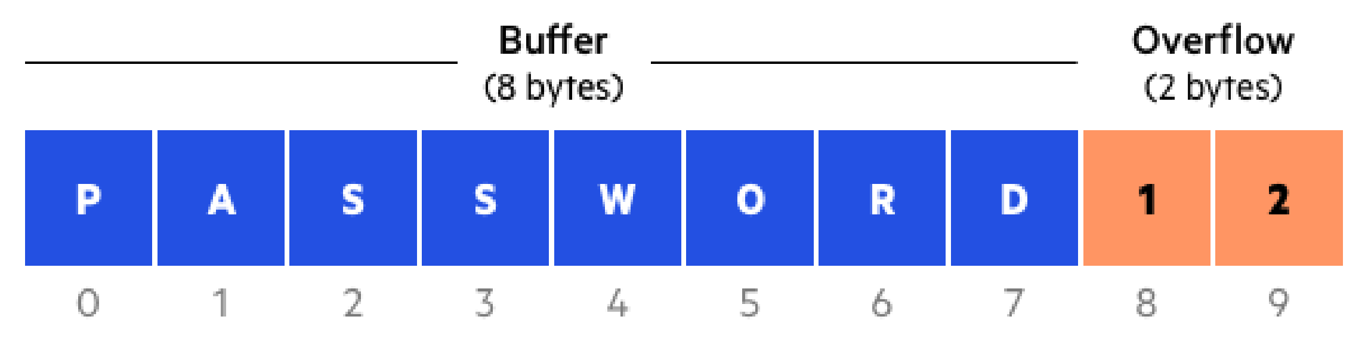
Performance Comparison between C++ and Rust Programming Languages in Real-Time Environments

Bryce Armstrong, Daniel Choi, Jawaher Alkhamis, Joshua Ripley, Kayla Morris
Mentor: Glen Uehara, General Dynamics Mission Systems
Ira A. Fulton Schools of Engineering, Arizona State University
GD Mission Systems

Motivation

C++ is widely used in real-time embedded systems due to its low-level computing power, minimal latency, and extensive feature set. However, C++ exposes users to security risks associated with memory management, such as

- ➢ Buffer Overflow
- ➢ Out-of-bound Access
- ➢ Null Pointer Dereference



These memory safety issues can have far-reaching consequences for system security.

Objective

The Rust programing language incorporates ownership, compile-time validations, and non-disposable memory handling to ensure memory safety and has emerged as an alternative to C++.

Performance between C++ and Rust was compared to validate whether Rust can be a viable alternative to C++ in real-time environments where deterministic execution and minimal latency are required.

Method

Five sensors were tested, each connected to a Raspberry Pi. Real-time applications were implemented in C++ and Rust. Execution time, CPU utilization, and RAM utilization were measured and compared in both C++ and Rust programming languages.

CIKXW1000
WiFi Sensor

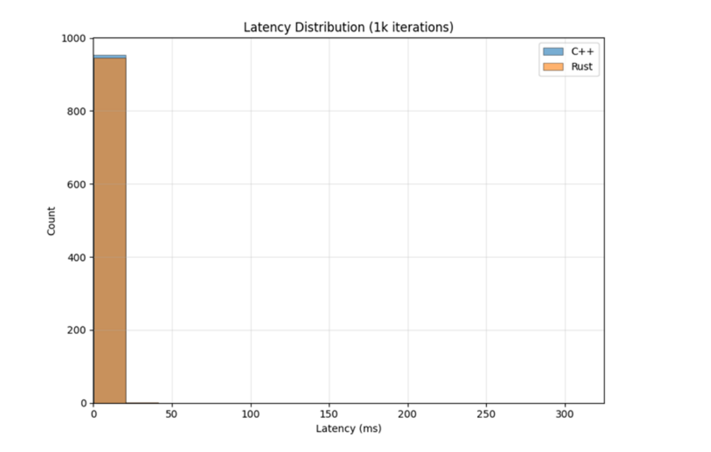
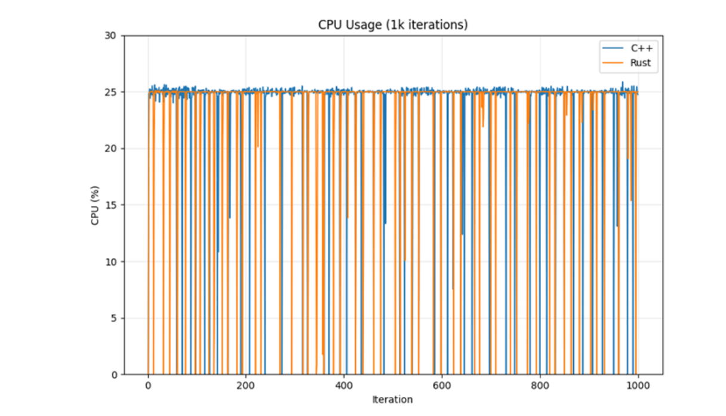
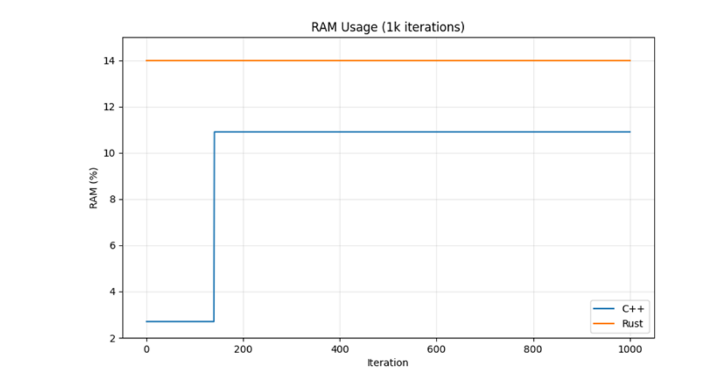
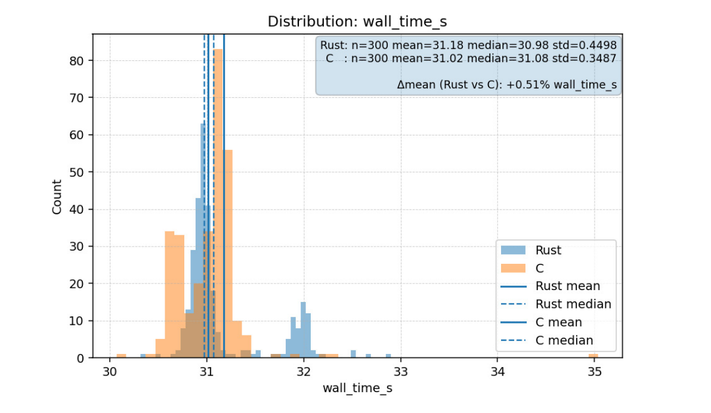
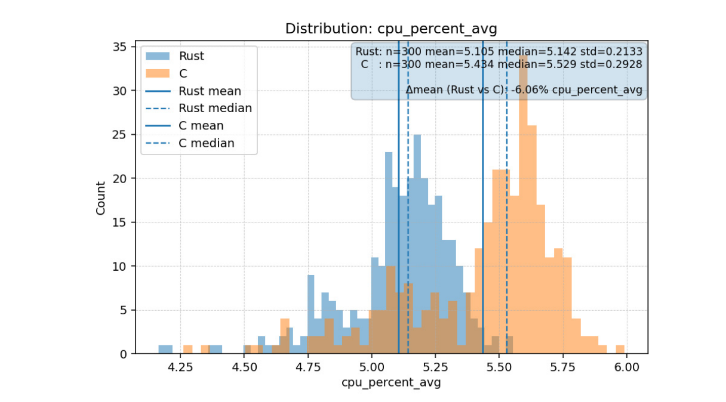
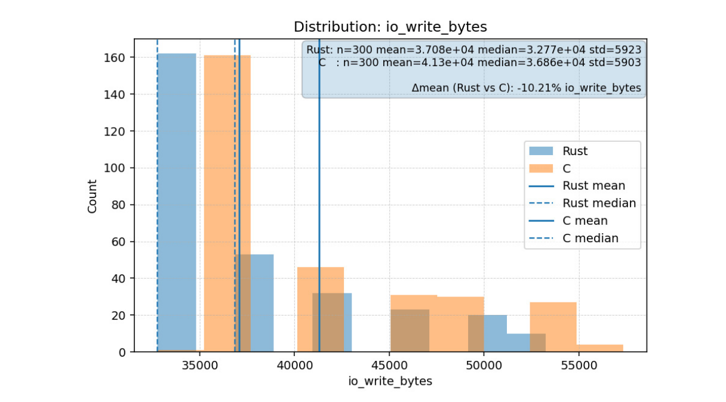
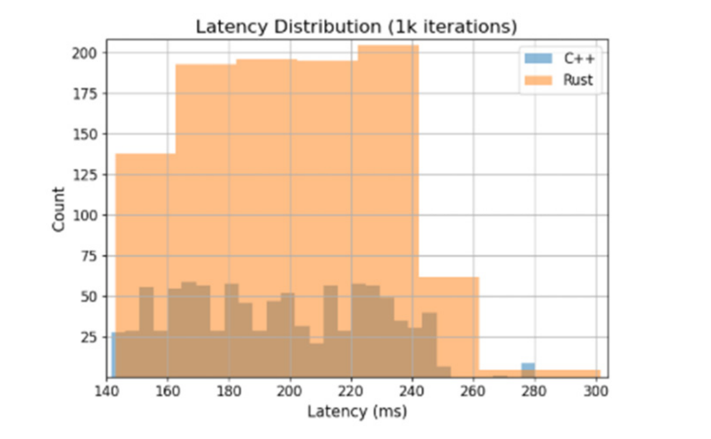
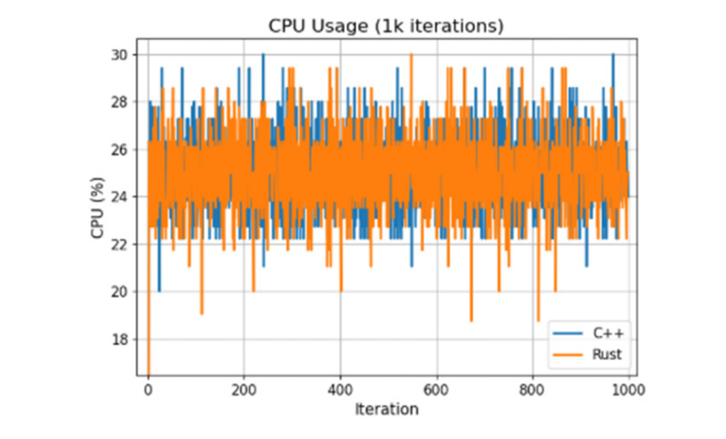
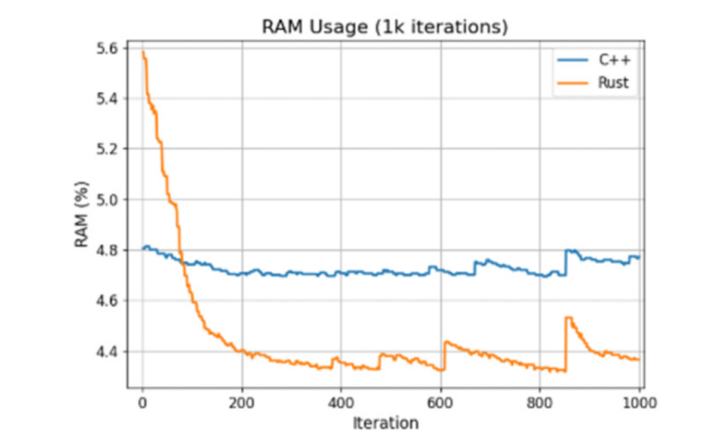
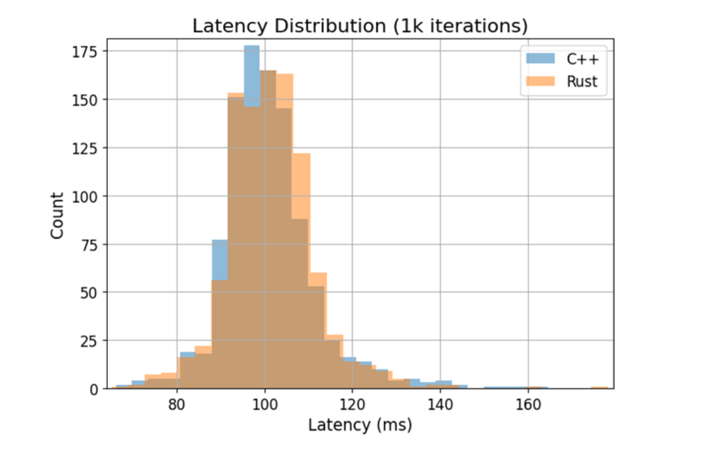
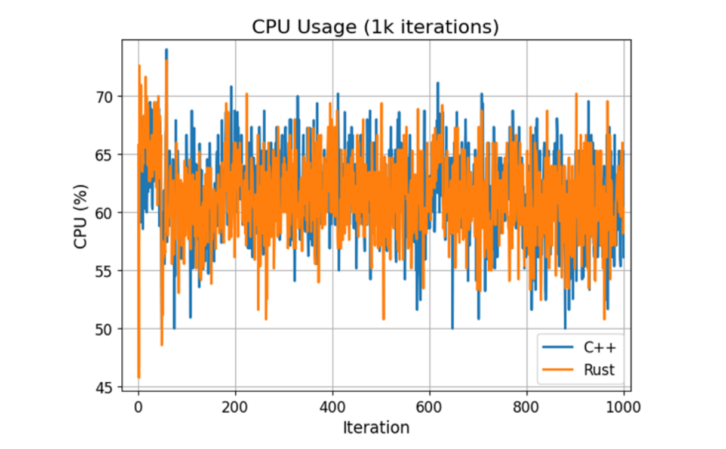
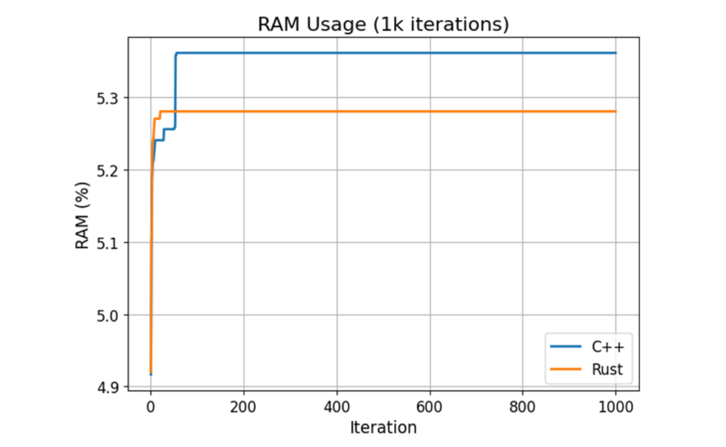
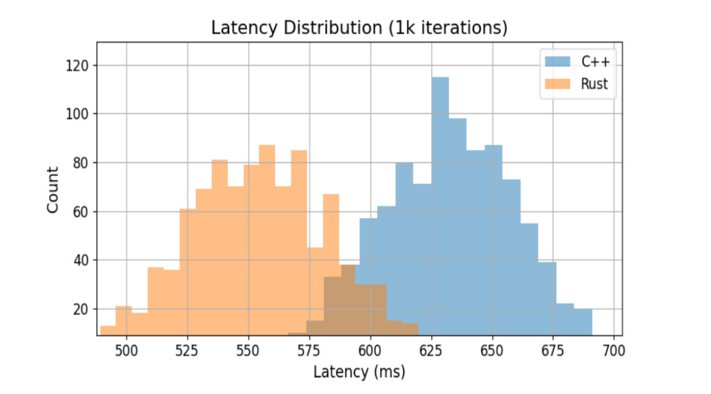
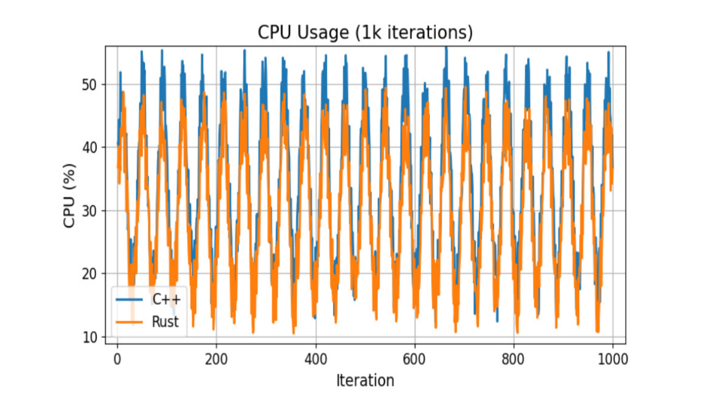
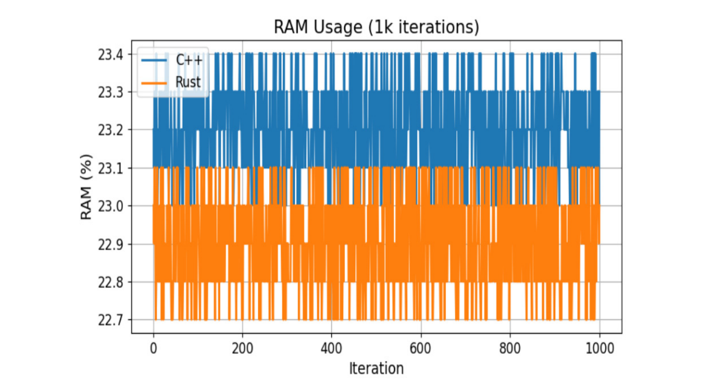
RTL2832U
Software Defined
Radio

GT-U7 GPS

Lenove FNK0056
Camera Module

BME280
Temperature and
Humidity Sensor

Results

Sensor	1k Loops	Ave Loop Time (ms)	Ave CPU (%)	Ave RAM (%)	Latency	CPU Usage	RAM Usage
WiFi	C++	48.08	23.72	9.75			
	Rust	55.20	23.36	14.00			
SDR	C++	1549980	5.12	5.36			
	Rust	1561130	5.11	7.46			
GPS	C++	195.87	24.98	4.73			
	Rust	197.89	24.97	4.45			
Camera	C++	101.02	61.6	5.35			
	Rust	101.58	61.2	5.28			
Temp & Humidity	C++	632.47	34.15	23.20			
	Rust	554.15	30.11	22.90			

Conclusion

Results implicate Rust being capable of delivering near identical real-time performance as C++ while bolstering memory safety and reducing CPU utilization. Generally, Rust yields analogous latency and throughput for all recorded sensors with minor increases in RAM usage attributable to the ownership model. Results from this study implicate Rust as a suitable technology for deployment in embedded and defense sectors where it can offer the same level of responsiveness as C++ all while enhancing security and reliability. Additionally, this study highlights Rust as a viable and contemporary alternative to legacy C systems without losing performance, while maintaining use of existing hardware, requiring only minor upgrades to RAM infrastructure at large scale applications.