language toolchains. In *2017 IEEE International Conference on Industrial Technology (ICIT)*, pages 1328–1333, 2017.

[4] J. Biolchini, P. G. Mian, A. C. C. Natali, and G. H. Travassos. Systematic review in software engineering". technical report. *PESC – COPPE/UFRJ*, 679/05(1):1–31, 2005.

[5] A. J. Cockx. Whole program compilation for embedded software: The adsl experiment. In *Proceedings of the Ninth International Symposium on Hardware/Software Codesign*, page 214–218. Association for Computing Machinery, 2001.

[6] C. Dietrich, M. Hoffmann, and D. Lohmann. Cross-kernel control-flow–graph analysis for event-driven real-time systems. In *Proceedings of the 16th ACM SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems 2015 CD-ROM*, LCTES'15, New York, NY, USA, 2015. Association for Computing Machinery.

[7] C. Dietrich, M. Hoffmann, and D. Lohmann. Global optimization of fixed-priority real-time systems by rtos-aware control-flow analysis. *ACM Trans. Embed. Comput. Syst.*, Jan. 2017.

[8] T. Dyba, B. Kitchenham, and M. Jorgensen. Evidence-based software engineering for practitioners. *IEEE Software*, 22(1):58–65, 2005.

[9] C. Farcas and W. Pree. Virtual execution environment for real-time tdl components. In *2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007)*, pages 93–100, 2007.

[10] J. Fredriksson, K. Sandström, and M. Åkerholm. Optimizing resource usage in component-based real-time systems. In G. T. Heineman, I. Crnkovic, H. W. Schmidt, J. A. Stafford, C. Szyperski, and K. Wallnau, editors, *Component-Based Software Engineering*, pages 49–65. Springer Berlin Heidelberg, 2005.

[11] R. M. Gomes and M. Baunach. Code generation from formal models for automatic rtos portability. In *2019 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 271–272, 2019.

[12] Hiroyuki Yagi, W. Rosenstiel, J. Engblom, J. Andrews, K. Vissers, and M. Serughetti. The wild west: Conquest of complex hardware-dependent software design. In *2009 46th ACM/IEEE Design Automation Conference*, pages 878–879, 2009.

[13] Hongwei Li, Yaping Xu, Fangsheng Wu, and Changhong Yin. Research of "stub" remote debugging technique. In *2009 4th International Conference on Computer Science Education*, pages 990–994, 2009.

[14] Ji Chan Maeng, Jong-Hyuk Kim, and Minsoo Ryu. An rtos api translator for model-driven embedded software development. In *12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)*, pages 363–367, 2006.

[15] H. Joe and H. Kim. Effects of dynamic isolation for full virtualized rtos and gpos guests. *Future Gener. Comput. Syst.*, 70(C):26–41, May 2017.

[16] B. Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele Univ.*, 33, 08 2004.

[17] B. A. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 07 2007.

[18] J. Lin, A. Cheng, D. Steel, M. Wu, and N. Sun. Scheduling mixed-criticality real-time tasks in a fault-tolerant system. *International Journal of Embedded and Real-Time Communication Systems*, 6:65–86, 04 2015.

[19] S. N. Mafra and G. H. Travassos. Estudos primários e secundários apoiando a busca por evidências em engenharia de software. *PESC – COPPE/UFRJ*, 687/06(1):1–33, 2006.

[20] L. E. G. Martins and T. de Oliveira. A case study using a protocol to derive safety functional requirements from fault tree analysis. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pages 412–419, 2014.

[21] Nam Ho and A. Dinh-Duc. A hardware/software approach to detect memory corruptions in embedded systems. In *The 2010 International Conference on Advanced Technologies for Communications*, pages 285–290, 2010.

[22] J. F. Navas, J.-P. Babau, and J. Pulou. Reconciling run-time evolution and resource-constrained embedded systems through a component-based development framework. *Science of Computer Programming*, pages 1073 – 1098, 2013.

[23] S. Patil and L. Kapaleshwari. Embedded software - issues and challenges. In *SAE Technical Paper*. SAE International, 04 2010.

[24] T. Pop, P. Hnětynka, P. Hošek, M. Malohlava, and T. Bureš. Comparison of component frameworks for real-time embedded systems. *Knowl Inf Syst*, 40(1):127–170, July 2014.

[25] H. Posadas, J. A. Adamez, E. Villar, F. Blasco, and F. Escuder. RTOS modeling in SystemC for real-time embedded SW simulation: A POSIX model. *Design Automation for Embedded Systems*, 10(4):209–227, Dec. 2005.

[26] S. Rambo. 2017 embedded market survey. *Embedded, ASPENCORE*, 33:102, 12 2017.

[27] S. Son, C. Lim, and N.-N. Kim. Debugging protocol for remote cross development environment. In *Proceedings Seventh International Conference on Real-Time Computing Systems and Applications*, pages 394–398, 2000.

[28] S. Tian, Y. Liu, W. Xia, J. Li, and M. Yang.

Advanced ecu software development method for fuel cell systems. *Tsinghua Science & Technology*, pages 610 – 617, 2005.

[29] H. Yu, A. Gerstlauer, and D. Gajski. Rtos scheduling in transaction level models. In *Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES+ISSS '03, page 31–36, New York, NY, USA, 2003. Association for Computing Machinery.

[30] Zhigang Gao and Zhaohui Wu. Implementation synthesis of embedded software under the group-based scheduling model. In *12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)*, pages 190–198, 2006.