Josh Blatt

3B MECHATRONICS ENGINEERING

Personal Profile

I am a US/Canadian citizen studying Mechatronics Engineering at the University of Waterloo with strong interests in blockchain / decentralized technologies, embedded systems, controls, robotics, and AI.

Skills

Languages

- C/C++
- Rust
- Python
- MATLAB

Development Tools

- Git
- Vi / Vim
- CMake
- VS Code
- Make
- Visual Studio
- Linux
- CLion
- Eclipse
- PyCharm

CAD Software

- SolidWorks
- AutoCAD

Team Skills / Methodologies

- Agile
- Scrum
- JIRA
- TDD

Contact Me

Mobile: +1 416-577-7559
Website: www.joshblatt.ca
Email: jablatt@uwaterloo.ca
LinkedIn: www.linkedin.com/in/josh-blatt/
GitHub: www.github.com/joshblatt

Coursework

- Real-Time Operating Systems (C)
- Microprocessor Systems and Interfacing (C)
- Actuators and Power Electronics
- Sensors and Instrumentation
- Microprocessors & Digital Logic (Assembly)
- Data Structures and Algorithms (C++)
- Digital Computation (C++)

Education

University of Waterloo
BASc in Honours Mechatronics Engineering
September 2018 - April 2023

Interests

- Electronics
- Guitar
- Crypto
- Chess
- Rock Climbing
- Cutting Hair
- Snowboarding
- Video Games
- Rubik's Cubes
- TV Shows
- Music
- Hiking

Work Experience

EMBEDDED SOFTWARE ENGINEER

Deep Trekker | May 2021 - August 2021

- Independently designed and implemented new magnetic calibration algorithm in C using least-squares sphere fitting to calculate magnetic bias values and gyro data for sample point selection, reducing calibration error by 59x
- Programmed new state translation system for Pivot ROV, reducing state interdependencies by 72% and eliminating potential race conditions
- Implemented EEPROM I2C failure handling into the Revolution ROV's primary bootloader
- Reduced pub-sub message queue overflow occurrences by 18% by improving task prioritization

WINDOWS CORE SOFTWARE DEVELOPER

AMD | September 2020 - December 2020

- Refactored Radeon kernel-debug tool's code repository to make files easier to find and amended the CMake build architecture accordingly
- Added Radeon kernel-debug tool support to new areas of driver code using C++, allowing developers to easily extract new information from the live ASIC
- Wrote a C++ class to export Radeon telemetry data to MongoDB, rather than storing the parsed data in a CSV file as it was previously

EMBEDDED DEVELOPER

ecobee | January 2020 - April 2020

- Improved firmware for thermostat temperature slider to allow for a greater degree of parameter customization, including slider sensitivity, scroll speed, and inactivity distance
- Programmed new multi-speed fan QA PCB which was added to existing thermostat HVAC simulators, enabling automated testing of the hardware
- Developed numerous UI screens for the new thermostat multi-speed fan installation flow using C++ and created automated tests with Python to ensure its functionality

EMBEDDED TEST DEVELOPER

ecobee | May 2019 - August 2019

- Independently designed and programmed a Sensor Simulator using C++ to test the functionality of the SmartSensor and the SmartSensor for Doors and Windows, enabling the automation of 50+ test cases which previously took one week of manual testing
- Programmed Python script to parse and analyze sensor packet data to make product decisions, including how often and how far apart retry packets should be sent
- Designed integration test plan to verify functionality of production units of the SmartSensor, which was used for all customer-facing releases
- Assembled, soldered, and tested the hardware for HVAC simulator circuit boards used for QA

Projects

Mome (WIP) | December 2020 - Present | Repository

- Mome is a new C-like low-level programming language with easier syntax being developed independently using C
- Similarly to C, it will have features such as memory management, pointers, structs, and unions, with some added features including cleaner typedef struct syntax, automatic variable type assignment if not specified, and no semi-colon requirement
- Wrote the compiler's lexer, which breaks down the code being compiled into recognizable tokens to be interpreted by the parser, and implemented unit tests
- Created stretchy-buffer data structure (essentially a vector for C) using preprocessor macros and wrote unit tests to ensure functionality
- Currently writing compiler's parser and unit tests associated with it