Comparação de Linguagens para Sistemas Embarcados: Rust, C/C++ e MicroPython

Fabiano Fruett
19 de novembro de 2024

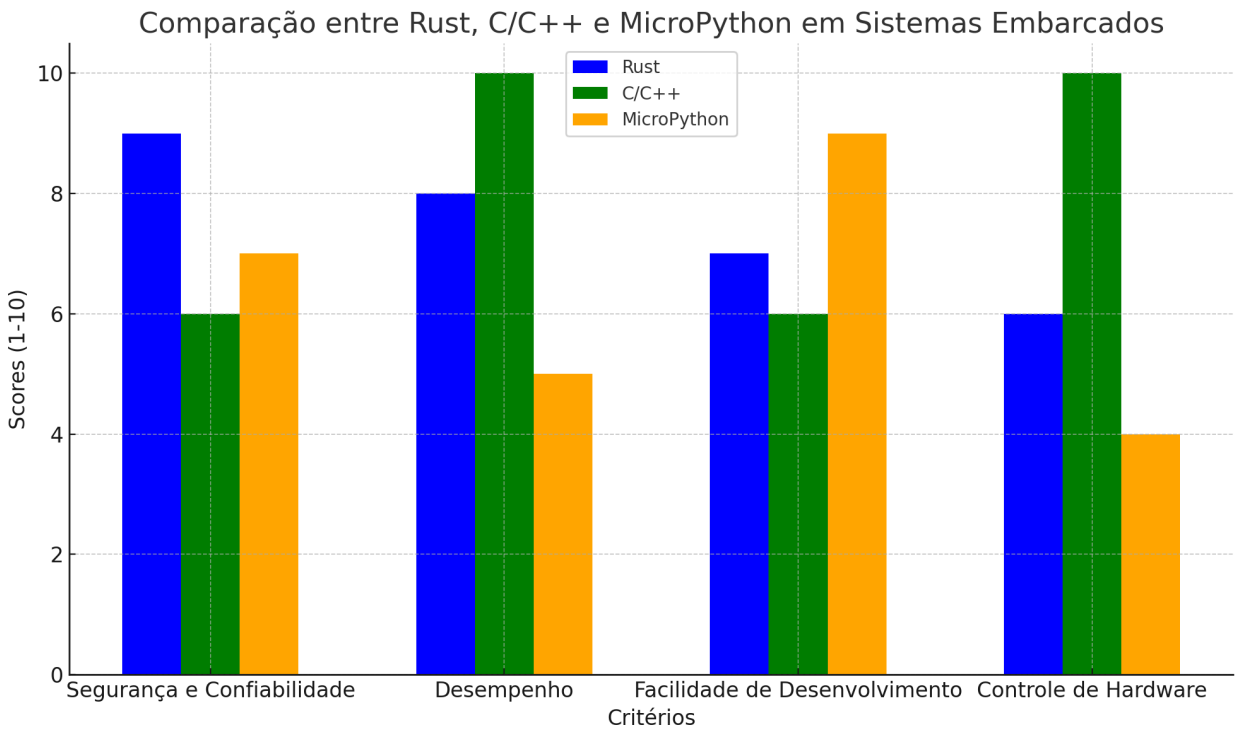
Todo microcontrolador é composto por circuitos eletrônicos que manipulam massivamente sinais digitais, conhecidos como bits. Um bit é a menor unidade de informação, podendo representar dois estados: 0 ou 1. O agrupamento de 8 bits forma um Byte, que é a unidade básica usada para armazenar e processar dados em microcontroladores. Esses dispositivos operam utilizando uma linguagem puramente numérica, conhecida como linguagem de máquina, que instrui o microcontrolador a realizar operações diretamente sobre os bits.

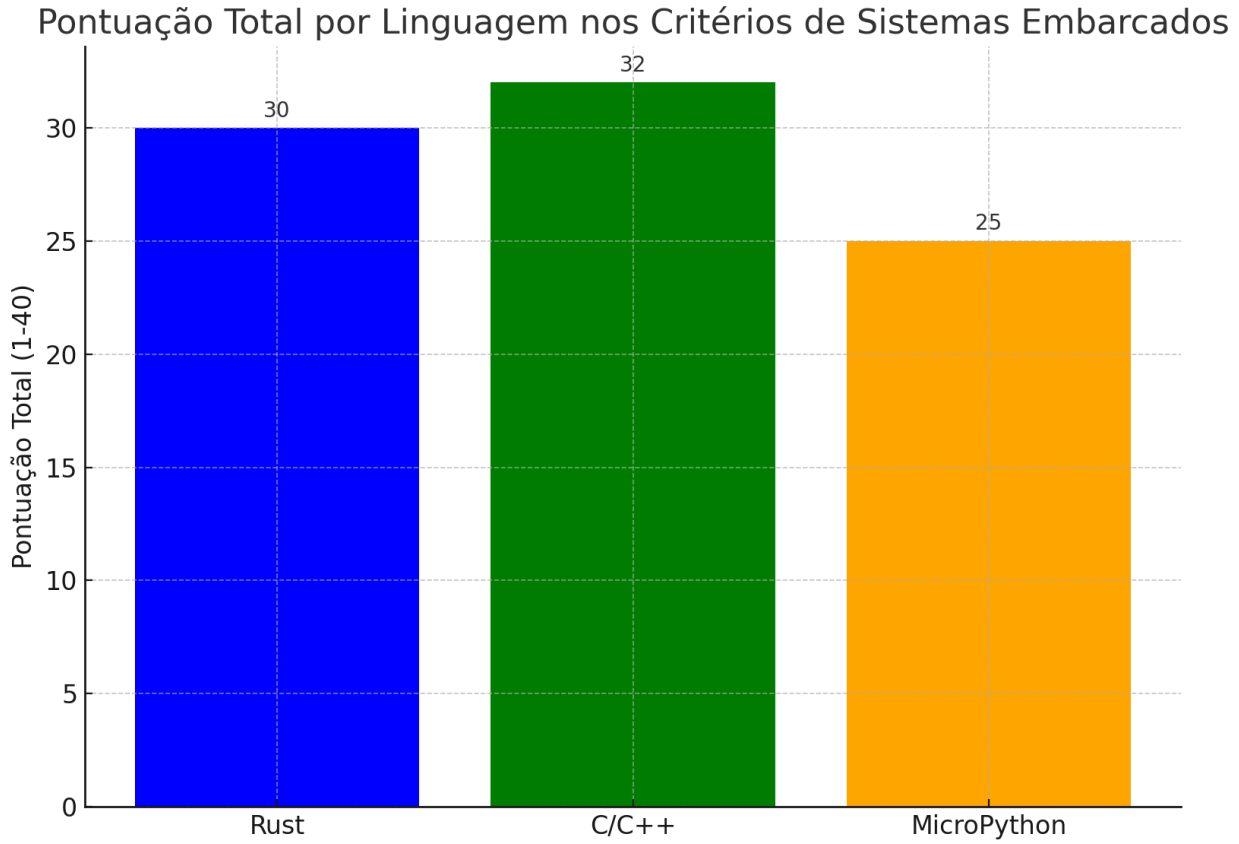
No entanto, a linguagem de máquina não é natural para os seres humanos, pois exige um conhecimento profundo do funcionamento do hardware e é de difícil leitura e escrita. Para facilitar a programação, os desenvolvedores utilizam linguagens de programação de mais alto nível, que oferecem abstrações sobre o hardware, permitindo que os bits sejam manipulados de maneira mais intuitiva e eficiente.

Entre essas linguagens, o Assembly se destaca como uma opção de muito baixo nível, oferecendo controle direto sobre o hardware e uma representação quase direta das instruções de máquina. No entanto, devido à sua complexidade e foco em detalhes específicos do hardware, o Assembly não será considerado nesta comparação. Em vez disso, focaremos em linguagens de nível mais alto, como C/C++, Rust e MicroPython, que oferecem um equilíbrio entre desempenho, segurança e facilidade de uso.

A escolha da linguagem que utilizamos para programar um microcontrolador depende de diversos fatores, como o tipo de aplicação, as restrições de recursos, o nível de controle sobre o hardware, a facilidade de desenvolvimento e a segurança do sistema. Cada linguagem oferece diferentes níveis de abstração e controle, influenciando diretamente o desempenho, a confiabilidade e a facilidade de uso na implementação dos sistemas embarcados.

A seguir apresentamos um comparativo que mostra as avaliações de **Rust**, **C/C++** e **MicroPython** em diferentes critérios importantes para sistemas embarcados: segurança e confiabilidade, desempenho, facilidade de desenvolvimento e controle de hardware.


Comparação entre Rust, C/C++ e MicroPython em Sistemas Embarcados
Rust
C/C++
MicroPython
Scores (1-10)
0
2
4
6
8
10
Segurança e Confiabilidade
Desempenho
Facilidade de Desenvolvimento
Controle de Hardware
Critérios


Pontuação Total por Linguagem nos Critérios de Sistemas Embarcados
Pontuação Total (1-40)
0
5
10
15
20
25
30
30
32
25
Rust
C/C++
MicroPython

Os critérios apresentados para a comparação entre **Rust**, **C/C++** e **MicroPython** foram baseados em uma combinação de práticas amplamente reconhecidas no desenvolvimento de sistemas embarcados e na literatura técnica e comunitária sobre essas linguagens. Embora não tenha sido diretamente retirado de uma única fonte específica, aqui estão algumas referências amplamente aceitas para esses critérios:

1. **Segurança e Confiabilidade**:
    - Rust é reconhecido pela sua segurança de memória, com uma forte ênfase na prevenção de erros comuns em sistemas embarcados, como em *The Rust Programming Language Book*.
    - C/C++ são mencionados como mais propensos a erros de memória e falhas em vários textos clássicos de sistemas embarcados, como em *Programming Embedded Systems in C and C++* de Michael Barr.
2. **Desempenho**:
    - A comparação de desempenho entre Rust e C/C++ é discutida em várias análises técnicas e benchmarks, como em Rust Performance Comparisons.
    - Para MicroPython, a documentação oficial da MicroPython e comparações feitas em fóruns técnicos frequentemente indicam que, embora seja fácil de usar, seu desempenho não é tão otimizado quanto linguagens compiladas como C/C++.
3. **Facilidade de Desenvolvimento**:
    - MicroPython é amplamente elogiado pela comunidade educacional por sua facilidade de uso e documentação amigável. Você pode encontrar essas discussões em publicações sobre ensino de sistemas embarcados, como *MicroPython for the Internet of Things* de Charles Bell.
    - Rust é elogiado por suas ferramentas modernas e ecossistema crescente, embora tenha uma curva de aprendizado mais íngreme, conforme observado em fóruns de desenvolvedores como Rust Embedded Working Group.
4. **Controle de Hardware**:
    - C/C++ ainda dominam quando se trata de controle de hardware em sistemas embarcados devido à sua maturidade e vasto suporte de bibliotecas, conforme descrito em obras como *Making Embedded Systems* de Elecia White.
    - Comparações de controle de hardware em Rust e MicroPython podem ser vistas em discussões no Stack Overflow, blogs técnicos e fóruns de desenvolvedores.

Resumo:

C e C++ ainda dominam quando se busca o máximo desempenho e controle total sobre o hardware. C é particularmente popular para o desenvolvimento de sistemas operacionais e é amplamente usado em microcontroladores embarcados, presentes em quase todos os dispositivos eletrônicos. A linguagem proporciona um ambiente de alto nível que permite gerar código pequeno e eficiente, ideal para microcontroladores mais econômicos e de baixo consumo de energia. Assim, C deve continuar relevante no desenvolvimento de sistemas embarcados mais restritivos, onde a otimização de custo e eficiência são essenciais.

MicroPython é a escolha ideal para prototipagem rápida e ensino, principalmente em projetos de eletrônica e sistemas embarcados mais simples. A linguagem é leve e fácil de aprender, o que facilita o aprendizado de conceitos de programação e permite interações rápidas com o hardware. No entanto, MicroPython não é recomendado para sistemas embarcados que exigem alta eficiência e restrições rigorosas de tempo real, já que a linguagem sacrifica parte do desempenho para oferecer uma experiência mais acessível.

Rust é uma excelente opção quando segurança e confiabilidade são prioritárias. Com um sistema de gerenciamento de memória robusto, Rust previne falhas comuns de programação, como estouros de buffer e uso de ponteiros nulos, tudo isso sem comprometer drasticamente o desempenho. Além disso, Rust permite acesso ao hardware em baixo nível, o que o torna uma alternativa promissora para sistemas embarcados mais complexos que precisam equilibrar desempenho e segurança.