US Citizen

Hasaan Haq

310-940-5016 | hhaq@ucsd.edu | linkedin.com/in/hasaanhaq | hasaanhaq.online

Education

University of California San Diego **Expected December 2026**

B.S. in Computer Engineering

Relevant Coursework: Operating Systems, Computer Architecture, Advanced Digital Design, Systems Programming, Digital Logic Design, Signals and Circuits

Summary

Computer Engineering candidate with experience in **system validation**, **board bring-up**, **hardware-software debugging**, and **C/C++/Python automation** across embedded platforms. Developed automated **validation and simulation infrastructure** for regulated medical-device systems, with hands-on experience debugging firmware, peripherals, board interfaces, and hardware using **oscilloscopes** and **logic analyzers**.

Skills

Programming: C, C++, Python, Bash, Assembly, SystemVerilog
Embedded & Systems: Board Bring-Up, Firmware Development, Peripheral Drivers, Multithreading, RTOS, Linux, STM32, NXP, ARM
Digital Design: RTL, FPGA, ModelSim, Quartus, Static Timing Analysis (STA)
Interfaces & Debug: UART, SPI, I^2C, CAN, GPIO, GDB, Oscilloscopes, Saleae Logic Analyzers, DMMs

Experience

Software Development Engineer Intern June 2026 – Sept 2026
Boston Scientific - Electrophysiology *Waltham, MA*

- Eliminated nondeterministic results in a multithreaded **Qt simulation application** by implementing timestamp-based data windows and mutex-protected QThread synchronization, enabling repeatable replay of recorded electrophysiology sessions.
- Built **Python and Bash automation** to concurrently capture I/O from physical hardware, replay recorded inputs through the hardware simulator, and compare simulated outputs against hardware outputs to verify behavioral fidelity and data integrity.
- Implemented XML parsing and automated assertions to validate electrode, frequency, channel, and sample integrity across physical hardware and simulated data.

Firmware Engineer Intern Sept 2025 – Dec 2025
Root Access *Remote / New York, NY*

- Performed **board bring-up** across STM32 and NXP platforms, developing C/C++ firmware modules and integrating sensors, actuators, and UART, SPI, I^2C, and CAN interfaces.
- Validated firmware generated by an embedded CLI, identifying incorrect register configurations, HAL-layer defects, peripheral initialization issues, and invalid pin mappings across multiple MCU families.
- Reverse-engineered embedded IDE build systems, compiler configurations, linker behavior, and project structures to support cross-platform build and project-generation tooling.

Embedded Systems Engineering Intern Oct 2022 – Sept 2024
Calpak USA, Inc. *Los Angeles, CA*

- Reviewed schematics and OrCAD wiring diagrams for embedded control systems, identifying hardware interface and wiring issues before production testing.
- **Debugged and reverse-engineered** UART and SPI communication on an STM32F429 plethysmograph using a Saleae logic analyzer, documenting device behavior and enabling reliable peripheral communication.
- Built Python tooling to automate **firmware flashing and production validation** for 1,000+ STM32F103 boards, reducing test time by **60%** and increasing throughput to **80 boards/hour**.

Projects

Autonomous Plant-Health Rover | Raspberry Pi 5, Linux, ROS 2, Python, C++, OAK-D Lite | **Project Link**

- Built a Linux-based autonomous rover on a Raspberry Pi 5, developing ROS 2 perception and control nodes to navigate GPS waypoints and execute automated plant-health inspection routines.
- Integrated cameras, GPS, environmental sensors, ranging sensors, and servo-controlled soil probes, coordinating hardware interfaces and sensor telemetry across the embedded platform.
- Developed ROS 2 planning and control nodes for autonomous waypoint navigation, including path generation, obstacle handling, and closed-loop trajectory execution.

Operating System Kernel & Virtual Memory | Nachos, Java

- Extended the Nachos kernel with thread synchronization, process management, and Unix-style system calls, implementing locks, condition variables, thread joins, file I/O, and multiprocess execution.
- Built a demand-paged virtual memory subsystem with page-fault handling, lazy loading, per-process page tables, swap-backed eviction, and clock-based page replacement using reference and dirty bits.
- Implemented page pinning and synchronized global memory management to safely support concurrent address spaces, page eviction, and kernel-to-user memory transfers under memory pressure.