Nathan Spelts

+1 (425) 408-2655 • nate@nathanspelts.com • Vancouver, WA
nathanspelts.com • github.com/FashionablyNate • gitlab.com/nspelts • linkedin.com/in/nathanspelts

TECHNICAL SKILLS

Languages: Python, Rust, C/C++, Swift, Kotlin, Javascript, Java, SQL, Ruby, C#, Haskell, PHP

Experienced With: CAN Communication Protocol, ARM Cortex, Git, SVN, JFrog, Jenkins, LaTeX, Raspberry Pi

WORK EXPERIENCE

Hyster-Yale Materials Handling, Fairview, OR, USA — Aug 2023- Present

- Software Engineer I
  - Developed and maintained embedded C solutions for forklift microcontrollers, ensuring performance, reliability, and safety in industrial applications.
  - Collaborated with a cross-functional team of engineers, testers, and managers to deliver secure and innovative embedded software solutions.
  - Contributed to technical documentation and quality assurance processes, enhancing the development lifecycle and ensuring compliance with industry standards.

EDUCATION

Washington State University, Vancouver, WA, USA — Graduated May 2023

- Bachelor of Science in Computer Science
  - GPA: 3.89 / 4.00
  - Coursework: Functional Programming | Object Oriented Programming | Statistics | Linear Algebra | Calculus | Assembly | Operating Systems | Data Structures | Databases | Software Engineering | Cryptography | Algorithms
  - References: Dr. Grant Williams - grant.s.williams@wsu.edu

PROJECTS

CAN Logger with Ratatui Interface — Proprietary Code

- Designed and developed a CAN logger application using the Rust programming language.
  - Displayed live and recorded CAN traffic for real-time debugging and post-analysis.
  - Reverse engineered the .blf file format to enable reading and writing industry-standard CAN log files.
  - Optimized CAN data processing for high throughput using async programming and efficient Rust techniques.

Rust Driver for Vector XL Devices — Proprietary Code

- Developed a Rust driver for Vector XL devices, enabling CAN message transmission and reception via a C foreign function interface.
  - Designed a robust API to streamline communication with Vector XL hardware for automotive integration.
  - Implemented functionality to transmit and receive CAN messages with high performance and low latency.
  - Employed comprehensive unit testing with mocking and integration tests to ensure reliability and maintainability.

RSA Signatures — gitlab.com/wsuv1/cs427_cryptography/rsa_signatures

- Implemented a simplified, highly insecure version of RSA in Rust for digital signatures, featuring two modes: ”sign” and ”verify”.
  - Developed the ”sign” mode to generate random primes, modulus, and totient, and calculate an encryption key. Utilized ELFhash for hashing the message, encrypted the resulting number with the encryption key to generate a digital signature, and verified its integrity.
  - In ”verify” mode, the program read inputs and matched signatures to hashes computed from the message, identifying forged messages. Key algorithms used included Miller-Rabin primality test, ElfHash, and exponentiation by squaring.