Momen Ali

+1 (613) 501-1145 www.momenali.com momen.musa.ali@gmail.com linkedin.com/in/momen-m-ali
github.com/omen-mali

Education

Carleton University Expected June 2028
B.Eng., Computer Systems Engineering, 3rd Year of Study, 4 Co-op Terms *Ottawa, ON*
- **Relevant Courses:** Data Structures & Algorithms, Operating Systems, Embedded Real-Time Systems, Computer Architecture, Software Engineering, Electronics

Technical Skills

Languages: **C/C++**, **C#**, **Java**, **Python**, SQL, Bash
Frameworks & Libraries: **.NET**, NuGet, OpenXML SDK, JavaSwing, STM32 HAL, POSIX/pthreads, YOLOv8, OpenCV
Build & DevOps: **Maven**, Gradle, **Jenkins**, **Git**, **Azure DevOps**, JFrog Artifactory, CMake, SonarQube
Tools & IDEs: IAR Embedded Workbench, Visual Studio, VS Code, Eclipse, Android Studio, JIRA
Platforms & Systems: **Linux/Unix** (Ubuntu), **QNX Neutrino RTOS**, STM32H745 (Cortex-M7/M4), Android, Windows

Experience

Embedded Software Developer Co-op Sep 2025 – Apr 2026
Siemens Healthineers (Epocal) *Ottawa, ON*
- Developed protocol-level **C++11** firmware features on a dual-core **STM32H745** (Cortex-M7/M4) system, implementing **I²C** peripheral interfaces and **IPC**-based inter-core messaging within the hardware-layer software stack of the epoc Blood Analysis System
- Implemented **Bluetooth** BLE pairing fixes and structured protocol fault handling across **4+ fault subtypes**, delivering **∼15–20 firmware features and bug fixes** spanning IPC messaging and **Java/C#** host platform updates
- Reduced flash utilization from **96% to ∼80%** by offloading **∼70 unit tests** to **IAR Embedded Workbench** Simulator and evaluated **∼10 ARM simulation utilities** for peripheral fidelity and integration accuracy

Software Developer Co-op Jun 2026 – Present
Canada Revenue Agency (CRA) – TIDAS, Production Assurance Division *Ottawa, ON*
- Implementing **input-recording** features in the PA GUI, a **JavaSwing** testing and automation framework providing a single API to test applications across **JUnit, TestNG, and LoadRunner** backends
- Building editable **script-automation** generation capturing tester interactions for downstream QA reuse, and configuring **Maven** builds with **JFrog Artifactory** dependency resolution across cloud platforms within a **Jenkins** CI/CD pipeline

Projects

epoc Protocol Code Generator – Siemens Healthineers
C#/.NET, OpenXML SDK, Java, C++, JFrog Artifactory, NuGet
- Architected an automated **C#/.NET** code generator producing **100+ protocol classes per platform** across **C++, C#, and Java**, using the **OpenXML SDK** to parse Word/Excel specification documents into strongly-typed source
- Designed an extensible OOP architecture with abstract base classes and generic typed classes, delivering output via a **NuGet** and Java (`javac/.jar`) pipeline with versioned **JFrog Artifactory** artifacts, unifying protocol synchronization across 3 cross-team codebases

Wi-Fi Spectrum Analyzer GitHub
QNX Neutrino RTOS, C/POSIX, CMake, Raspberry Pi 4, pthreads
- Built a real-time **2.4/5 GHz** Wi-Fi sensor acquisition system on **QNX Neutrino RTOS** across a **5-module producer-consumer pipeline**, using **POSIX pthreads** and `wpa_supplicant` for live spectrum collection, interference analysis, and channel scoring
- Engineered deterministic **SCHED_FIFO/SCHED_RR** scheduling with **PTHREAD_PRIO_INHERIT** mutexes, and configured **CMake** cross-compilation targeting QNX aarch64 on **Raspberry Pi 4**

OS Process Scheduling Simulator
C, Python, Matplotlib, gcc, Linux
- Implemented a discrete-event CPU scheduling simulator in **C** covering **FCFS, Round Robin, and Extended Priority** algorithms, alongside demonstrative multi-process applications using **Unix system calls** (fork, exec) and POSIX semaphores
- Built a **Python/Matplotlib** visualization layer generating comparative metric charts and quantitative wait-time analysis across scheduling strategies

RoadSense – AI Road Damage Detection GitHub
Python, YOLOv8, OpenCV, PostgreSQL, Supabase
- Deployed a **YOLOv8** inference engine with **OpenCV** preprocessing detecting 4 road damage types from dashcam video in **16-frame batches** with weighted severity scoring, as backend lead on a 4-person team
- Processed camera footage into geolocated **PostgreSQL** records through a **6-step modular Python pipeline** with type-safe constraints and **JSONB** bounding box storage