UNIVERSIDADE FEDERAL DO CEARÁ

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

WENDRIL AVILA GADELHA FERREIRA

CAT FRAMEWORK: UM FRAMEWORK MAPE-K PARA COMPUTAÇÃO AUTÔNOMA EM SISTEMAS EMBARCADOS

RUSSAS

2025

WENDRIL AVILA GADELHA FERREIRA

CAT FRAMEWORK: UM FRAMEWORK MAPE-K PARA COMPUTAÇÃO AUTÔNOMA EM SISTEMAS EMBARCADOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientador: Prof. Dr. Reuber Regis De Melo

RUSSAS

2025

WENDRIL AVILA GADELHA FERREIRA

CAT FRAMEWORK: UM FRAMEWORK MAPE-K PARA COMPUTAÇÃO AUTÔNOMA EM SISTEMAS EMBARCADOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em: 11/08/2025

BANCA EXAMINADORA

Prof. Dr. Reuber Regis De Melo (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Pablo Luiz Braga Soares
Universidade Federal do Ceará (UFC)

Prof. Dr. Cenez Araújo De Rezende
Universidade Federal do Ceará (UFC)

A todos aqueles que, ainda nutridos pela humanidade, acreditam que a computação, livre da ganância e guiada pelas necessidades dos povos, pode ser um meio para transformar o mundo em um lugar mais justo. Que a tecnologia, em vossas mãos, sirva ao bem comum e não ao lucro.

AGRADECIMENTOS

Ao Prof. Dr. Reuber Regis De Melo por me orientar em meu trabalho de conclusão de curso.

Ao Prof. Dr. Pablo Luiz Braga Soares e ao Prof. Dr. Cenez Araújo de Rezende pela disponibilidade de compor a banca examinadora deste trabalho.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

Aos amigos de jornada, Micael de Araújo Lima e Johnath Costa, pela companhia e parceria ao longo de tantos anos, desde os tempos do ensino fundamental até este momento marcante da conclusão do curso. Compartilhar esse caminho com vocês tornou a caminhada mais leve e significativa.

Aos meus pais, verdadeiros pilares da minha vida. A vocês devo tudo o que sou e tudo o que ainda serei. Vosso exemplo, apoio incondicional e amor foram fundamentais para que eu chegasse até aqui.

A minha noiva e, em breve esposa, Gabriele Fernandes de Lima, pelo imenso apoio prestado ao longo de anos do curso de engenharia de software. Seu amor e carinho me motivaram a ir além das minhas próprias capacidades.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

"É preciso sonhar, mas com a condição de crer em nosso sonho. De examinar com atenção a vida real, de confrontar nossa observação com nosso sonho, de realizar escrupulosamente nossa fantasia. Sonhos, acredite neles."

(Vladimir Lenin)

RESUMO

O aumento da complexidade no desenvolvimento de software e a crescente demanda por sistemas inteligentes e autoadaptativos impulsionam a necessidade de modelos arquiteturais que padronizem e modularizem sua implementação. Nesse contexto, a IBM propôs o modelo arquitetural MAPE-K, baseado em um ciclo de controle fechado estruturado em quatro etapas principais: monitoramento, análise, planejamento e execução, todas interligadas a uma base de conhecimento para aprimorar a tomada de decisões futuras. Este trabalho propõe a criação do Cat Framework, um framework de código aberto para estruturar e modularizar o MAPE-K em componentes reutilizáveis e integráveis a sistemas embarcados baseados em microcontroladores programados em C++. O objetivo é padronizar a implementação do MAPE-K nesse contexto, facilitando sua compreensão e adoção por desenvolvedores, além de fornecer uma interface de linha de comando para simplificar a compilação e a configuração dos projetos. Espera-se que o framework contribua para o desenvolvimento de sistemas embarcados mais eficientes e adaptáveis.

Palavras-chave: arquitetura; MAPE-K; sistemas autônomos; sistemas embarcados; framework.

ABSTRACT

The increasing complexity in software development and the growing demand for intelligent and self-adaptive systems drive the need for architectural models that standardize and modularize their implementation. In this context, IBM proposed the MAPE-K architectural model, based on a closed control cycle structured in four main stages: monitoring, analysis, planning and execution, all interconnected to a knowledge base to improve future decision-making. This work proposes the creation of the Cat Framework, an open-source framework to structure and modularize MAPE-K into reusable components that can be integrated into embedded systems based on microcontrollers programmed in C++. The objective is to standardize the implementation of MAPE-K in this context, facilitating its understanding and adoption by developers, in addition to providing a command-line interface to simplify the compilation and configuration of projects. The framework is expected to contribute to the development of more efficient and adaptable embedded systems.

Keywords: architecture; MAPE-K; autonomous systems; embedded systems; framework.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CLI	*Command Line Interface*
GoF	Gang Of Four
IoT	Internet of Things
MAPE	Monitor, Analyze, Plan, Execute

SUMÁRIO

1 INTRODUÇÃO

O avanço da tecnologia da informação tem impulsionado a complexidade dos sistemas computacionais, tornando seu desenvolvimento, configuração e manutenção desafios cada vez maiores. Essa crescente complexidade exige que os sistemas sejam capazes de adaptar-se dinamicamente a diferentes contextos operacionais, reduzindo a necessidade de intervenção humana. Nesse cenário, em 2001, Paul Horn, vice-diretor de pesquisa da IBM, introduziu o conceito de computação autônoma, inspirado no sistema nervoso autônomo humano. A proposta visa a criação de sistemas que gerenciem a si mesmos, otimizando o uso de recursos computacionais e respondendo automaticamente a mudanças em tempo real (HORN, 2001; FOX, 2010).

A partir dessa ideia, a IBM desenvolveu o modelo arquitetural MAPE-K, que estrutura sistemas autônomos a partir de um ciclo de controle fechado composto por quatro etapas principais: Monitoramento, Análise, Planejamento e Execução, todas sustentadas por uma base de conhecimento que armazena informações para apoiar decisões futuras (KEPHART; CHESS, 2003). Esse modelo tem sido aplicado com sucesso em áreas como automação industrial, robótica e desenvolvimento de software (MALBURG *et al.*, 2023; ROMERO-GARCéS *et al.*, 2022; HALIMA *et al.*, 2023).

Embora o modelo MAPE-K seja robusto, sua aplicação em sistemas embarcados ainda enfrenta desafios. Abordagens como Nakagawa *et al.* (2022) mostram frameworks modulares com componentes bem definidos para cada fase do ciclo MAPE-K, aplicados em cenários reais de robótica e IoT. Por outro lado, Mecibah e Boutekkouk (2024) destaca a ausência de padronização e interoperabilidade entre plataformas embarcadas, observando que poucos frameworks cobrem todas as etapas do ciclo de forma integrada.

Diante desse cenário, este trabalho propõe o *Cat Framework*, um *framework* de código aberto voltado para a implementação modular e padronizada do ciclo MAPE-K em sistemas embarcados, com suporte a microcontroladores programados em C++. A proposta visa oferecer uma solução reutilizável, documentada e de fácil integração, promovendo maior eficiência, flexibilidade e acessibilidade no desenvolvimento de sistemas autônomos embarcados.

1.1 Motivação

A crescente demanda por sistemas inteligentes e adaptativos exige soluções que facilitem a aplicação de arquiteturas autônomas em contextos com recursos limitados, como os sistemas embarcados. Apesar do potencial do MAPE-K, sua adoção nesse cenário ainda é incipiente devido à falta de frameworks que conciliem modularidade, facilidade de uso e compatibilidade com plataformas de baixo custo.

Além disso, muitos projetos carecem de ferramentas que auxiliem na padronização de código, na documentação e na manutenção, dificultando a curva de aprendizado para novos desenvolvedores. O Cat Framework busca preencher essa lacuna, oferecendo um modelo claro e estruturado para a implementação do MAPE-K nesse contexto.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um *framework* modular baseado no modelo MAPE-K para sistemas embarcados, promovendo padronização, reutilização de componentes e integração com microcontroladores programados em C++.

1.2.2 Objetivos Específicos

Para atingir o objetivo geral, este trabalho busca:

- Analisar *frameworks* existentes para a implementação do MAPE-K e avaliar sua aplicabilidade em sistemas embarcados;
- Definir e modelar um novo *framework* componentizado e orientado a sistemas embarcados;
- Desenvolver uma interface de linha de comando (CLI) para facilitar a compilação e configuração de projetos;
- Implementar um caso de uso prático que demonstre a viabilidade do *framework*;
- Validar a solução por meio da análise dos resultados obtidos na implementação prática.

1.3 Organização do Trabalho

Este trabalho está estruturado em seis capítulos, organizados da seguinte forma:

- Capítulo 1 – Introdução: apresenta o contexto da pesquisa, a motivação para o desenvolvi-

mento do Cat Framework, os objetivos do trabalho e sua relevância no cenário de sistemas autônomos embarcados.

- Capítulo 2 – Fundamentação Teórica: discute os principais conceitos que embasam a proposta, como sistemas autônomos, sistemas embarcados, frameworks baseados em componentes, interfaces de linha de comando, padrões de projeto e a plataforma ESP-IDF.
- Capítulo 3 – Trabalhos Relacionados: analisa frameworks existentes com propostas semelhantes ao Cat Framework, comparando suas abordagens, limitações e contribuições.
- Capítulo 4 – Metodologia: descreve o processo de desenvolvimento do framework, incluindo o planejamento da arquitetura, a construção da interface de linha de comando, o experimento de validação e o cronograma de execução.
- Capítulo 5 – Resultados: apresenta a prova de conceito implementada com o Cat Framework, destacando a aplicação em um cenário real de monitoramento de redes Wi-Fi e os resultados obtidos.
- Capítulo 6 – Conclusões e Trabalhos Futuros: traz as considerações finais sobre o desenvolvimento do framework, suas limitações e possíveis direções para pesquisas futuras.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, será apresentado os principais termos e conceitos teóricos que fundamentam a proposta do framework.

2.1 Sistemas autônomos

Segundo Kephart e Chess (2003) em sua revisão a respeito da proposta de computação autônoma, ele define que sistemas autônomos são sistemas que gerenciam a si mesmos de acordo com o objetivo do administrador. Esse conceito surgiu em um manifesto publicado em 2001 por Paul Horn, vice diretor de pesquisa da IBM, como uma proposta para solucionar a crescente complexidade de administração de sistemas de alto nível. Horn destacou a necessidade de projetar e desenvolver sistemas computacionais capazes de operar de forma autônoma, ajustando-se a diferentes circunstâncias e otimizando o uso de seus recursos para lidar com as cargas de trabalho de maneira eficiente.

> It's time to design and build computing systems capable of running themselves, adjusting to varying circumstances, and preparing their resources to handle most efficiently the workloads we put upon them. These autonomic systems must anticipate needs and allow users to concentrate on what they want to accomplish rather than figuring how to rig the computing systems to get them there. (HORN, 2001).

Essa abordagem busca reduzir a necessidade de intervenção humana, permitindo que os usuários foquem nos resultados desejados, sem se preocupar com a configuração e o gerenciamento detalhado dos sistemas computacionais (KEPHART; CHESS, 2003).

Arquiteturalmente, sob essa perspectiva, um sistema autônomo é composto por um conjunto de elementos autônomos, onde cada elemento pode desempenhar ações no ambiente, servir como insumo para outros elementos e, ao mesmo tempo, gerenciar seu próprio estado interno e comportamento. Um elemento autônomo implementa um laço de controle inteligente que automatiza alguma função de gerenciamento e externaliza essa função de acordo com um comportamento definido pela interface de gerenciamento (COMPUTING *et al.*, 2006). Para que isso ocorra, é definido que este laço contenha as seguintes etapas:

- Monitoramento: Coleta, agrega e filtra dados provenientes de algum recurso interno ou externo.
- Analisador: Provê um mecânismo que correlaciona e modela situações complexas e determina se alguma ação é necessária.

- Planejador: É o objeto que projeta ações que devem ser tomadas para se atingir um objetivo.
- Executor: Responsável por executar as ações planejadas em um efetuador seja interno ou externo.

Todas essas etapas interagem com uma base de conhecimento, que pode ser implementada como um registro, dicionário, banco de dados ou outras estruturas de armazenamento. Essa base contém políticas, planos, detalhes do ambiente, dados processados pelas etapas e outras informações essenciais. Durante o ciclo, esses dados devem ser utilizados para aprimorar a detecção de mudanças, o planejamento de ações e a compreensão do ambiente, garantindo um processo mais eficiente e adaptável.

2.2 Sistemas embarcados

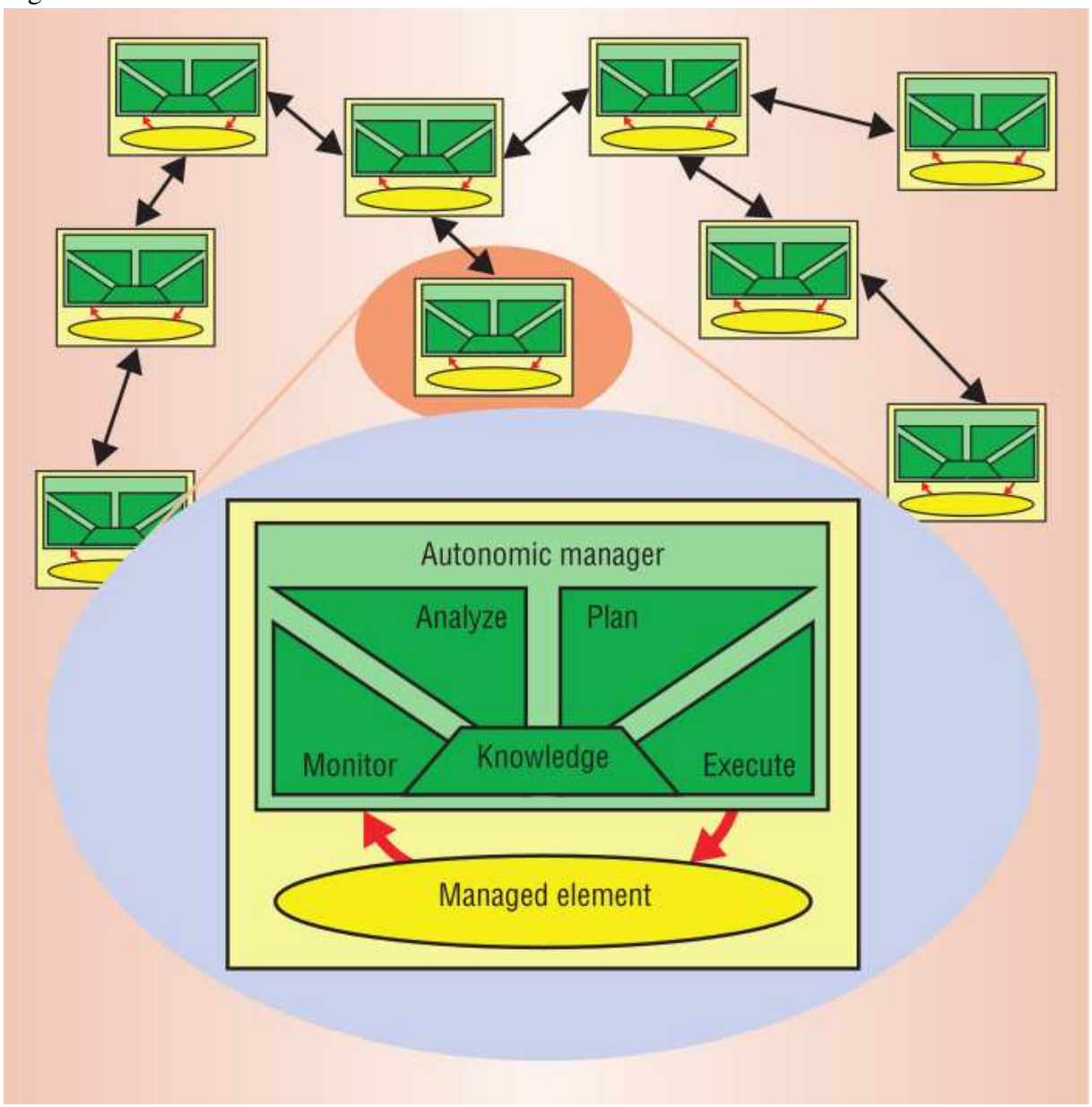
Seguindo na linha de construção do framework, é importante entender do que se trata um sistema embarcado. Segundo GANSSLE (2003), sistemas embarcados são uma combinação de hardware e software de computador e outras peças mecânicas ou componentes adicionais, projetados para desempenhar uma função específica de um sistema maior. Ao longo dos anos, esses sistemas ganharam significativa notoriedade devido ao baixo custo de produção e à popularização das placas de microcontroladores, como o Arduino e o ESP, que tornam a prototipagem de sistemas mais acessível e simplificada. (GANSSLE, 2003)

Muitas aplicações para essa categoria de sistemas têm surgido, como nas áreas de Internet das Coisas, robótica, indústria, entre outras. No entanto, a evolução do desenvolvimento de software nem sempre acompanha o ritmo do desenvolvimento de hardware. Nesse contexto, é necessário fornecer bibliotecas e frameworks de apoio que facilitem o desenvolvimento individual de atividades a nível de arquitetura de sistema de maneira que sejam compostos de maneira incremental. (WÄTZOLDT *et al.*, 2012)

2.3 Frameworks seriais e paralelos

No contexto dos estudos sobre frameworks, é comum distingui-los entre seriais e paralelos, conforme sua abordagem de execução. Frameworks seriais operam de forma sequencial, processando uma tarefa por vez, o que facilita o controle de fluxo e o gerenciamento de recursos, mas limita o desempenho em sistemas mais complexos. Já os frameworks paralelos

Figura 1 – Estrutura de um elemento autônomo sob um sistema autônomo.



Fonte: (KEPHART; CHESS, 2003).

são projetados para executar múltiplas tarefas simultaneamente, dividindo o trabalho entre diversos núcleos ou unidades de processamento, o que permite ganhos expressivos de desempenho, especialmente em aplicações de alta demanda (RATHOD *et al.*, 2014). Considerando essas definições e os requisitos da proposta deste trabalho, o Cat Framework se enquadra como um framework paralelo pois se utiliza de threads para gerar novos componentes.

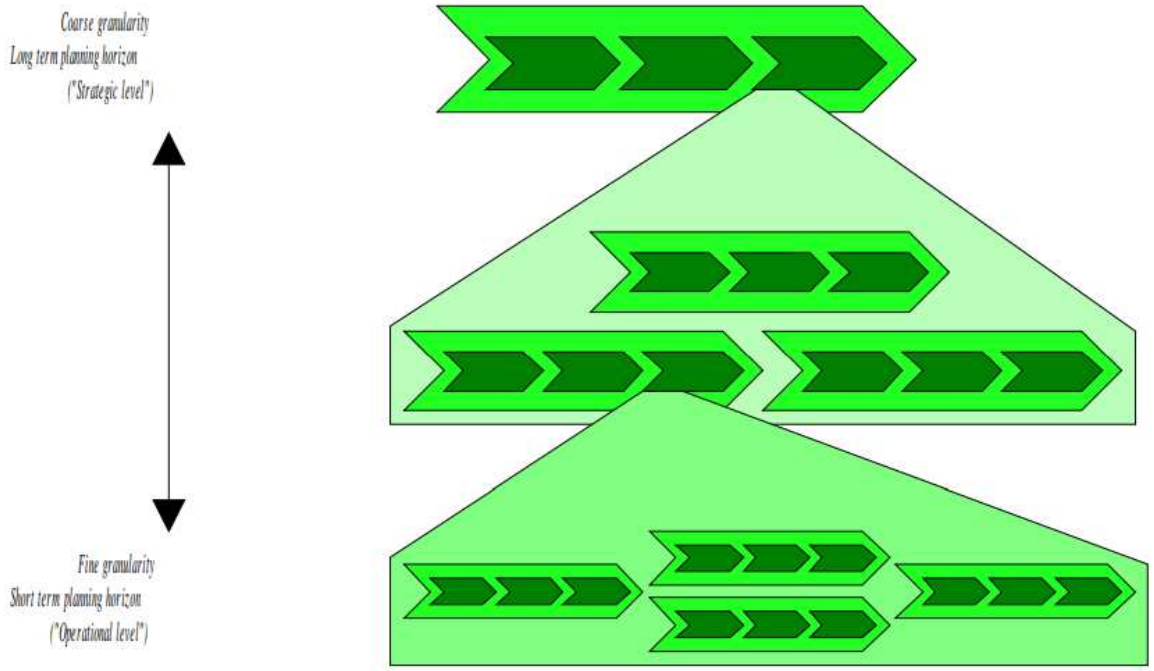
2.4 Framework baseado em componentes

Um framework é uma coleção de componentes de software reutilizáveis que tornam mais eficiente o desenvolvimento de novas aplicações. (Amazon Web Services, 2024b). Dentre

as diversas arquiteturas de frameworks, a baseada em componentes, no contexto deste artigo, se destaca pela sua eficácia pois eleva a padronização e o reuso de software a um nível de excelência na engenharia de software. Como resultado, frameworks que adotam esse paradigma tendem a alcançar sucesso na formação de comunidades de desenvolvedores engajadas e sustentáveis a longo prazo. (TIWARI; KUMAR, 2020).

Ao definir uma arquitetura componentizada, no contexto de sistemas embarcados, é essencial tomar preucações para não cair em caso de arquitetura monolítica (BARRETT; PROPER, 2021). Embora hierarquias de componentes possam surgir, é fundamental garantir que cada componente, independentemente de sua granularidade, possa operar de forma autônoma. Além disso, deve ser possível acoplar e desacoplar os componentes com facilidade, promovendo flexibilidade e modularidade no sistema. Sendo assim, adotar uma abordagem de modularização das camadas de componentes pode ser uma alternativa mais plausível. A Figura 2 ilustra melhor essa modularização de componentes.

Figura 2 – Granularidades e horizonte de planejamento.



Fonte: (BARRETT; PROPER, 2021).

Essa abordagem permite organizar os componentes em subgrupos hierárquicos, possibilitando a definição de prioridades, especialmente em sistemas com recursos computacionais limitados. No contexto de frameworks, um módulo pode conter componentes ou outros módulos relacionados a um determinado escopo de execução. Essa estrutura modular viabiliza o acoplamento e desacoplamento dinâmico de módulos inteiros com base em critérios como integridade

do sistema, prioridade de execução e eficiência operacional.

Além disso, é possível estabelecer metas gerais para um módulo, garantindo que seus componentes trabalhem em conjunto para alcançá-las. A criação e destruição de componentes dentro de um módulo podem ocorrer dinamicamente, conforme os critérios que serão abordados no capítulo 3 deste trabalho.

2.5 Interface de linha de comando

Uma interface de linha de comando (CLI) é um mecanismo de software que usado para interagir com o sistema operacional usando o teclado (Amazon Web Services, 2024a). No contexto de frameworks, muitos implementam ferramentas de interface de linha de comando (CLI) para oferecer recursos exclusivos e facilitar diversas tarefas, como iniciar um novo projeto, compilar o código, executar testes automatizados, entre outros. Uma CLI proporciona ao desenvolvedor maior praticidade e controle na administração do projeto, permitindo a execução de operações complexas de forma eficiente.

Interfaces de linha de comando no contexto de frameworks são uma ferramenta poderosa para a padronização de código do projeto. Ao iniciar um projeto utilizando-se de uma CLI, a ferramenta pode gerar os arquivos base de acordo com um padrão seguindo as melhores práticas da linguagem e da organização de pastas dos criadores do projeto.

2.6 Padrões de projeto

Os padrões de projeto de software foram introduzidos na literatura em meados da década de 90 por quatro projetistas, que segundo os mesmos são soluções gerais para problemas recorrentes no desenvolvimento de software. (GAMMA *et al.*, 1995). Os projetistas, entitulados por Gang Of Four (GoF), documentaram um total de 23 padrões de projetos que resolvem problemas recorrentes na criação de softwares que utilizam do paradigma de orientação a objetos.

Todos esses padrões são divididos em três categorias:

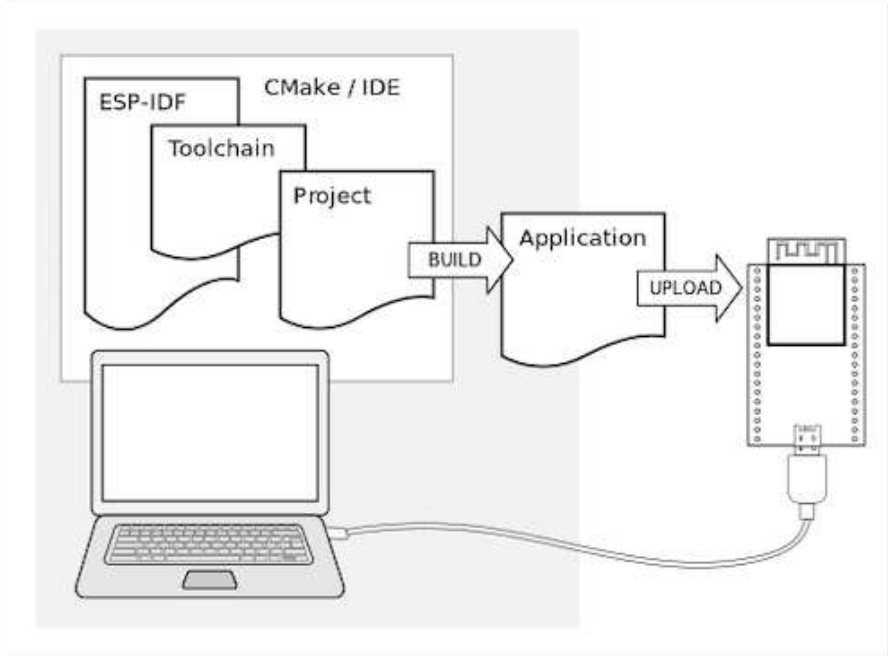
- **Padrões criacionais:** Úteis para lidar com a criação de objetos de modo flexível e reutilizável;
- **Padrões estruturais:** Compõe classes e objetos a fim de formar estruturas lógicas maiores;
- **Padrões comportamentais:** Definem como objetos interagem entre si.

No desenvolvimento de frameworks para linguagens orientadas a objetos, os padrões de projeto se tornaram um pilar essencial. De acordo com LANDIN et al. (1995), o design de um framework é difícil e todos problemas devem ser previamente previstos. Usar padrões de projetos no design de um framework fornece um vocabulário em comum entre os projetista e os desenvolvedores, reduz a complexidade do sistema uma vez que as abstrações foram previamente definidas, fornece a base de construção para abstrações mais complexas, fornece padronização na hierarquia de classes e consequentemente de todo o projeto. A reutilização de componentes é o ponto forte da estruturação desses frameworks com padrões de projeto. (LANDIN *et al.*, 1995)

2.7 ESP-IDF

O ESP-IDF (Espressif IoT Development Framework) é um framework de código aberto desenvolvido oficialmente pela Espressif Systems, voltado para o desenvolvimento de firmware para microcontroladores da família ESP32. Ele integra o kernel FreeRTOS, adaptado pela própria Espressif para oferecer suporte a múltiplos núcleos, multitarefa e gerenciamento eficiente de recursos. Além disso, o framework fornece uma ferramenta de linha de comando que facilita a configuração, compilação, monitoramento e implantação de projetos diretamente na placa (Espressif Systems, 2025). A Figura 3 ilustra as camadas e passos de desenvolvimento de *firmwares* para placas com microcontroladores ESP32.

Figura 3 – Visão geral do uso do ESP-IDF para desenvolvimento em placas com ESP32.



Fonte: (Espressif Systems, 2025).

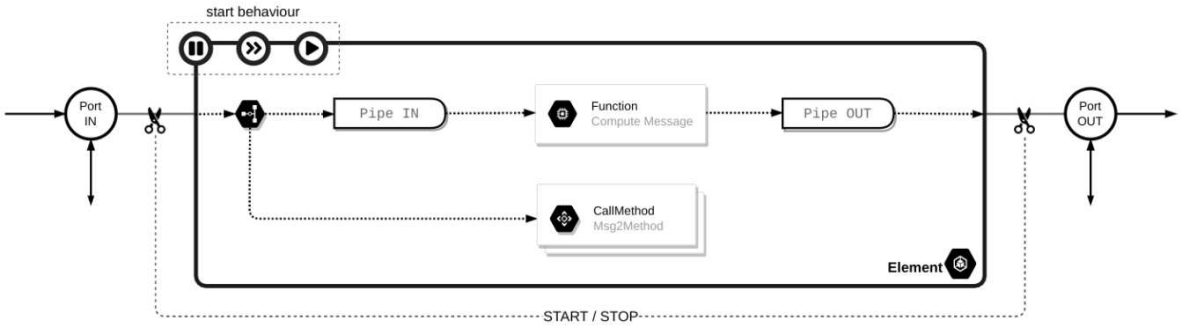
3 TRABALHOS RELACIONADOS

Como parte da pesquisa realizada para a proposta deste trabalho, foram levantadaos na documentação alguns frameworks de proposta semelhantes para análise que serão detalhados neste capítulos.

3.1 PyMAPE

O PyMAPE foi um trabalho realizado por Palombo (2021) em sua tese de mestrado na qual se propõe a ser um framework que suporta o desenvolvimento e a implantação de sistemas autônomos de maneira distribuída e descentralizada. (PALOMBO, 2021). O cerne de sua implementação está na abstração da figura do elemento. O elemento é a abstração das etapas que constituem o Monitor, Analyze, Plan, Execute (MAPE) onde tem seu comportamento de inicialização variável de acordo com sua respectiva categoria como mostra a figura 3:

Figura 4 – Visão interna de um elemento MAPE do PyMAPE



Fonte: (PALOMBO, 2021).

Um elemento pode assumir os seguintes comportamentos:

- Inicialização Manual: Um elemento não emite itens até que o método start() seja manualmente chamado.
- Inicialização ao se inscrever: O elemento assume um comportamente semelhante a um observador onde ele só será inicializado quando outros elementos estiverem inscritos para receber atualizações.
- Começar na inicialização: Assim que um objeto é criado, ele ja está pronto para transmitir e receber itens (mensagens).

Os elementos implementam uma adaptação do padrão de projeto observador, na qual utiliza de portas de entrada e saída permitindo assim um fluxo entre elementos independente da

ordem. Dessa forma, não necessariamente o fluxo do MAPE deve seguir a ordem tradicional gerando maior flexibilidade de implementação do laço de controle. Essa comunicação reativa entre elementos acontece por meio de itens que assumem dois tipos: CallMethod que pode invocar um comportamento em um elemento ou Message que apenas transmite alguma informação entre elementos.

Outro ponto relevante é a arquitetura de projeto implementada pelo framework. A aplicação é representada por um objeto que pode conter um ou mais níveis e pertence à classe "mape", responsável por inicializar a aplicação e suas dependências. Um nível, semelhante a um módulo descrito no capítulo dois, tem sua implementação opcional, ou seja, quando um laço não pertence a nenhum nível específico, ele é atribuído diretamente a aplicação.

Embora o PyMAPE permita modularizar a criação de laços por escopo, o que favorece a manutenção do software, a definição de hierarquias em camadas não é muito clara. Os níveis são estruturados de forma plana, organizados em uma lista, o que faz com que todos os laços, independentemente de seu escopo, compartilhem a mesma camada de prioridade. Isso pode sobrecarregar a classe de aplicação e tornar a leitura do código mais complexa contrariando a proposta de modularização em camadas de componentes deste trabalho.

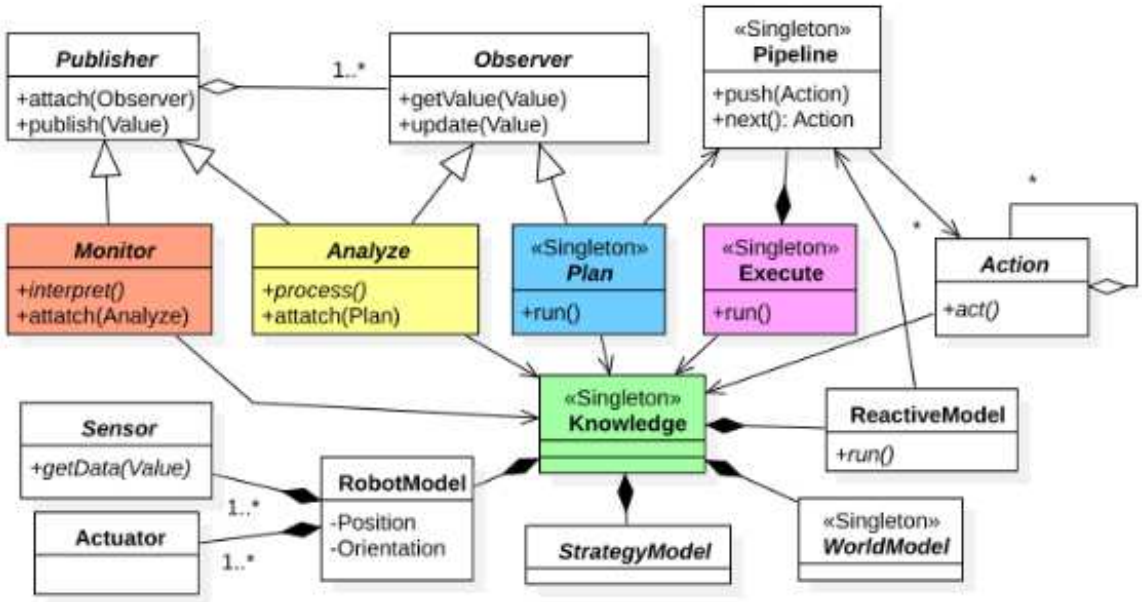
3.2 RoCS Framework

RoCS é um framework projetado por Ramos et al.(2019) focado no campo da robótica e implementado na linguagem de programação C++. (RAMOS *et al.*, 2019). Diferente do PyMAPE, temos que cada etapa do MAPE-K tem sua própria classe como é possível ver na figura 4.

A comunicação entre os elementos do laço acontece de maneira diferente dependendo da etapa do laço. O monitorador utiliza o padrão observador e tem como inscrito o analisador, que por sua vez também implementa tal padrão e tem como inscrito o planejador. A partir do planejador, as etapas seguem o padrão único para toda a aplicação. O planejador insere ações na pipeline que posteriormente são lidas e efetuadas pelo executor. Dessa forma, é possível denotar que uma aplicação no RoCS é composta unicamente de um laço que pode conter vários sensores e analisadores porém apenas um centro de tomada de decisões. Todas as etapas são executadas em threads separadas e se comunicam com uma única base de conhecimento que contém modelos genéricos do ambiente, robô, estratégia e ações e está disponível globalmente.

Apesar da arquitetura ser de laço único para toda a aplicação, que vai contra a pro-

Figura 5 – Diagrama de classes do RoCS Framework



Fonte: (RAMOS *et al.*, 2019).

posta deste trabalho de ampla modularização, os modelos genéricos oferecem ao desenvolvedor uma base de implementação mais consistente facilitando o aprendizado e aplicação da arquitetura MAPE-K.

3.3 Trabalhos não mencionados

Durante a revisão da literatura, outros frameworks foram analisados com o objetivo de embasar a formulação do Cat Framework. Entretanto, devido à similaridade com os já aqui discutidos, não foram incluídos na análise comparativa. A escolha por não compará-los se deu principalmente pela redundância de suas propostas, que não acrescentariam novos conceitos ou pontos de análise significativos aos já mencionados.

O trabalho de Silva (2021) propõe um framework genérico para a criação de sistemas autoadaptativos em ambientes de nuvem contêinerizados. Embora bastante relevante, sua comparação não foi realizada devido à semelhança com os frameworks já abordados. (SILVA, 2021).

O trabalho de Nakagawa et al. (2022) propõe a implementação de um controlador externo e uma estrutura baseada no padrão MAPE-K para aprimorar as funcionalidades de sistemas Internet of Things (IoT) e robótica. O estudo apresenta um framework que facilita a conversão de eventos utilizando o ciclo MAPE-K. Embora o laço seja implementado em sistemas embarcados, o foco da pesquisa não está na evolução arquitetural do MAPE-K, mas na extensão

e evolução do comportamento desses sistemas por meio de eventos emitidos por um controlador externo. Por essa razão, o trabalho não foi incluído na comparação deste capítulo. (NAKAGAWA *et al.*, 2022).

3.4 Comparação entre os trabalhos

Apesar da existência de outros frameworks citados anteriormente que se propõe a implementar o padrão MAPE-K para arquitetura de laços de controle, a proposta do Cat Framework diverge em sua finalidade e busca melhorias a fim de obter um modelo mais amplo que apoie o desenvolvimento no ramo da computação embarcada como mostra a tabela 1.

Tabela 1 – Comparação entre frameworks

Framework	Área de aplicação	Componentizado	Fácil instalação	CLI para projeto
PyMAPE	Geral	Sim	Sim	Não
RoCS	Robótica	Não	Não	Não
Cat	Embarcados	Sim	Sim	Sim

Fonte: o autor.

4 METODOLOGIA

Este capítulo descreve a metodologia adotada para o desenvolvimento do *Cat Framework*, abrangendo desde a definição da arquitetura até a implementação e validação da proposta por meio de uma prova de conceito. As etapas foram estruturadas conforme descrito a seguir.

4.1 Diagramação da proposta inicial da arquitetura

A primeira etapa consistiu na modelagem da arquitetura do *framework*, utilizando a linguagem de modelagem UML com o auxílio da ferramenta LucidChart. O *Cat Framework* foi projetado com foco em acessibilidade para desenvolvedores iniciantes e modularidade para facilitar a manutenção e reutilização de componentes. A visão geral da arquitetura do *Cat Framework* é ilustrada na Figura 6.

A arquitetura proposta adota o padrão de projeto *Composite*, no qual um módulo pode conter um ou mais componentes, incluindo a possibilidade de aninhamento de múltiplos módulos. Cada componente possui sua própria base de conhecimento e compartilha, adicionalmente, uma base comum associada ao módulo. O padrão *Observer* foi aplicado para permitir que componentes ativem seus sucessores após a execução de suas ações.

Além disso, definiu-se uma estrutura específica para as ações e estratégias:

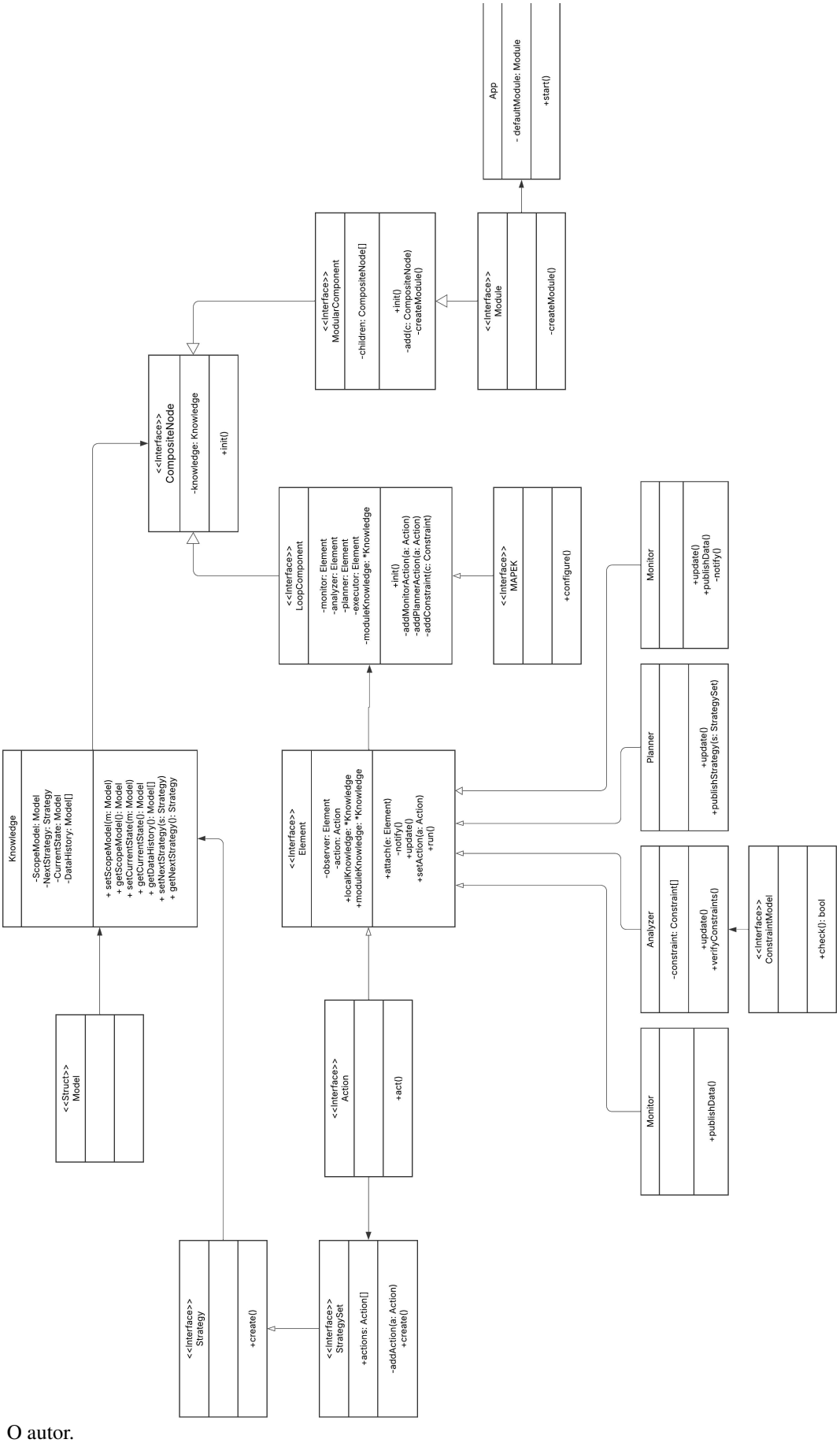
- Ações: Encapsulam funções executadas nas etapas de monitoramento e planejamento.
- *Constraints*: Validações booleanas aplicadas na etapa de análise, que indicam a necessidade de intervenção autônoma.
- Estratégias: Sequências de ações definidas para o executor aplicar conforme critérios estabelecidos.

4.2 Interface de linha de comando

Para facilitar a adoção e padronização de projetos baseados no *Cat Framework*, foi desenvolvida uma *Command Line Interface* (CLI) em C++, compilada para sistemas Linux com arquitetura AMD64. A CLI possui funcionalidades para:

- Inicializar um novo projeto com estrutura de pastas e arquivos padrão;
- Realizar a compilação do projeto com o uso do ESP-IDF;
- Executar o upload do firmware para a placa ESP32-S2-MINI;
- Ativar o modo debug para visualização de logs em tempo real.

Figura 6 – Diagrama de classes do Cat Framework



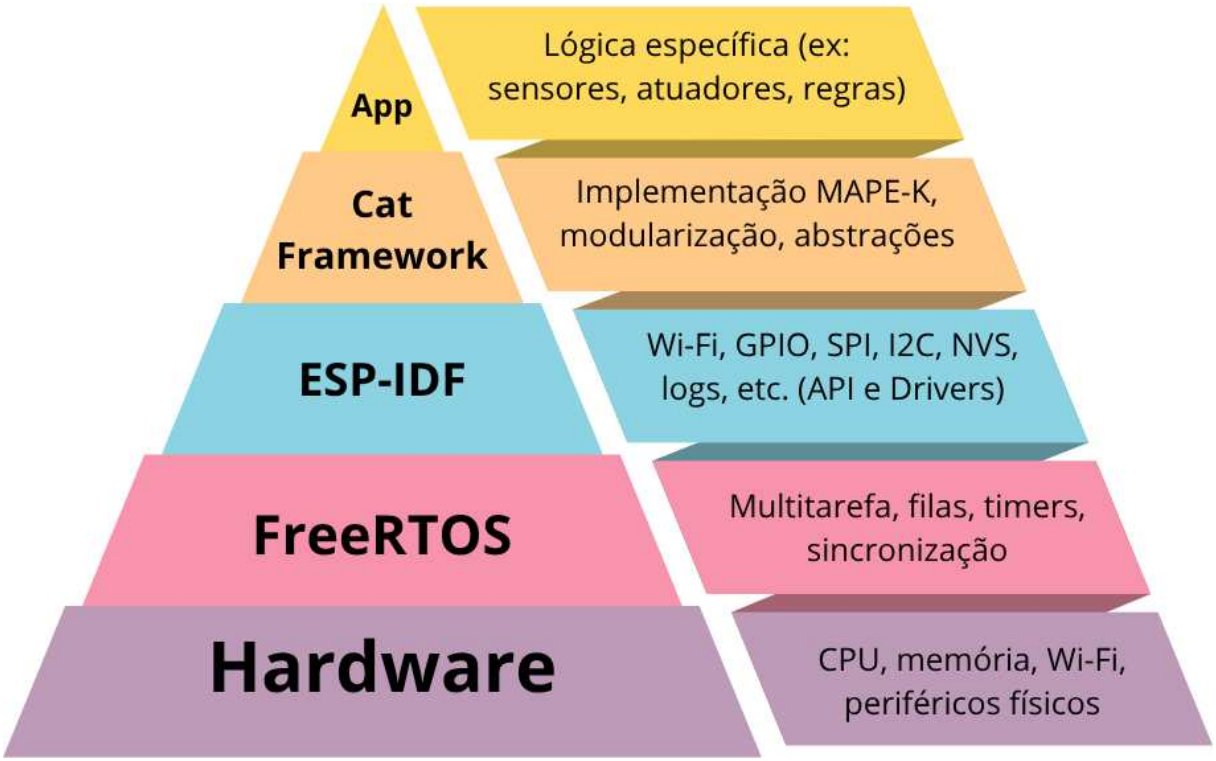
Fonte: O autor.

Essa ferramenta tem como objetivo auxiliar o desenvolvedor na criação e gerenciamento de projetos, seguindo boas práticas e facilitando o entendimento da estrutura interna do *framework*.

4.3 Prova de conceito

O *Cat Framework* foi desenvolvido sobre o ESP-IDF, utilizando suas bibliotecas nativas para gerenciar as funcionalidades de baixo nível do microcontrolador ESP32-S2, como controle de Wi-Fi, manipulação de GPIOs e execução multitarefa. Por meio da integração com o kernel FreeRTOS, o Framework irá estruturar suas ações e monitoramentos utilizando tarefas paralelas (tasks), permitindo que diferentes módulos do sistema operem de forma concorrente e responsiva. Além disso, o processo de compilação, upload e monitoramento em tempo real é realizado por meio da ferramenta de linha de comando fornecida pelo próprio ESP-IDF, garantindo integração direta com o hardware e aproveitamento dos recursos avançados do chip. A Figura 7 ilustra as camadas envolvidas em uma aplicação *Cat*.

Figura 7 – Arquitetura Geral de uma aplicação Cat.



Fonte: O autor.

Para validar a aplicabilidade e funcionalidade do framework, foi desenvolvida uma prova de conceito utilizando a placa ESP32-S2-MINI. O experimento consistiu na criação de um

sistema capaz de monitorar redes Wi-Fi ativas no ambiente. A cada variação detectada nas redes, um sinal luminoso era ativado, representando a execução de uma ação autônoma em resposta ao evento detectado.

O experimento foi conduzido em uma máquina com sistema operacional Linux, tendo como pré-requisitos a instalação do Python na versão 3.13.5 ou superior e do ESP-IDF na versão 5.4 ou superior, com suporte às placas da família ESP32, e que foi utilizado para a construção (*build*) e gravação (*flash*) da aplicação.

4.4 Cronograma

Tabela 2 – Cronograma

Etapas	Out-Dez/2024	Jan-Fev/2025	Mar/2025	Abr-Jul/2025	Ago/2025
Pesquisa bibliográfica	X	X			
Levantamento da proposta		X			
Escrita do trabalho		X	X		
Defesa da primeira etapa			X		
Implementação do projeto				X	
Prova de conceito				X	
Análise dos resultados					X
Defesa da segunda etapa					X

Fonte: o autor.

5 RESULTADOS

Neste capítulo, apresenta-se a prova de conceito de uma aplicação utilizando o *Cat Framework* desenvolvido neste trabalho. O repositório contendo a implementação completa do framework está disponível em Ávila (2025a), enquanto o repositório correspondente à aplicação utilizada na prova de conceito pode ser acessado em Ávila (2025b). Essa prova de conceito teve como objetivo verificar o comportamento do framework em um cenário de monitoramento de redes Wi-Fi, empregando a placa ESP32-S2-MINI. Os resultados apresentados nesta seção referem-se à capacidade do sistema de detectar alterações no ambiente — especificamente mudanças nas redes Wi-Fi ativas — e reagir de forma adequada por meio do acionamento de um sinal luminoso. As medições foram realizadas após a instalação correta do ambiente experimental, conforme descrito na metodologia.

5.1 Implementação do Projeto

Como parte da proposta deste documento, foi desenvolvido um projeto prático, ainda que simples, utilizando o *Cat Framework*. Depois de baixado o *framework*, como demonstrado em Ávila (2025a), na raiz do repositório, estão disponíveis dois *scripts* de instalação: *install.sh* (para plataformas Linux amd64) e *install − esp.sh* (para dispositivos da família *ESP*32). Para este experimento, utilizou-se o script destinado ao ESP32 com o parâmetro *esp*32*s*2.

Durante a instalação, são executadas as seguintes etapas:

- Compilação da biblioteca principal;
- Geração dos templates;
- Instalação da interface de linha de comando (CLI) do Cat.

A ferramenta CLI, ilustrada na Figura 8, oferece o comando *help*, que exibe uma

Figura 8 – Interface de linha de comando do Cat Framework.

```
[wendril@wendril-vivobook ~]$ catcli help

  /\_/\   Cat Framework CLI
 ( o.o )  "Simplify. Evaluate. Move forward."
<  ^ ^
------------------------------------------------------------

Cat CLI - available commands:
  help                              Show this help message
  init <project> [--platform <amd64|esp32s2|esp32s3|esp32>] [--debug <true|false>]
                                    Initialize a new project (default: amd64, debug: false)
  build                             Build the project for the current platform in cat.config.txt
  run                               Build and run the project for the current platform in cat.config.txt
[wendril@wendril-vivobook ~]$
```

Fonte: elaborado pelo autor (2025).

breve descrição dos comandos disponíveis. Entre os principais comandos, destacam-se:

- *init*: cria um novo projeto, recebendo como parâmetros o nome, a plataforma (*esp*32*s*2) e a opção de ativar o modo debug, que permite a visualização do console serial;
- *debug*: realiza a compilação do projeto, gerando os arquivos na pasta *dist*;
- *run*: compila e realiza o flash da aplicação, identificando automaticamente as portas seriais disponíveis.

Ao iniciar um projeto, a CLI gera automaticamente a seguinte estrutura de pastas e arquivos (Ver Figura 9):

- modules/: contém os módulos do sistema (componentes funcionais);
- mapek/: implementação do ciclo MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge);
- behavior/: define ações executáveis pelos componentes;
- constraints/: define restrições e regras de negócio;
- strategy/: contém as estratégias aplicadas pelo Planner;
- cat.config.txt: arquivo de configuração do projeto gerado pelo comando init;
- CMakeLists.txt: define o processo de compilação com o ESP-IDF;
- sdkconfig.defaults: configurações padrão utilizadas na compilação pelo ESP-IDF.

Figura 9 – Estrutura de pastas e arquivos de um projeto Cat.

Fonte: elaborado pelo autor (2025).

5.2 Resultado do Experimento

O projeto implementado trata-se de um WiFi Scanner, responsável por monitorar redes ativas no ambiente e acionar um sinal luminoso sempre que uma alteração for detectada. A aplicação foi construída com base em uma única instância do modelo MAPE-K, responsável por coordenar todo o ciclo de monitoramento e resposta.

A implementação teve início com a configuração do arquivo *main.cpp*, na pasta *main*,

a fim de inicializar dos módulos de Wi-Fi e instanciar o módulo principal do sistema. Na etapa seguinte, desenvolveu-se o modelo *apnum*, destinado a representar o dado a ser monitorado, especificamente a quantidade de redes disponíveis.

A partir dessa definição, iniciou-se a implementação das etapas do MAPE, começando pela função de monitoramento, localizada na pasta *behavior*, responsável por coletar as redes disponíveis durante aproximadamente três segundos e publicar o modelo na base de conhecimento. Com o resultado atual da medição registrado nessa base, implementou-se uma constraint que compara os valores obtidos com o histórico, sinalizando ao planejador sempre que houver alteração na quantidade de redes existentes na região.

Devido à simplicidade do projeto, a função de planejamento pôde ser mantida conforme o padrão disponibilizado pelo framework, limitando-se a definir uma estratégia básica para as ações do executor. Em cenários de maior complexidade, é possível empregar modelos de inteligência artificial capazes de produzir ações condizentes com o nível de abstração do ambiente monitorado.

Por fim, foi desenvolvida a função executora do ciclo, presente na pasta *behavior*, responsável por acionar o LED em três piscadas consecutivas, com intervalos de 150 milissegundos entre cada uma.

Conforme ilustrado na Figura 10, o comando *run* foi executado com o modo debug ativado, conforme especificado no arquivo $cat.config.txt$. Essa configuração possibilitou a visualização, em tempo real, dos logs gerados pela placa durante a execução.

O comportamento do sistema foi consistente com as expectativas: o LED piscava sempre que uma nova rede era detectada ou uma rede deixava de ser visível, evidenciando a eficácia do laço de controle implementado.

Figura 10 – WiFi Scanner sendo executando usando o Cat Framework.

Fonte: elaborado pelo autor (2025).

6 CONCLUSÕES E TRABALHOS FUTUROS

6.1 Considerações gerais

Este trabalho apresentou o desenvolvimento e a aplicação do *Cat Framework* em uma prova de conceito voltada ao monitoramento de redes Wi-Fi utilizando a placa ESP32-S2-MINI. A implementação demonstrou a capacidade do framework de realizar varreduras ativas e emitir sinais visuais diante de variações detectadas, validando a eficácia arquitetural do modelo MAPE-K adotado.

Os resultados obtidos indicam que o *Cat Framework*, ainda em estágio inicial, mostra-se promissor como ferramenta de apoio à adoção do modelo MAPE-K em projetos embarcados que demandam monitoramento adaptativo. Sua proposta de modularidade, aliada à simplicidade e flexibilidade da interface de linha de comando, contribui para a redução da complexidade no desenvolvimento de sistemas autônomos.

6.2 Limitações e trabalhos futuros

Como trabalho futuro, recomenda-se a integração do *Cat Framework* com técnicas de aprendizado de máquina, com foco na análise preditiva dos dados coletados, ampliando assim sua capacidade de adaptação inteligente. Além disso, sua compatibilidade deve ser estendida a outras plataformas de hardware e sensores, aumentando o escopo de aplicações possíveis.

É importante também aprimorar a nomenclatura e a estrutura de inicialização dos componentes do framework, visando facilitar sua compreensão mesmo com uma leitura superficial da documentação. Essa melhoria impacta diretamente a curva de aprendizado de novos desenvolvedores.

Outro aspecto relevante para evolução do projeto é a implementação de mecanismos de persistência de dados, permitindo o salvamento em arquivos e a restauração do estado do sistema após reinicializações.

Adicionalmente, a comunicação entre os componentes do ciclo e a base de conhecimento deve ser otimizada. Atualmente, o uso intensivo de conversões de tipos e ponteiros inteligentes compartilhados pode gerar aumento significativo no consumo de memória e processamento, o que é crítico em sistemas embarcados com recursos limitados.

Por fim, propõe-se tornar a ferramenta de linha de comando mais independente do

ESP-IDF, minimizando a necessidade de integração com bibliotecas externas e reduzindo o tempo de compilação e consumo de recursos.

REFERÊNCIAS

Amazon Web Services. **O que é uma CLI?** 2024. <https://aws.amazon.com/pt/what-is/cli/>. Acesso em: 5 mar. 2025. Disponível em: <https://aws.amazon.com/pt/what-is/framework/>.

Amazon Web Services. **O que é uma framework?** 2024. <https://aws.amazon.com/pt/what-is/framework/>. Acesso em: 5 mar. 2025. Disponível em: <https://aws.amazon.com/pt/what-is/framework/>.

BARRETT, T.; PROPER, H. A. Component based solutions under architecture. **arXiv preprint arXiv:2105.08702**, 2021.

COMPUTING, A. *et al.* An architectural blueprint for autonomic computing. **IBM White Papel**, Citeseer, 2006.

Espressif Systems. **ESP-IDF Programming Guide - Getting Started (ESP32)**. [S.l.], 2025. Acesso em: 8 ago. 2025. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/index.html>.

FOX, J. A formal model for dynamically adaptable services. **arXiv preprint arXiv:1011.2652**, 2010.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design patterns: elements of reusable object-oriented software**. [S.l.]: Pearson Deutschland GmbH, 1995.

GANSSLE, J. **Embedded systems dictionary**. [S.l.]: CRC press, 2003.

HALIMA, R. B.; HACHICHA, M.; JEMAL, A.; KACEM, A. H. Mape-k patterns for self-adaptation in cyber-physical systems. **Journal of Supercomputing**, v. 79, n. 5, 2023.

HORN, P. Autonomic computing: Ibm's perspective on the state of information technology. IBM Researcher, 2001.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. **Computer**, IEEE, v. 36, n. 1, p. 41–50, 2003.

LANDIN, N.; NIKLASSON, A.; BOSSON, G.; REGNELL, B. Development of object-oriented frameworks. **Department of Communication System. Lund Institute of Technology, Lund University. Lund, Sweden**, Citeseer, p. 79, 1995.

MALBURG, L.; HOFFMANN, M.; BERGMANN, R. Applying mape-k control loops for adaptive workflow management in smart factories. **Journal of Intelligent Information Systems**, Springer, v. 61, n. 1, p. 83–111, 2023.

MECIBAH, Z.; BOUTEKKOUK, F. Designing of self-adaptive embedded system: an overview of recent advances and prospects. In: UNIVERSITY OF OUM EL BOUAGHI, ALGERIA. **Proceedings of the 1ère Conférence Nationale sur l'Application d'Intelligence Artificielle et le Développement Durable (CNAIADD'2024)**. [S.l.], 2024.

NAKAGAWA, H.; TSUCHIDA, S.; TRAMONTANA, E.; FORNAIA, A.; TSUCHIYA, T. Embedded system evolution in iot system development based on mape-k loop mechanism. **arXiv preprint arXiv:2205.13375**, 2022.

PALOMBO, E. **A software framework to support the development and deployment of autonomous systems**. Dissertação (Mestrado em Ciência da Computação) — Universit‘a degli Studi dell’Aquila, L’Aquila, 2021.

RAMOS, L.; DIVINO, G. L. G.; LOPES, G. C.; FRANÇA, B. B. N. de; MONTECCHI, L.; COLOMBINI, E. L. The rocs framework to support the development of autonomous robots. **Journal of Software Engineering Research and Development**, v. 7, p. 10–1, 2019.

RATHOD, A. B.; KHADSE, R.; BAGWAN, M. F. Serial computing vs. parallel computing: A comparative study using matlab. **International Journal of Computer Science and Mobile Computing**, v. 3, n. 5, p. 815–820, 2014.

ROMERO-GARCéS, A.; HIDALGO-PANIAGUA, A.; GONZáLEZ-GARCíA, M.; BANDERA, A. On managing knowledge for mape-k loops in self-adaptive robotics using a graph-based runtime model. **Applied Sciences**, v. 12, n. 17, 2022. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/12/17/8583>.

SILVA, A. N. d. O. d. Um framework de adaptação baseada no padrão mapek. 2021.

TIWARI, U. K.; KUMAR, S. **Component-based software engineering: Methods and metrics**. [S.l.]: Chapman and Hall/CRC, 2020.

WÄTZOLDT, S.; NEUMANN, S.; BENKE, F.; GIESE, H. Integrated software development for embedded robotic systems. In: SPRINGER. **Simulation, Modeling, and Programming for Autonomous Robots: Third International Conference, SIMPAR 2012, Tsukuba, Japan, November 5-8, 2012. Proceedings 3**. [S.l.], 2012. p. 335–348.

ÁVILA, W. **Cat Framework: Um Framework MAPE-K para Computação Autônoma em Sistemas Embarcados**. 2025. <https://github.com/TheWendril/Cat-Framework>. Acesso em: 08 ago. 2025.

ÁVILA, W. **Cat-Network-Scan: Prova de Conceito com o Cat Framework para Monitoramento de Redes Wi-Fi**. 2025. <https://github.com/TheWendril/Cat-Network-Scan>. Acesso em: 08 ago. 2025.