14º Congresso de Inovação, Ciência e Tecnologia do IFSP - 2023

Linguagem Rust como alternativa ao desenvolvimento de sistemas embarcados de 8 bits

SONAGERE P. [1], DEOTTI D.[2]

[1] Pierre Sonagere de Souza, Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Câmpus Campinas, pierre.sonagere@ifsp.edu.br.
[2] Diego Deotti, Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Câmpus Campinas, diegodeotti@ifsp.edu.br.

RESUMO: Este artigo apresenta os resultados preliminares da iniciação científica, que teve início no mês de junho de 2023. Esse documento relata os avanços na pesquisa e apresenta os dispositivos, algoritmos, característica e estratégias que serão utilizadas para realizar os testes da linguagem C e Rust, a fim de compará-las e gerar conhecimento na área. O foco desse trabalho são os microcontroladores de 8 bits, pois ainda são amplamente utilizados, porém devido as suas limitações de memória, necessitam de mais atenção no momento da implementação do *firmware*.

PALAVRAS-CHAVE: Sistemas Embarcados, Microcontroladores, Linguagem Rust.

Rust language as an alternative to 8-bit embedded systems design

ABSTRACT: This paper presents the partial results of the scientific research, which started in June 2023. This reports the team advances on the subject and shows the devices, algorithms, technical features, and strategies that will be used to carry out the C and Rust language tests, in order to compare them and generate knowledge. The focus is 8-bit microcontrollers, due its widely application, but due to their memory limitations, they need more care in the firmware implementation.

KEYWORDS: Embedded Systems, Microcontrollers, Rust programming language, C programming language

INTRODUÇÃO

A linguagem *Assembly* trouxe uma camada de abstração mais amigável do que o código de máquina, vinte anos depois, a linguagem C, de alto nível, simplificou a programação, impulsionando notáveis avanços na área de informática e eletrônica.

A linguagem C já é consolidada para essa tarefa, e é utilizada atualmente por diversas empresas e grupos de pesquisa, enquanto a linguagem *Rust* é jovem, porém bastante promissora possuindo potencial para ser sua sucessora. C é uma linguagem de longa data, predominantemente utilizada nessa área, enquanto Rust, uma linguagem mais atual, tem seu foco em segurança e desempenho. Sua característica principal é um sistema rigoroso de acesso a memória (*ownership*), que possui um conjunto de regras que são verificadas em tempo de compilação (Matsakis; Nicholas, Turon; Aaron. 2018), e não de execução como em C ou outras, que utilizam o esquema de *garbage collector* (coletor de lixo) como o Java. *Rust* vem tendo tanta significância que mesmo o criador do sistema Linux já comentou sobre o uso da linguagem C por *Rust* em algumas partes do Kernel do sistema. (Alecrim; Emerson 2022).

Através de pesquisa bibliográfica, não foi encontrado material comparativo entre as duas linguagens para microcontroladores de arquitetura de 8 bits, como o Atmega328, por exemplo.

Dessa forma, esse projeto busca preencher essa lacuna, executando uma série de algoritmos e outros aspectos de avaliação de linguagens de programação, resultando em um comparativo de desempenho não se limitando apenas a aspectos técnicos, analisando também outras características entre ambas as linguagens de programação.

MATERIAIS E MÉTODOS

Este estudo tem como objetivo comparar o desempenho de duas linguagens de programação diferentes, C e *Rust*, em relação à sua eficácia na programação de microcontroladores de 8 bits. Para isso, serão realizados testes utilizando as duas linguagens, empregando o mesmo conjunto de dados de entrada e algoritmos de processamento.

Os testes serão realizados de forma controlada, utilizando a mesma configuração de bancada e os mesmos algoritmos, complexidades e condições para ambas as linguagens. Nesta seção, iremos apresentar os materiais e métodos que serão utilizados para realizar os testes e comparar os resultados obtidos.

O microcontrolador utilizado para testes será o Atmega328 da *Microchip©*, através da plataforma *Arduino©*. Esta foi escolhida para a realização dos testes por ser acessível e possuir *hardware* voltado a prototipação. Sua *IDE* não será utilizada, pois ela suporta a linguagem *wiring©* que não contempla os objetivos desse trabalho. Como plataforma de programação da "placa Arduino", será utilizada a ferramenta *avrdude* como gravador e o *avr-gcc* como compilador para a linguagem C. Para o código *Rust* ser gerado pelo seu compilador *(rustc*), é necessário utilizar uma camada de *Abstração de Hardware* (*HAL*), para isso, a ferramenta *rust nightly compiler* em conjunto com a biblioteca *libudev-dev* serão aplicadas, além do *ravedude* que será utilizado para a gravação do *firmware*.

Em conjunto com o microcontrolador, será utilizado um relógio em tempo real (RTC), modelo DS1307, para que a contagem de tempo seja independente, um botão para testar uma entrada digital, um potenciômetro para testar as leituras do conversor analógico/digital e um LED para testar as saídas digitais e analógicas *(PWM)*. Dessa forma o ambiente de teste, será composto por: Computador placa *Arduino(c)*, *LED*, potenciômetro, botão e o RTC. A configuração final fica como a apresentada na figura 1 abaixo:

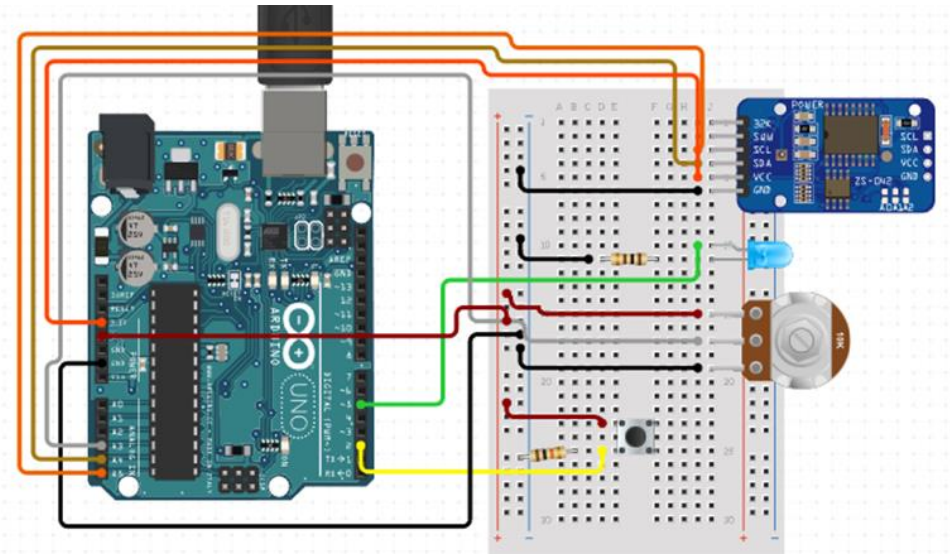

Figura 1 – Esquema simplificado do *hardware* utilizado para os testes

A avaliação da linguagem de programação, levará em consideração as seguintes características: eficiência, portabilidade, curva de aprendizado, tamanho da comunidade, paradigmas de linguagem suportados, acesso a interrupções, registradores e periféricos, manipulação de bits, gerenciamento da

memória, tamanho do arquivo binário, quantidade de código no cabeçalho *(overhead)*. E além disso, estabilidade nas versões da linguagem, ou seja, se ela mantém sua sintaxe funções, o que é requisito para aplicações de longa data evitando reescrita de código.

Os algoritmos utilizados para testar a eficiência da linguagem serão de complexidade$O(n \log n)$, $O(\log n)$, *O(n), O(n²) e* $O(n^3)$ além do funcionamento das interfaces de entrada/saída, acesso a periféricos, interrupções, memória e comunicação externa.

Para ler o tempo inicial e final do RTC, será utilizado rotina em linguagem *assembly* ou outro método similar. A tabela 1, abaixo, apresenta os algoritmos que serão utilizados para o teste:

TABELA 1. Lista de algoritmos selecionados para a comparação de eficiência da linguagem

Algoritmo	**Complexidade**	**Justificativa**
Leitura/Escrita do GPIO, ADC e etc	$O(1)$	Verificar a facilidade/desempenho de acesso as portas de entrada/saída.
Leitura escrita da memória de dados	$O(1)$	Verificar a integridade e suporte ao acesso da memória
Teste de interrupções e registradores	$O(1)$	Verificar a facilidade e o desempenho em lidar com interrupções externas e alterar registradores.
Teste de comunicação serial	$O(1)$	Verificar facilidade e capacidade de comunicação externa
Sequência de fibonacci	$O(n)$	Velocidade de cálculo
Bubble Sort	$O(n^2)$	Velocidade de execução
Quick Sort	$O(n \log n)$	Velocidade de execução
Busca binária	$O(\log n)$	Velocidade de execução
Multiplicação de matrizes	$O(n^3)$	Velocidade de execução

RESULTADOS PRELIMINARES E DISCUSSÃO

Através da leitura das referências, podemos concluir de forma preliminar que *Rust* é uma linguagem de programação multiparadigma, mais segura e moderna do que C e possui desempenho adequado para o uso em sistemas de computação de 32 bits.

O foco deste trabalho é em arquiteturas de 8 bits, portanto, conclusões mais assertivas serão expostas futuramente após os resultados de teste dos algoritmos apresentados na tabela 1. Até a presente data, apenas a etapa de estudo da linguagem foi finalizada. Os resultados práticos futuros serão ponderados com outros aspectos da linguagem não mensuráveis quantitativamente, como: amadurecimento da linguagem e estabilidade de sintaxe entre suas diferentes versões, pois para uma instituição reescrever o *firmware* de um produto que já em uso no mercado para a linguagem Rust se torna inviável. Uma vez que a linguagem C, sofre pouca alteração em sua sintaxe com o passar dos anos. Além disso, a curva de aprendizado da linguagem *Rust* é muito maior e muitas vezes os engenheiros de sistemas precisam ser treinados nela para que possam atuar na área.

CONCLUSÕES

Até a presente data conclui-se que a linguagem *Rust* é bastante promissora no seu emprego em sistemas embarcados de 8 bits, porém, ainda é necessário executar os testes práticos propostos para então ser mais assertivo. É importante dizer que, mesmo que a linguagem tenha todas as propriedades já expostas no texto, pode-se afirmar que ela não irá substituir a linguagem C em sua totalidade, mas sim ser uma alternativa para novos projetos, assim como, para a reescrita de *firmware* que necessite de

segurança em seu código dada a necessidade de diversas aplicações funcionarem sem parar como é o caso de algumas aplicações de internet das coisas, por exemplo.

O caso do uso de *Rust* é similar ao da linguagem C, que não extinguiu o uso da linguagem *assembly*, mas se tornou uma alternativa viável e popular na escrita de *firmware*. Entende-se que no futuro combinações entre duas, ou até três dessas linguagens poderão ser utilizadas no desenvolvimento de sistemas embarcados e plataformas microcontroladas.

AGRADECIMENTOS

Os autores agradecem o Instituto Federal de São Paulo pela oportunidade e apoio na realização desse trabalho, a todos os servidores do campus por apoiarem a área de pesquisa e ao Ministério da educação e ao CNPq.

REFERÊNCIAS

Sebesta; Robert W. Conceitos de linguagens de programação [recurso eletrônico]. Tradução técnica: Eduardo Kessler Piveta. 9. ed. Porto Alegre: Bookman, 2011.

Alecrim; Emerson. O linux é feito em C, mas Linus Torvalds já fala em usar a linguagem Rust. 22 jun. 2022. Acesso em 30 ago. 2023. Online. Disponível em : <https://tecnoblog.net/noticias/2022/06/28/o-linux-e-feito-em-c-mas-linus-torvalds-ja-fala-em-usar-a-linguagem-rust/> .

Wikipédia. Rust (linguagem de programação). 26 ago. 2023. Acesso em 30 ago. 2023. Online. Disponível em : <https://pt.wikipedia.org/wiki/Rust_(linguagem_de_programação)> .

Matsakis; Nicholas e Turon; Aaron. The Rust Programming Language. Acesso em: 1 set. 2023. Disponível em: <https://doc.rust-lang.org/book/title-page.html >.

Fukuda; Danilo. Uma avaliação comparativa das linguagens Rust e C para o ensino de programação. Acesso em: 1 set. 2023. Disponível em: < https://bdm.unb.br/bitstream/10483/31261/1/2021_DaniloYFukuda_tcc.pdf >.

Silva, J. C.; Santos, R. M.; Silva, A. L.; et al. A Comparative Study of Machine Learning Techniques for the Diagnosis of Alzheimer's Disease Based on Structural MRI Images. Electronics, Basel, v. 12, n. 1, p. 143-157, Dez. 2022.

CIRCUITO.IO. Seamless circuit design for your project. Disponível em: https://www.circuito.io/. Acesso em: 1 set. 2023.